The Introduction of Realism into SCADA Mimic Diagrams

using

Object Oriented Techniques and C++.

A Thesis

by

Mr P Kiernan

Submitted to Dublin City University,

School of Computer Applications

for the degree

of

Master of Science

May 1991

Dedicated to Sylvia whose
unfailing love and patience
made this work possible

ACKNOWLEDGEMENTS

I would like to thank the staff of the School of Computer Applications for their cooperation and help during this project and, in particular Mr J Doyle for his assistance I would like to thank Mr C Kerr for the financial contribution made on behalf of Datac Control to the project Most of all I would like to thank Dr M Scott for his positive encouragement, his many helpful comments and suggestions, and for his enduring patience

ABSTRACT

Supervisor Dr M. Scott PhD

Student Mr P Kiernan B Sc C Eng

This project shows how an interactive object based graphical user interface for a plant supervision or control application may be implemented using state-of-the-art software languages and tools In many current plant supervisory, control and data acquisition systems (SCADA) data presentation is limited to archaic character based graphics and text based prompts with little or no use of pointing devices With the emergence of object-oriented programming languages and graphics function libraries this project shows how a decisive upgrade in the graphics for these systems may be achieved, thereby bringing realism into plant mimic diagrams The mimic diagrams created in this project consist of a static background, acquired by the use of scanning devices or paint packages, and a dynamic background of icons, generated using object-oriented C++ classes The project is an object-oriented application and hence illustrates the object-oriented paradigm

The thesis introduces the area of supervisory, control and data acquisition systems It examines graphic standards and operating system options, and highlights the need for a user friendly extensible graphic interface to telemetry systems It also shows how object-orientation should provide for systems that are not only easier to extend and maintain but may also spawn parts which may be used for future projects The thesis, based on experience gained throughout the project, examines C++, classes, inheritance, problems associated with C++ environments and the dangers of product incompatibility It discusses graphic elements, such as bitmaps, icons and menus, and shows how object-orientation may be applied to them It expounds on real-time considerations and icon animation and details the full project implementation including compilation and memory management systems used Finally it points to the future, to the impact of object-oriented programming on technical management, to object-oriented databases and the object-oriented SCADA workstation of the future, and to changes imminent in C++ itself

Declaration

No portion of this work has been submitted in support of an application for another degree or qualification in Dublin City University or any other University or Institute of Learning

Contents

List of Figures

1 Object-Oriented Programming and SCADA Systems.

1.1 Introduction.

The first chapter of this thesis introduces the concepts of supervisory, control and data acquisition (SCADA) systems and the role that object-oriented programming plays in development of user interfaces for such systems It focuses on graphics packages and standards, and how realism may be introduced into SCADA mimic diagrams It highlights how this project makes the leap from procedural development to object-oriented development for SCADA user interfaces

SCADA systems cover everything from simple lowly data loggers to large distributed process control systems The underlying factor being the collection and display of plant data Ireland's closest neighbour, the UK, has already 400 telemetry scanners and 5000 outstations assigned to the major utilities Furthermore over the next five years, the UK Department of Trade and Industry "forecasts that the number of scanning telemetry systems will more than double and the number of outstations will quadruple by 1992" [LILL89] There are also the plethora of data acquisition arrangements that complement automation operations based on system profiles such as the Manufacturing Automation Protocol Demand for industrial data acquisition systems will continue to grow throughout the 1990's according to market researchers Frost and Sullivan Despite the fact that the SCADA industry is quite conservative, migration of intelligence away from the centralised computer to PCs and workstations is occurring rapidly The time is ripe for the move from procedural system development to object-oriented development Graphics are extremely important to SCADA users and are also an ideal candidate for object-orientation Moreover users are demanding friendly object-oriented and icon based systems

With the graphics packages and object-oriented programming systems (OOPS) now available a decisive upgrading in the graphics available for SCADA systems should bring more realistic mimic diagrams to users screens Object-orientation should provide for systems that

are not only easier to extend and maintain but may also spawn reusable parts for future projects

A telemetry or SCADA system typically consists of a hierarchical network of data collection stations, communicating over leased lines, the public switched telephone network or radio This communications network is generally terminated in a central station The central station is used for monitoring the real time data fed through the entire system. Other central station functions could include data logging, statistical data analysis and plant control Generally the network topology would be configurable and would depend on the structure of the system or plant under supervision Network complexity could range from a single logging system to a complex tree structure of data collection stations, communications substations and a central station with a standby backup system

A project to upgrade an existing central station is in progress A part of this project [DATA89] may include decentralisation of certain heretofore central station operator facilities This may be achieved by the provision of a PC network for the connection of SCADA graphic workstations Mimic diagrams of plant and machinery are the key element in plant supervision The mimic diagram realism project has produced, a number of methods for achieving realistic static mimic diagram backgrounds, a hierarchy of dynamic object oriented icons representing plant transducers and actuators, and a framework for mimic generation It is envisaged that the work carried out on this project together with network technology, database facilities and future object-oriented graphics development may form the basis for a graphic SCADA workstation to be used as an integral part of a full SCADA system

1.2 The Current Status.

The present generation of SCADA systems with distributed intelligence allows data access at nodes throughout a telemetry network. However in many systems the data presentation is limited to character based graphics with a limited character set and text based prompts with limited or no use of pointing devices. Figures 1.1 and 1.2 show character based mimic diagrams from a water supply monitoring and control system. The migration from proprietary outstations to workstations and PCs, and the use of PC networks now enables these archaic unfriendly interfaces to be replaced with new graphics based interfaces. Hence the way is clear for the emergence of the object-oriented SCADA graphics workstation. These workstations to be positioned either at the data acquisition site or remote from the site and communicating to the site station via a telemetry network. The key elements in the new interface are object-orientation and realism.

SCADA systems are generally real-time systems and as such the question of timing and system throughput becomes significant. However "most operations are not time-sensitive, and the few most critical operations are probably going to be done in native machine language to get the most throughput" [FORD90]. As the message passing overhead with an object-oriented language is generally no heavier than an equivalent procedure call in a non-object-oriented language, according to Thomson of Object Technology International [FORD90], then the use of object-oriented languages for real-time systems should not be a problem.

A limited number of icon based SCADA workstation products have just begun to appear on the market. The icons are generally very primitive with poor animation features. They promise a user level object-oriented style of environment but are not written in an object-oriented language and hence do not have some of the inherent features of a real object-oriented development such as high level of maintainability and reusability.

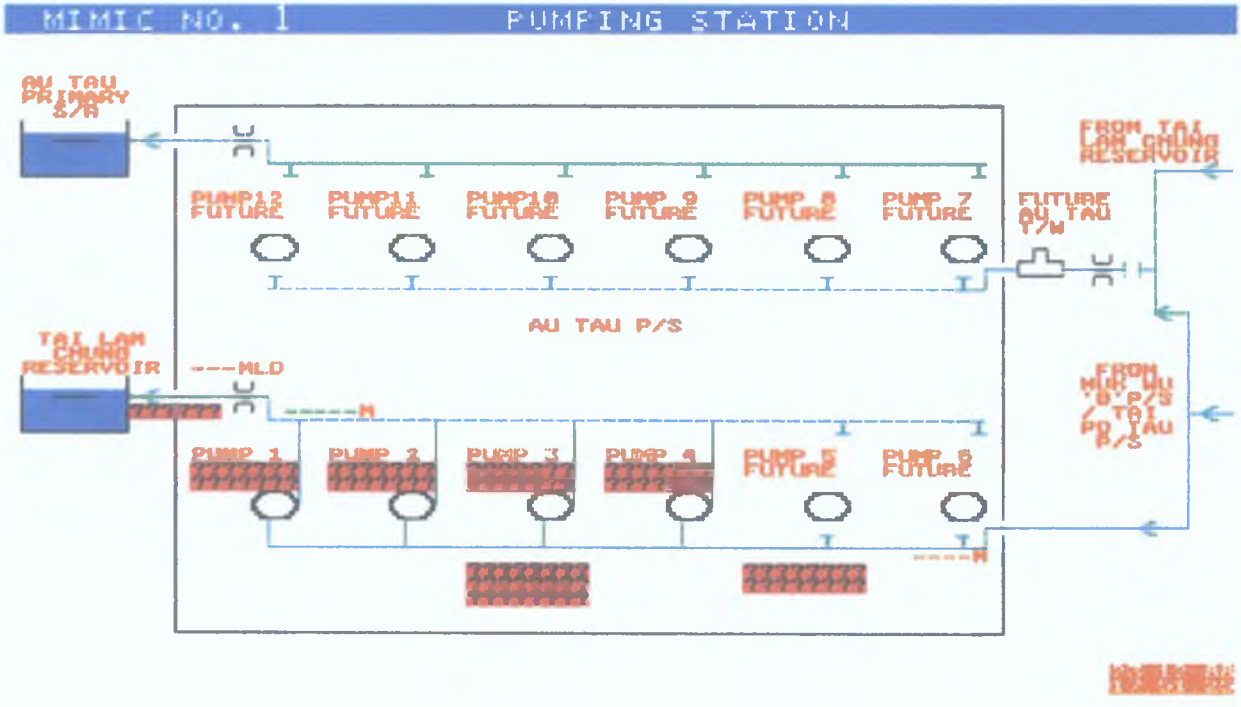


Figure 1.1 Pumping Station.

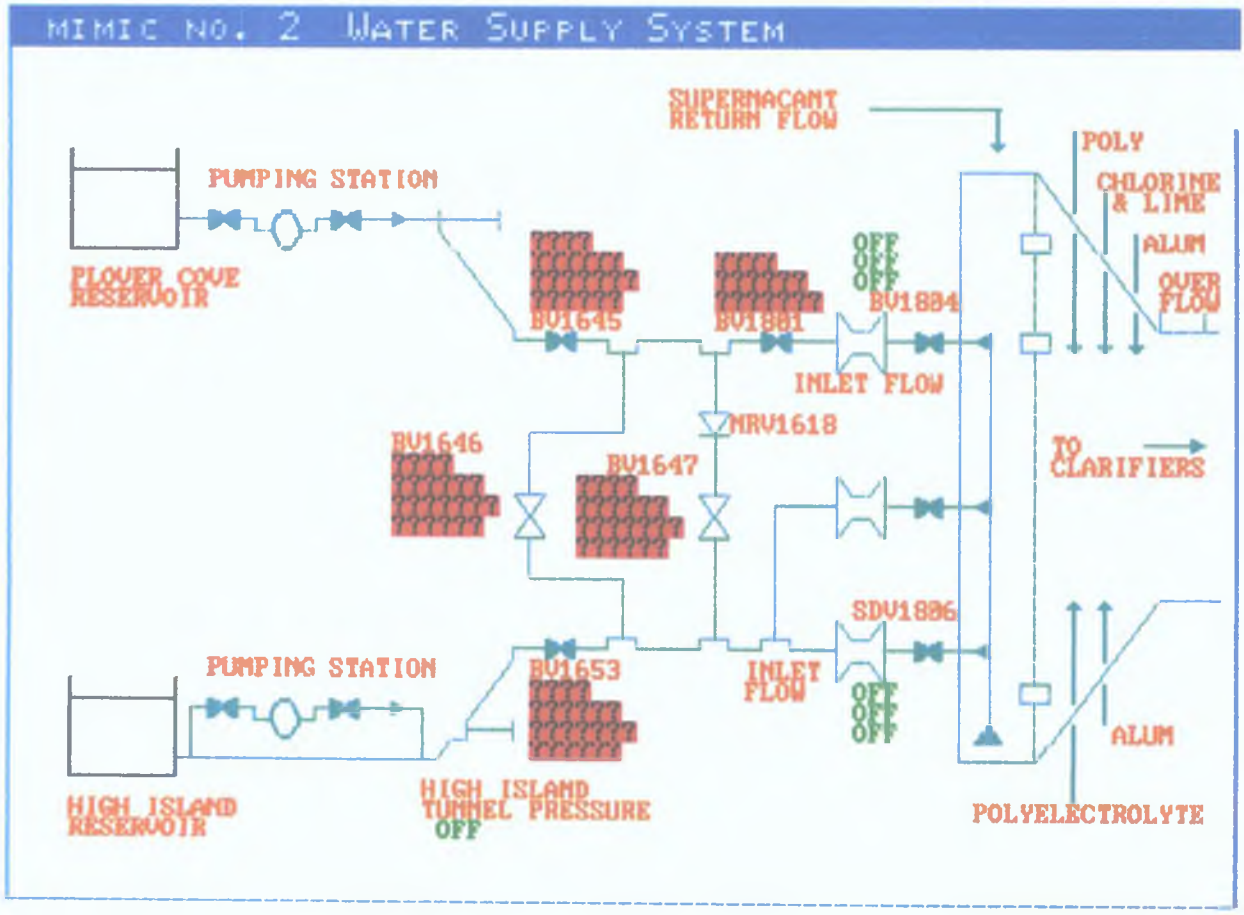


Figure 1.2 Water Supply System.

A major feature with SCADA systems is their inherent commonality SCADA system developers continually find themselves reusing procedures from past projects. This situation is ideal for the object-oriented paradigm which centres around the generation of libraries of reusable software components which can just be plugged-in to new applications Coupling the idea of reusable software components with the new graphics packages now available to form a library of reusable graphic objects, would be very useful in the development of SCADA systems

1.3 Graphics and Standards

"We are drowning in information and starved for knowledge computer graphics may be a lifebelt to help us stop drowning" [MARS87] This is very applicable to data collection systems Long lists of transducer identifiers and their status or value do not provide for rapid transfer of information to a plant operator

Marshall's book 'Computer Graphics in Application' outlines many of the graphic packages available [MARS87] The only *de facto* standard available in computer graphics is the Graphic Kernel System (GKS) [HOPG83] It was designed over six years by the International Standards Organisation (ISO) and became a standard in 1983 GKS is defined independently of programming languages Its main objective is the production and manipulation of pictures in a way that does not depend on the graphical device or computer used "GKS is a device-independent kernel system with all the graphics commands being stored in a device-independent format in a metafile" [OCON89] GKS uses device drivers to convert the device independent commands into device specific commands, hence input/output devices can be changed without reprogramming The result is a system which is device independent but has a slow response [OCON89] GKS is a large system because it must cover a very wide range of operations and devices Recent graphic products which do not conform to the GKS model have proved themselves to be far superior to the GKS system

An example of such a product is the Halo graphics package [HALO88] This package provides a library of graphic primitive commands which can be called from a C program It also provides the capability of providing a captured video frame image as a mimic background A graphic interface could be designed by encapsulating graphic primitives in segments and using control blocks and linked list structures to string segments together to form icons

The basis for such system development is expounded on by [FOLE84] Certainly an object style user environment could be constructed in this way, however it would only approximate the power of a truly object-oriented graphic development.

Other graphic packages like Zortech's [ZORT90b] Flash graphics are an integral part of a larger programming environment These packages provide a more homogeneous interface to the binding language, and like the Halo package provide a rich set of graphic primitives Most graphic primitive libraries provide device drivers for a variety of monitors and pointing devices The use of such graphics packages negates the necessity of writing low level routines to draw graphic primitives In addition the Zortech graphics package is supported by the C++ object-oriented programming language

The choice of IBM PCs or compatibles as graphic workstations for SCADA applications has come about because of low hardware cost, ease of ruggedising the hardware, availability of relatively cheap software development tools, compatibility with other hardware and software, and ease of expansion [TINH88] "Because of the true 'open bus' nature of the architecture, it has become a standard platform for industrial measurement and control" [GERO90] PCs represent the optimal choice for stand-alone systems with signal counts of less than a few thousand and as workstations to larger systems such as DEC VAX under VMS, or VME systems under OS/9 There is no doubt that whether it is real time monitoring of instrumentation, transducer control or off-line plant analysis, PC graphics are the key to low cost automation As seen at the Birmingham Control and Instrumentation Exhibition [CIEX89] there are procedural language based products on the market However these products are likely to be soon surpassed by far more flexible object-oriented systems

1.4 Object-Oriented user Interfaces and Realism.

Showpieces like The Open Software Foundation (OSF)/Motif Graphical User Interface [DEIN90] and Hewlett Packard's NewWave Environment [SHOW90] bear witness to the power and flexibility of object-oriented graphical user interfaces Most users are busy enough maintaining expertise in their own area without having to become computer experts as well The user interface should be unambiguous and easy to use even for casual users without sacrificing flexibility and effectiveness for experienced users Clever use of pointing devices, menus, windows and information templates can help achieve this goal Of particular importance in the SCADA area is that plant information should be imparted intuitively to the user The traditional system with line-diagram mimics and values scattered over the screen, or tables of values failed to achieve this

The graphical interface developed in this application allows the user to select a background picture or mimic from disc and then to build up a set of dynamic icons on top of the background The background would typically represent some plant or part of a plant The interface is a true OOPS interface programmed in object-oriented style via C++

Visual realism is introduced by using scanned images from plant photographs or schematic diagrams using an AT&T scanner [ATTS88] and the associated SCANWARE software [SCAN88] The scanned images can be edited by using ZSoft PC-PAINTBRUSH [PCPB88] allowing them to be enhanced or allowing backgrounds of partial images and schematic diagrams Alternately the scanned images can be converted to Media Cybernetics Halo graphics format and can be edited using the Dr Halo III interactive graphic editor [DRHA87] to produce, as above, enhanced images or backgrounds of mixed images and schematic diagrams A program has been developed as part of the project to convert AT&T scanner files [EXE89] and Media Cybernetics files to background files compatible with Zortech Flash Graphics

Schematic mimic diagrams from other systems can be converted into Media Cybernetics format using the memory resident GRAB facility of Dr Halo III, and from this format converted as above into files compatible with Zortech Flash graphics

The interface uses a two button mouse pointing device and keyboard for input Menus, windows and templates are used to display data and for input of data Dynamic icons which mimic real world transducers can be drawn, erased, dragged across the screen and overlaid The dynamic mimic is generated by choosing a set of icons to match signals which would be measured in the plant or part of plant Operator input icons can also be included for direct plant control The interface provides a rich set of dynamic icons implemented in an object-oriented language Typical SCADA applications in the water, gas, electricity and oil industries would use a base set of dynamic icons and a customised set for each different application In true object-oriented system design fashion the accumulation of a full set of reusable dynamic icons for a wide range of SCADA applications would occur over several projects, with the existing set acting as a strong base for further development

A mimic diagram consists of a fixed background and a set of dynamic icons Figures 1 3, 1 4, 1 5 and 1 6 are screen dumps from the DESIGN EXE program developed in this project Figure 1 3 shows a mimic background consisting of a methane gas plant schematic, scanned with an AT&T scanner and converted to a Flash Graphics compatible file, with a mimic foreground of various icons Figure 1 4 shows a converted scanned picture of a pumphouse overlaid with dynamic foreground icons, while the mimics of figures 1 5 and 1 6 are converted scanned pictures of a water tower and treatment plant overlaid with icons for plant supervision and control In some instances a simple text background as opposed to a scanned image or schematic background may be sufficient A mimic of this type is shown in figure 1 7

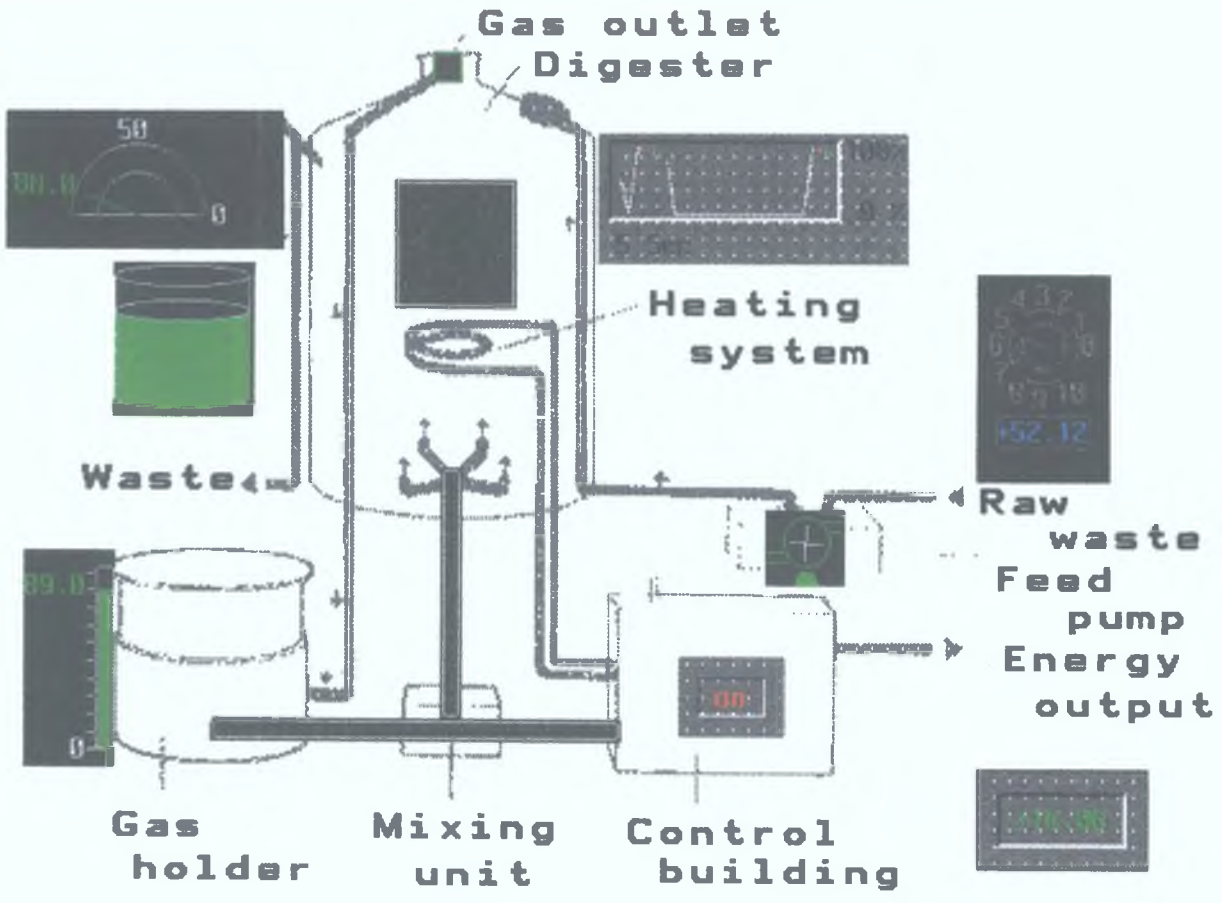


Figure 1.3 Methane Gas Plant

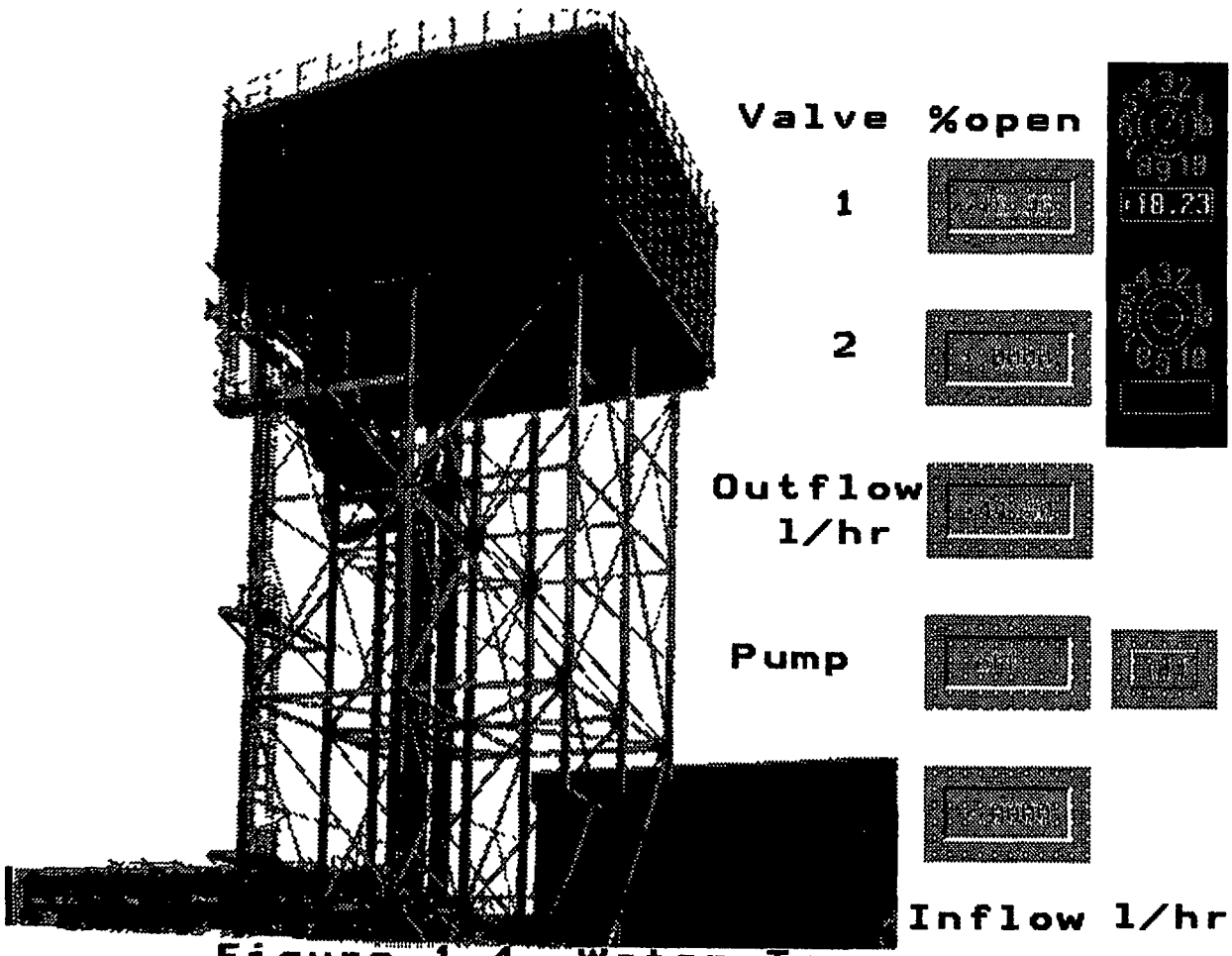


Figure 1.4 Water Tower.

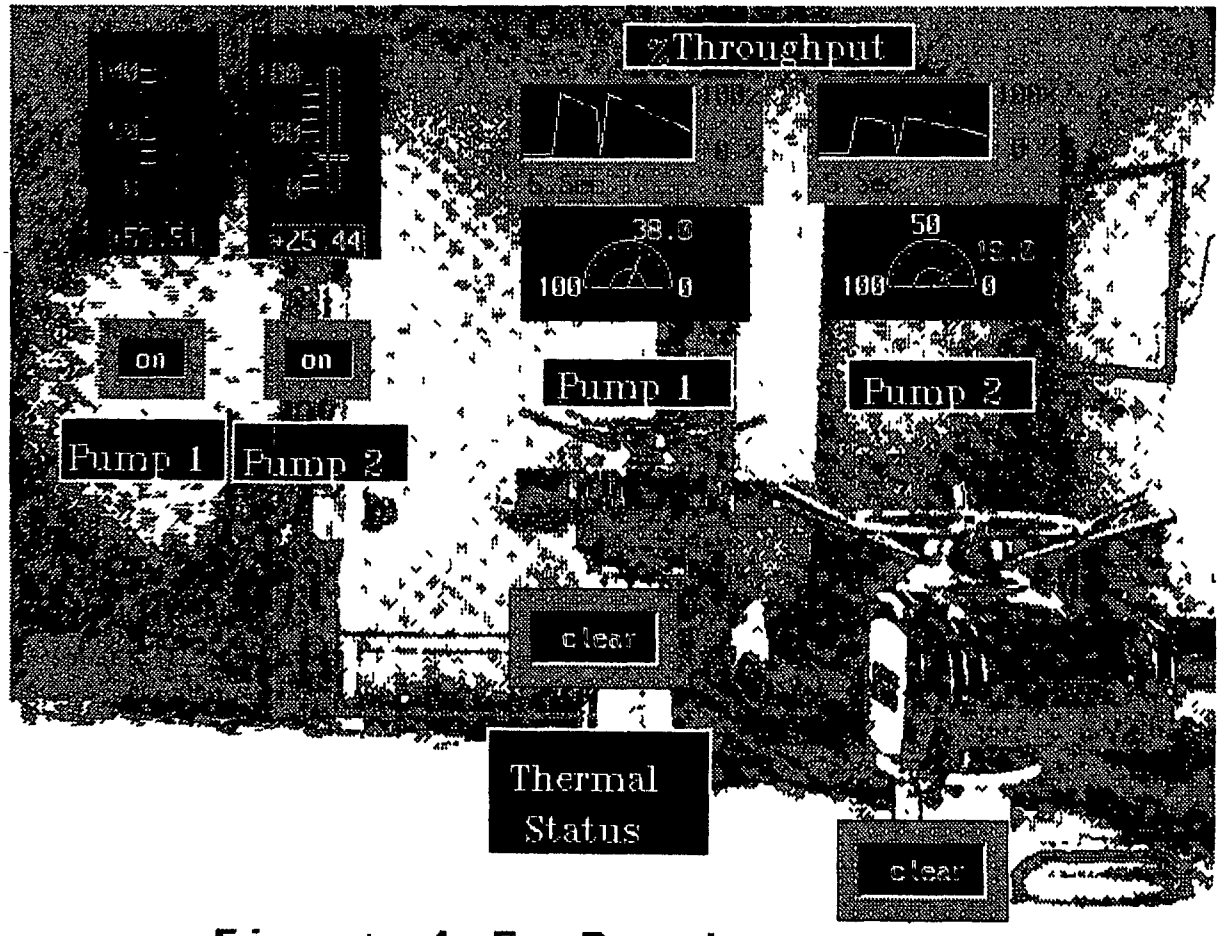


Figure 1.5 Pumphouse.

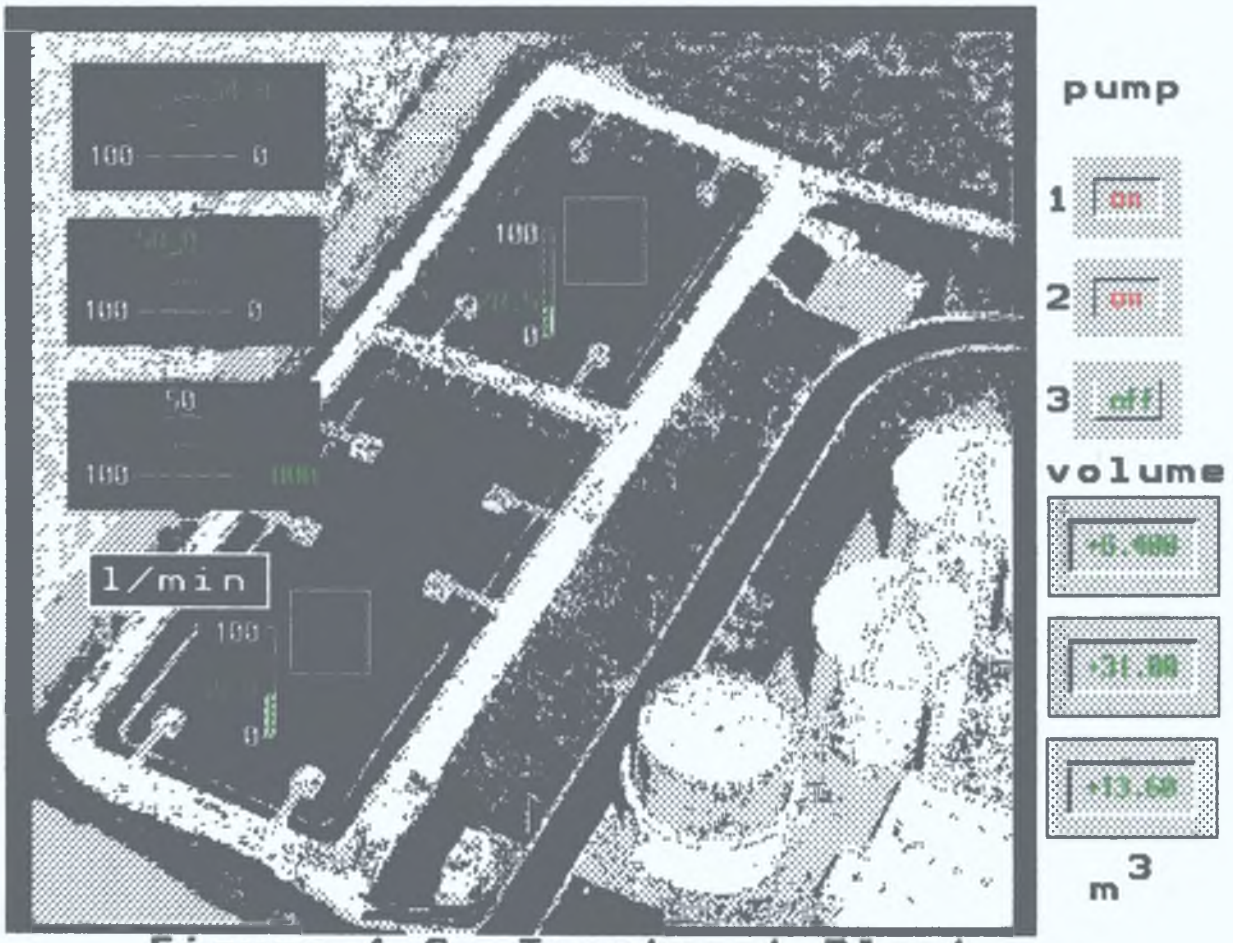


Figure 1.6 Treatment Plant.

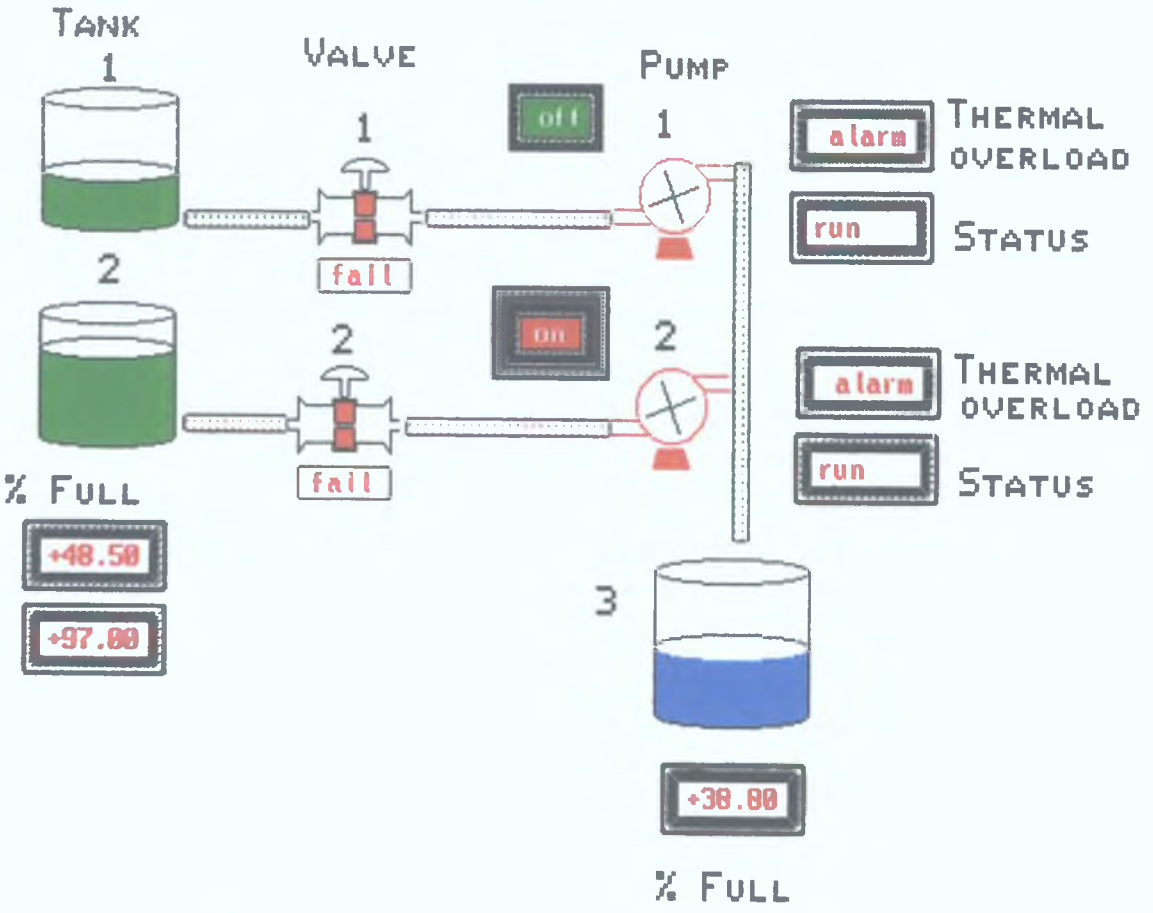


Figure 1.7 Processing Plant.

The mimic diagram can be animated with simulation data in order to view the operation of the icons Because this is a real-time application attention has been paid to ensuring that all icons are updated within a reasonable length of time with respect to the time constant of the plant under supervision The typical target applications for this project have a time constant resolution in the order of seconds

1.5 Operating Systems and Executives.

There are now a wide range of options facing a system developer when it comes to choosing an operating system or executive on which to run the proposed application Possibilities included UNIX style systems for PCs such as PCUNIX and PC/1X [CHIR87], operating systems such as FLEX [FLEX88], QNX [QNX88], OS/2 [OS288], and IBM DOS [DOS88] and DOS executives such as Microsoft WINDOWS [WIND87] to name but a few The choice depending on cost, availability and compatibility of object-oriented and graphics software tools for the various environments, the target hardware configuration, complexity of implementation and end user requirements

Bearing in mind the requirement to target development for a low cost SCADA workstation to be run on a non expensive and popular platform the cost of FLEX and QNX implementations invalidated their use in this project On the basis of the lack of ready availability of object-oriented tools and the relatively high cost of procedural tools for UNIX type systems, PCUNIX was deemed unsuitable for this application

Decisions made in the course of this project in assessing the suitability of WINDOWS, MSDOS and OS/2 operating systems were heavily dependent on the development and the compatibility of various tools, and will be outlined in the following chapters of this thesis dealing with the project implementation

User requirements must also feature in the choice of operating system However the visibility of the operating system depends on the implementation and on other application programs which may be required within the context of the overall application These issues are discussed at length in the forthcoming chapters

2. C++ Tip of the Silver Bullet

2.1 Introduction

The purpose of this chapter is to highlight the Object-Oriented paradigm, to illustrate the differences between Object-Oriented and procedural Languages and to show how C++ classes, inheritance and memory allocation are used in this project

Criticisms directed towards the C++ language are examined in the light of experience gained from the project

Finally the focus is turned on Zortech C++, the C++ compiler and development environment used in the implementation of this project

C++ is defined as a superset of C and "is a general purpose programming language designed to make programming more enjoyable for the serious programmer" [STRO87]

2.2 Object-Oriented versus Procedural Languages

It is in the very nature of procedural languages to emphasise procedures and separate data structures Hence although a data structure may be defined only once, it may be referenced by many procedures Making a change to that data structure may spawn a plethora of modifications to those procedures. Procedural language based systems do not like change However change is essential for software upgrade or reuse The key idea in the object-oriented approach is that data and procedure are not separated Data and object-oriented procedures or methods are encapsulated in a structure known as a *class* Classes are the building blocks of an object-oriented system

"A software industrial revolution based on reusable and interchangeable parts will alter the software universe" [COX90] Everything in the procedural world is composed of routines and programs that may never have been seen before and may never be seen again Procedural language based systems generation can be compared to a cottage industry where everything is custom built Such a system is time and resource wasteful, and very difficult to schedule Also [WILS90] suggests that defined requirements may not be always implemented correctly and that requirements themselves may be incorrect and are often subsequently changed over time In many cases the cost of maintaining software far exceeds the cost of writing it

Object-oriented development can be compared to a highly organised production system where the end product has been constructed from pre-assembled software units This **encapsulation** means that software integrated circuits can be used to produce software cards, which can be used to produce software racks and hence an entire system can be assembled This means that the class or basic unit must be water-tight and used as a black box.

Classes encapsulate their data and methods. An instance of a class is known as an *object* Communication to and from the object is via its methods The implementation of the method is invisible to the outside world, and hence can be changed or modified without affecting its users The data definition, within the object, determines whether the object's data may be *private* to itself or *public* to its users

Key to this software revolution is the idea of reusable objects It is possible to reuse modules of a procedural system along with new modules to produce a new system In fact good programmers already use the principles of object-orientation in structuring their systems However in most cases the language does not support it, and this makes it difficult An object-oriented language is no panacea but it does actively support encapsulation and the class structure which is essential to the production of reusable software objects Encapsulation defines a boundary that encompasses the scope of the objects software, the object interface methods or protocol and a protected internal implementation Reusability, maintainability and extensibility may be said to be the key goals of object-oriented development By utilizing reusable software components the time and cost of development may be significantly reduced "Commonality must be actively sought when the system is designed, both by designing classes specifically as building blocks for other types and by examining classes to see if they have similarities that can be exploited in a common base class" [STR088]

Maintainability costs may be reduced by the ability to localise modifications to the implementation of one or more classes and by the ability to inherit facets of one class from another, rather than re-writing existing classes This inheritance concept becomes a very powerful tool especially when dealing with object-oriented languages which support multiple inheritance In this case a class may inherit facets from many other classes An object-oriented system will contain many classes in an hierarchical tree structure

At the top there will be one or more very general classes, from which classes may be derived The classes at the bottom will tend to have specific applications, although they will inherit characteristics from their respective parent classes

Because of the encapsulation principle an object oriented system may be tested by carrying out incremental testing of each class to be used in the overall project When a hierarchy of classes is used, then parent classes should be completely tested before derived classes are tested Both structural and functional (black box) tests are required for objects In a procedural system testing of a module will typically only start when the module containing several procedures or functions has been completed, whereas in an object oriented implementation, testing can commence once the object is complete Hence unit testing may be considered much earlier in the object oriented environment

In an ideal situation all classes should be path tested, particularly for critical applications Typically the class testing should be carried out by the class developer who will know the detailed internal implementation of the classes As a result "an independent tester or developer who is not involved in the design and generation of code for a specific class may find it difficult to perform adequate testing of the class" [FIED89] The idea that classes can be reused as base classes without retesting of inherited code is contended by [PERR90] Perry and Kaiser describe criteria for adequate testing and show why inherited code may need to be retested

Object-oriented software design focuses on the translation of the physical or logical elements of a real world system into classes This is the main challenge of object-oriented design Objects lend themselves easily to the process of mapping the elements of real world systems to the elements of a software system and hence the real world problem decomposition is far more intuitive than in a procedural system design Object-oriented development is fundamentally different from traditional procedural methods in

which the primary criteria for decomposition is that each unit or module represents a step in the overall process An object is an element whose behaviour can be characterised by the actions it may suffer and the actions it may require of other objects Instead of breaking the proposed system down into units or modules that denote operations, the system is synthesized into objects that exist in the real world model of the system In this project the real world system model is broken down into iconic objects representing transducers, menus, information I/O windows and mouse models. Among the examples of object-oriented design available, Booch [BOOC86] shows how a cruise-control system synthesis from objects such as wheels, brakes, accelerator and engine may be achieved The steps in object-oriented development consist of

Identifying the objects,

Identifying the operations suffered by and required of each object,

Establishing the visibility of each object to other objects,

Establishing the objects interface, i e its methods which form the objects protocol, and

Implementation of the objects

Methodologies have recently been proposed for object-oriented systems analysis (OSA) An example of such a methodology which imposes a natural modularization on the system model through an emphasis on objects, where the object has a one-to-one correspondence with an actual object in the system, is given by Kurtz, Ho and Wall [KURT89]

The concepts of object-oriented programming are quite different than those of procedural programming It is not just a syntax difference, there is a completely different philosophy involved which makes the learning curve quite steep Hybrid languages like C++ provide a transformation from the procedural world and C to the

object-oriented environment With practice comes perfection, and it is only with practice that objects can be written which properly utilise the inheritance feature of object-oriented languages Similarly it is only with experience that objects can be written for a specific application and simultaneously be written in such a way as to be useful in future projects The process of developing a library of reusable components would typically stretch over several projects

"Of all the monsters that fill the nightmares of our folklore, none terrify more than werewolves, because they transform unexpectedly from the familiar into horrors For these, one seeks bullets of silver that can magically lay them to rest The familiar software project, at least as seen by the non-technical manager, has something of this character, it is usually innocent and straightforward, but is capable of becoming a monster of missed schedules, blown budgets, and flawed products So we hear desperate cries for a silver bullet-something to make software costs drop as rapidly as hardware costs do" [BROK87] The silver bullet has arrived in the form of object-oriented programming It is "no longer the wave of the future" [TAZE90] and is rapidly permeating into all areas of programming

2.3 Classes and Inheritance.

The architecture of an object-oriented system is dependent on the methods that define the operations that can be performed on the objects internal data, not on the objects internal structure itself "The binding of underlaying data with an associated set of procedures and functions that can be used to manipulate the data is called **encapsulation**, the inaccessibility of the internal structure of the underlying data is called **data hiding.**" [WIEN88]

A *class* is a user-defined type An example from this project is the *cadshape* base class This class is the parent for many other classes including classes which generate icons representing plant equipment and measurement devices, and classes which generate text windows

The *cadshape* class header file, CADSHAPE HPP, is now expanded with explanatory text provided between the various sections of the file

To avoid multiple declarations when this class is included in various programs and other classes

```
#ifndef CADSHAPE_HPP
#define CADSHAPE_HPP
```

The **methods** of a class use various system functions

```
#include <time.h>          // system time functions
#include <stdio.h>         // standard I/O
#include <fg h>            // Zortech Flash graphics functions
#include <dos.h>           // Dos functions
#include <stdlib.h>        // Standard lib functions
```

The class is declared by placing the keyword *class* before the class name required·

```
class cadshape {
```

A *friend* function or class has the right to access the private part of another class Here the *template* class has access to the private data of the cadshape class This means that all of template's methods or member functions are friends of the class cadshape and can access its private as well as its public section A friend declaration can be placed in either the private or public part of a class declaration The *template* class was developed in this project for displaying and editing class data

```
friend class template;
```

Access to data in the protected area is restricted to the methods of this class and derived classes Non-member functions cannot use this data Part of the protected data section of the *cadshape* class is given below It contains data members required by the *cadshape* class itself and its derived (or child) classes

```
protected
unsigned x_center, y_center,      // (x,y) position
float mul_saved;                  // Expansion factor
fg_box_t max_extent_box,          // Maximum object area
fg_box_t extent_box,               // Extents box
fg_box_t test_box,                // Extents test box
fg_box_t background_box,          // For background behind object
float xmintx,ymintx,              // Test extents
float xmaxtx,ymaxtx,              // Test extents
int xmintx_outside,xmaxtx_outside; // Outside extents flags
int ymintx_outside,ymaxtx_outside, // Outside extents flags
int status_extn;                  // Extents active or off
int extent_overlap;               // Extents overlapping
unsigned int identifier,           // Cadshape
int cad_type;                     // Cadshape type
float pi;
```

The declaration *float pi* cannot be defined as a constant as the class construct only allows declarations, not definitions at this point A value can be assigned to *pi* in the class constructor

The public section can contain both methods and data. The methods form the class interface or protocol A struct would be simply a class where all members are public The public section is defined as shown below, by using the *public* keyword

public.

Virtual functions allow declaration of functions in a base class that can be redesigned in each derived class The keyword *virtual* indicates that the function *pdraw()*, as given below, can have different versions for different derived classes and that it is the task of the compiler to find the appropriate one for each call of *pdraw()* The type of the function is declared in the base class and cannot be redeclared in a derived class A derived class that does not need its own unique version of the virtual function need not define one, in which case the base class virtual function is called for both the base and derived class Virtual functions must have some definition in the base class, even if it is just empty

```
virtual void pdraw() {}              // Draw the object
virtual void panimate() {}           // Animate the object
virtual void perase() {}             // Erase the object
virtual void pextent_pgen() {}       // Generate objects extents
virtual void pextent_pdraw() {}      // Draw extents box
virtual void pextent_perase() {}     // Erase extents box
virtual void analog_pinput() {}      // Operator analog output
virtual void digital_pinput() {}     // Operator digital output
```

A member function with the same name as the class is called a **constructor** A constructor may not be a friend or virtual function It is used to construct new instances or objects of the class type Where there is a base and derived class, the base class is constructed first A return value cannot be specified for a constructor and it cannot use the return statement. The lifetime of an object is limited to the scope in which it is created. The constructor for the *cadshape* class is an empty function and is shown below

cadshape( ) {}

A member function of a class *cadshape* named *~cadshape* is called a **destructor** A destructor cannot take arguments and cannot specify a return value It is used to deallocate memory from the free store which is allocated by the constructor A destructor can be called explicitly, but must make explicit use of the class pointer -> or class member operators The destructor for derived classes is executed before the destructor for the base class Care should be taken in the allocation and deallocation of free store and generation and deletion of file buffers across base and derived types This is discussed in the context of the use of these facilities in the implementation section The destructor for the *cadshape* class is shown below

virtual ~cadshape( ) {}

The implementation for class member functions can be placed in the hpp **header** file or in a separate cpp file If it is in the hpp file then they are compiled as part of the program into which the hpp file is included The cpp files are compiled separately and linked into the application

The *cadshape* header file is now closed as shown below

},

The *#ifndef CADSHAPE_HPP* statement from the beginning of the file, used to avoid multiple declarations, is also closed as shown below

#endif CADSHAPE_HPP

Some examples of class methods defined in the *cadshape* class header file are given below

void idmod(int new_id){ identifier=new_id, }

```
unsigned int idinq(){ return identifier, }

unsigned * getcoords(){
        coords[0]=x_center, coords[1]=y_center,
        return coords,
        }


void pextent_set(int extenth_onoff) {status_extn=extenth_onoff;}

int pextent_view() { return status_extn;}
```

A class may be **derived** from one or more base classes The derived class inherits the properties of the base class, including its data and methods, public and protected The derived class can override base virtual functions and have its own data and member functions Classes can be derived from derived classes forming a class hierarchy The *square* class has been derived from the *cadshape* class in this project The *square* class header file is now expanded with explanatory text provided between the various sections of the file

As for the *cadshape* class, to avoid multiple declarations, the *square* class uses

```
#ifndef SQUARE_HPP
#define SQUARE_HPP
#include "cadshape hpp"
```

The class declaration takes the form

```
class derived_class_name · public parent_class_name{}
```

where the keyword *public* is optional and *parent_class_name* must be previously declared

```
class square . public cadshape {
```

The *protected* data section for classes derived from this class

```
protected:
   fg_box_t small_box;              // square
   float height,width;
```

Methods (or member functions) with public access form the objects interface, and are the means by which the object can be manipulated by non-member functions, non-friends and classes which are not derived from this class The public section is now opened thus

```
   public
```

Unless redefined in the derived class, as below, members of the base class can be referred to as if they are members of the derived class The scope resolution operator can be used to refer to a base member explicitly In this case the hpp file contains the declarations only, the definitions being kept in the cpp file where an #*include square.hpp* statement would be used

```
   void pdraw(),                    // Draw the object
   void panimate(),                 // Animate with raw data
   void pextent_pgen(),             // Generate objects extents
   void pextent_pdraw(),            // Draw extents box
   void pextent_perase(),           // Erase extents box
```

A *constructor* is called whenever an object of a class with a constructor is created An object can be created as a global or local variable, through explicit call of a constructor or through the use of the *new* operator It could also be allocated as a data member of another class

The base class constructor arguments are specified in the definition of the constructor for a derived class In this case the *cadshape* constructor takes no arguments The constructor for the *square* class is given below

```
   square(unsigned x, unsigned y, int id, float m, float raw_value,
```

```
        unsigned state, int extent_onoff) ; () {
   x_center = x;  y_center = y;
   identifier=id;                    // Cadshape identifier
   cad_type=0,                       // DIP
   mul=m;                            // mul factor transfer
   max+mul=30,min_mul=0;             // Expansion limits
   conv_m=1,                         // Conversion multiplier
   conv_c=0;                         // Conversion constant
   value=(raw_value*conv_m)+conv_c,  // y=mx+c, Eng units from raw
                                     // transducer data
   status-state;                     //Digital status

   status_extn=extent_onoff,         // Extents testing on/off
   height=12, width=12,              // Object minimum size
   action_default=20;                // Re-run increment timer
   action=action_default,
```

Member functions can be called within the constructor

```
   pextent_set(status_extn),
   max_extent_box[0]=x_center-(a*max_mul)-width/2,
   max_extent_box[1]=y_center-(a*max_mul)-height/2,
   max_extent_box[2]=x_center+(a*max_mul)+width/2,
   max_extent_box[3]=y_center+(a*max_mul)+height/2,}
```

Declaration only, definition within cpp file

```
   void perase(),
```

Destructors are not inherited. If a base class has a destructor and no destructor is declared for the derived class then a default destructor is generated. This generated destructor calls the base destructor(s) In this case the *square* class destructor calls the member function perase().

```
   ~square() { perase(),}
   #endif SQUARE_HPP
```

2.4 Memory Allocation.

An named object may be **static** or **automatic** A static object is allocated at the start of the program Its lifetime extends throughout the execution of the program and it is deallocated only when the program ends. An automatic object is allocated each time the block in which it is declared is entered Its scope is limited to the block and it is deallocated when the block is exited. In many cases it is useful to allocate a block of memory at run time, particularly if the size of the block is not known at compile time Such allocation is known as dynamic allocation The *new* operator can be used to allocate memory on the free store (or heap) and the *delete* operator frees the memory allocated by *new* Memory allocated by the *new* operator is not limited to the scope of the block in which it is created An object created by *new* exists until it is explicitly deallocated by the *delete* operator, at which time it is free to be reused again by the *new* operator *New* returns a pointer to the object it has allocated When the object is an array a pointer to its first element is returned *Delete* takes this pointer as an argument The free store operators are implemented by the functions

void operator new (long),*
void operator delete (void),*

The *delete* operator may be called as

delete (void),*

or

delete [size] (void),*

where size relates to the size of the memory block allocated

A program segment from the project provides an example.

```
// file. segment from BITMAPGN4 CPP
   #include <fg h>  // Zortech flash  graphic  library  header file
   main (argc,argv){
```

```
    int argc,
    char **argv,
    unsigned byte_length;     // length of dynamic pixel buffer
    fg_box_t part_screen;     // viewport
    byte_length=sizeof(fg_color_t) * fg_box_area(part_screen);
    fg_color_t far *pixel_in=new fg_color_t[byte_length];
    /*.       */
    delete [byte_length] pixel_in;
}
```

Deleting a zero pointer has no effect However the effect of applying the *delete* operator to a pointer not returned by the *new* operator is undefined and may be harmful

The *new* operator may be used to create an instance of a class For example

```
// square is a class derived from the cadshape class
cadshape* cs_1 = new square(x1,y1),
cadshape* cs_2 = new square(x2,y2),
/* .. . ..*/
delete cs_1;
delete cs_1,
```

Not deleting the object *cs_2* is not an error, but is a waste of space Deleting *cs_1* twice is dangerous, the effect is not specified by the language but by the particular implementation

A dynamic buffer may be allocated within a class definition but should be deallocated within the destructor. Hence an object can be created whose size is not known at compile time It is also possible to use the C memory allocation and deallocation functions such as *malloc(), calloc, free(), realloc(), farmalloc()* and *farfree()* The declarations for these functions are kept in most implementations in the STDLIB H and DOS H header files An example of allocation in a class constructor and deallocation in a

destructor is given by the segment of the *valve* class shown below This class is derived from the *square* class.

```
// file  segment from VALVE HPP
// Second level inheritance

   #ifndef VALVE_HPP
   #define VALVE_HPP
   #include "square.hpp"
   class valve :public square {
       // data  private  to valve
       fg_box_t max_extent_box,            // maximum object area
       unsigned long pixel_buffer_length, // length for buffer
       fg_color_t * pixel_buffer,          // buffer pointer
    public:
       // Methods implemented in  cpp  file
       void pdraw(),
       void perase(),
       // Constructor
       valve(unsigned x, unsigned y) ; (x,y) {
            x_center=x;  y_center=y,
            height=10,
            width=10;

            max_extent_box[0]=x_center-width,
            max_extent_box[1]=y_center-height,
            max_extent_box[2]=x_center+width,
            max_extent_box[3]=y_center-height,
            // Determine the size for the dynamic using the sizeof
            // operator
            pixel_buffer_length=(sizeof(fg_color_t)*
            fg_box_area(max_extent_box));
            // Allocate memory  on  the  heap

            // Pixel_buffer is a
            // pointer to this dynamic buffer
            pixel_buffer=malloc (pixel_buffer_length),
```

```
    }
    // Destructor
    ~valve() { free  (pixel_buffer),        // Deallocate
    }},
#endif VALVE_HPP
```

"The allocator used by the *new* operator stores the size of an object with the object in order for the *delete* operator to function correctly" [STRO87] Hence

cadshape = new square(x,y);*

also stores the size of the *square* cadshape object

If every instance of a class is allocated using the *new* operator then that class does not need a destructor since the *delete* operator can free the space used by the object In this project it was found that when *delete* was used to delete a user defined class object which was allocated with *new* then the object destructor was invoked This conforms to the language version 2 specification [STRO90a] However this can provide some problems for derived classes and cascading destructor invocation This shall be discussed in the relevant implementation sections The problem was overcome by using the *free* operator instead of the *delete* operator The *free* operator does not invoke the destructor but carries out the required clean up of the object and heap

2.5 Criticisms of C++

One of the criticisms directed towards C++ is that "many people switch to C++ because they want the benefits claimed for object-oriented programming unfortunately neither C++ nor any other object-oriented language can deliver these without support from class libraries, development environments and design methods" [DANI90] There certainly is some truth in this statement and to-date class libraries for C++ have been limited and difficult to use In fact, as seen in this project, up to very recently class libraries for graphical user interface (GUI) generation were in very short supply It is still early days in the software integrated circuit industry However there is still the concept of reuse of privately produced objects In this case, great care has to be taken in the development of classes with insight to the future possible use of classes Reusability is however only one of the features of object-oriented languages A very valuable feature is the ability to write code without making irrevocable decisions about data structures The emergence of object-oriented design methods would help immensely in the implementation of object-oriented projects Identification of objects and their methods is extremely important and comes through experience with object-oriented design

Daniels also criticises the object interface/implementation dependency, in that if an objects private section is added to, then a recompilation of all objects or programs using this object is required [DANI90] This may take up appreciable amounts of time Certainly in C++, as was found in this project, if a classes private section in a header file is modified then all child objects and programs including the changed object must be recompiled and the application relinked.

Other criticisms and problems relating to the C++ language are discussed in appendix 1 section 5

2.6 Zortech's Offering.

Zortech C++ [ZORT90b] is a pure native code compiler as opposed to other implementations which are hybrid designs Hybrid implementations use a C++ preprocessor or translator to convert the C++ code to C which is then compiled by some standard C compiler. An example of such a system is Glockenspiel's C++ compiler [GLOC89] This C++ hybrid compiler uses the AT&T translator to translate the C++ code to C, along with the Microsoft C compiler to compile the intermediate C code Both pure and hybrid C++ languages look to the AT&T translator specifications for the current C++ language standard The American National Standard Institute (ANSI) set up a committee, named X3J16, to standardise C++ Its first meeting was held in December 1989 and the AT&T C++ reference manual [STRO90a] has been chosen as the starting point for the formal standardisation of C++ Jones examines the standardization attempts being carried out by the ANSI C committee, named X3J11, the C++ committee and compatibility issues between the two languages [JONE91] In the course of this project versions of both the Zortech and Glockenspiel compilers have been used The compatibility issues in using these products with other graphics and windowing packages and the integrated environments, tools and class libraries that accompany these products will be discussed in the implementation sections of this thesis After in-depth research into the various C++ compilers and graphics tools available, Zortech C++ version 2 1 was chosen as the language to implement the project The reason for this choice is discussed in the implementation sections

Zortech C++ is a two pass compilation system, with an optional third pass providing a global optimization facility to provide tight and efficient object code Zortech C++ supports the language features of version 2 0 of the C++ language as proposed by AT&T with the cfront translator version 2 0, including multiple inheritance and type safe linkage Zortech C++ uses the following file extensions

c	C source files
h	C header files
cpp	C++ source files
hpp	C++ header files
asm	Assembly source files
obj	Object files
lib	Library files

Zortech C++ also allows the use of cxx for C++ source files as used by the Glockenspiel compiler Also if required the h extension can be used for C++ header files although the hpp extension is preferred Zortech C++ produces standard object files which can be linked using normal linkers

In AT&T C++ 2 0 a call to *_entry* is inserted at the beginning of *main()* in order to call all the module constructors In Zortech C++ this call is handled by the C OBJ startup module AT&T C++ 2 0 has a proprietary revised version of the streams I/O library Zortech C++ has an enhanced version of the original streams library Hence Zortech users requiring portability should use the version 1 x format for stream I/O

There are a number of additional features in Zortech C++ which are not part of the C++ language These are discussed at length in the Zortech C++ version 2 0 compiler reference manual [ZORC89]

The Zortech C++ version 2 1 compiler supports Rational Systems Dos Extender DOS/16M [RDOS90], as does the Glockenspiel C++ version 2 0c compiler [GLOC90], and a Virtual Code management overlay system This allows very large programs to be compiled and shall be discussed in the implementation section

The debugging tools supplied with Zortech C++ version 2.1 allow debugging of programs of greater than 640Kbytes by virtue of the Virtual Code Manager (VCM) and Rational DOS extender system. A window of allocated buffers and classes is provided within the

debugger and was found to be of significant value during the debugging of programs in this project

The Phar Lap DOS Extender system [PHAR89] is used with the DOS 386 Zortech compiler version 2 18 [ZTC90a] The use of these software tools should open up the full 32 bit flat addressing space of 80386 and i486 microprocessors The details concerning this technology and its use are discussed in the implementation section

3. The Research and Development Path.

3.1 Microsoft C, GKS and Media Cybernetics.

Initially the Graphics Kernel System was considered for this project, but due to its primitive nature in comparison to other graphics packages available it was not used.

Media Cybernetics produce the Halo graphics package [HALO87] and Dr Halo III [DRHA87], a paint package The Halo graphics package is a Microsoft C [MICR87] compatible library of graphic primitive functions with C binding, suitable for a project such as this Dr Halo III allows the generation of image files, given the PIC file designation, by use of a pointing device and keyboard It is ideally suitable for the generation of schematic bitmapped background files These background files could be diagrams of plant or parts of plant, and can be loaded to the screen from disc, and vice-versa by invoking various Halo Graphics functions The Halo Graphics and Dr Halo III packages use the same image file format and form a powerful graphics facility

Halo Graphics supports a wide variety of graphic functions, device drivers, printer drivers, mouse and locator drivers It also has a learn mode invoked via LEARNMH EXE, from which the various functions can be called interactively A batch file of graphic functions with PIX designation, constructed with any text editor, can also be called from within the learn mode Hence animation techniques using screen page swapping (for multiple page graphic modes), rubberband functions and partial screen save and restore were examined

A C program was written to exercise the various functions available via Halo Graphics, such as disc image file read and write, line, point and shape draw, and the animation techniques described above. The Microsoft C compiler was used to compile the program and Codeview to debug it [CODE87],[CODE89]

3.2 Zortech C++ and Flash Graphics.

In order that object-oriented programming techniques could be used to their full potential it was decided to use a C++ compiler Zortech C++ version 1 07 [ZORT88] provides a set of graphic routines and routines to support the Microsoft mouse pointing device The Zortech example programs were used to examine the Flash Graphics and C++ combination From this a program was put together which allowed the user to draw, move and erase some basic geometrical objects The set of objects were derived from a basic class called the *cadshape* class

The Flash Graphics and Halo Graphics functions were compared to find that even though the Flash Graphics functions are extremely fast, Halo provides a much richer set of functions The test programs which were written and alluded to above were rewritten replacing all Flash Graphic functions with equivalent Halo Graphic functions However this did not operate correctly as the Zortech implementation was not compatible with Microsoft compatible products [ZORC88] To adopt Zortech's technology would have meant leaving the Microsoft compatible environment and using Flash Graphics instead of Halo Graphics thereby loosing direct image file support and many other graphic functions

3.3 Glockenspiel's Advantage.

It was decided to maintain compatibility to mainstream Microsoft products and therefore all the programs developed under the Zortech C++ version 1 07 compiler were transformed from the Zortech environment to the Glockenspiel advantage C++ version 1 2 environment [GLOC88] Hence all CPP and HPP designated files became CXX and HXX respectively The Glockenspiel C++ compiler calls three phases in its execution First a Glockenspiel preprocessor [GCPP88] is called, then the AT&T translator [CFXX88] converts the C++ code to C code which is compiled by the Microsoft compiler This means that the Glockenspiel environment is in fact compatible with other third party libraries which are compatible with Microsoft C, and that the Codeview Debugger can be used The compatibility issue of Halo graphics and the C++ environment was thus resolved

The Halo graphics function allowing image file read from file to screen was included in the programs allowing background files, which were edited using the Dr Halo III Paint package, to be displayed The dynamic class shapes could be drawn over these backgrounds These background files, designated PIC, are in Media Cybernetics format Backgrounds could also be derived using the Media Cybernetics Grab facility [GRAB87] This facility is a memory resident screen grabber which on being invoked transfers the contents of the screen to a Media Cybernetics image file The file HK301 PIC, later to become HK301 ZPI, was derived from a proprietary SCADA system by this method

The *cadshape* set of classes was expanded However the inclusion of many C++ classes within the one program was seen to cause heap or free store exhaustion errors on compilation of the program Also if a program contained multiple nested conditional statements then the compiler gave 'parser stack overflow' or 'out of environment space' errors and terminated the compilation prematurely These error messages were the first indication of problems associated with memory space in the project

3.4 PForce++ Class Libraries

The PForce++ libraries version 1 04 [PFOR87] package provides a set of general purpose C++ character based classes for screen windowing, pop-up menus, options and choice lists It also provides classes for database facilities

It was decided to examine this package with a view to incorporating windows into the programs alongside the Halo graphics facilities There were some initial problems with the software supplied in archive files in that some of the functions which were supposed to be included in the PFPAM4L LIB were not, and some of the header files with the declarations for functions BTR HXX were corrupted However after these problems were overcome sample programs were written which included Halo graphics calls to examine the compatibility of the products, all under Glockenspiel C++ It was found that the only graphics mode where both of the graphics packages were compatible were Halo EGA mode 4 and Pforce Mode 3 or 4 and that even at this both packages need to be individually initialised The initialisation functions of PForce++ *crtsetmode()* and Halo *initgraphics()* or *startgraphics()* were found to be mutually exclusive Regardless of which initialisation was called first neither supported the others graphic mode It was envisaged that a plant mimic would consist of a dynamic foreground sitting on top of a passive background image or schematic of plant Hence one of the key problems is that the background image must be on the screen most of the time This could be achieved only by using PForce for menus and option lists, where the background image is not necessary When the PForce functions were needed the Halo environment could be exited and the Pforce environment initialised Therefore it would be possible by use of the Halo ram image save and restore functions *imsave()* and *imrest()* and continual use of the respective calls to the initialisation packages to run both packages together However the overheads of such a scheme would make it a non-viable option On the basis of compatibility the PForce++ option was not implemented.

3.5 MetaWindows/Plus and Halo Graphics.

MetaWindows/Plus version 3 4 [META89] provides a general library of functions for graphics primitives, windows and viewports Sample programs were written in C++ to test the viability of the co-existence within the one program of Halo graphics and MetaWindows/Plus graphics complied under the Glockenspiel compiler MetaWindows/Plus is a Microsoft C compatible library Both MetaWindows/Plus and Halo graphics have their own strengths and weaknesses with respect to each other which will be alluded to shortly There is no initialisation problem for the packages not supporting each other as in PForce++ as seen above The test programs called up a Media Cybernetics background file from disc using the Halo graphics function *gread()* and then performed various Meta viewport, windowing and zoom facilities together with Halo functions It was seen that Halo screen functions did not observe Meta viewport boundaries but apart from that the packages could be used together Therefore it would be possible using this strategy to use pop-up menus without effecting the underlying background file and without the overhead of disc or ram image save functions as was required in the PForce++ implementation above

The sample programs above were first written in C and compiled with the Microsoft Compiler The programs were then modified for C++ and compiled under the Glockenspiel C++ compiler However the AT&T translator reported errors on the MetaWindows/Plus header files GRCONST H and GREXTERN H and the program WINDSRX C In WINDSRX C the declaration pointer==*NULL* caused an error and was replaced by pointer==0 to form WINDSRX CXX

The essential differences noted between MetaWindows/Plus and Halo Graphics are outlined below

Halo graphics support for image files generated by the sample programs, Dr Halo III or via the Media Cybernetics Grab facility is far superior to that supplied by MetaWindows/Plus

Halo graphics supports acquisition of images from a video camera, a facility not supported by MetaWindows/Plus, which could be of significant use for generating realistic plant backgrounds for mimics

Halo graphics has extensive **rubberbanding** facilities which are not parallelled by MetaWindows facilities From the sample programs it would seem easier to move objects around the screen using Halo functions

The MetaWindows/Plus viewports and windows facility provides for protected areas of activity on the screen This facility is not available in Halo graphics

Changing cursor styles is facilitated more easily by MetaWindows/Plus than Halo graphics

MetaWindows/Plus supports a **Zoom** function, allowing parts of a viewport to be expanded This function does not exist in the Halo graphics library

The sample programs written showed that both of these packages could be used together in a trivial application under Glockenspiel C++, however using two sets of linked-in libraries raised the issue of link and run-time memory requirements for any sizeable application

The Halo'88 graphics package [HALO88] does not add any of the features that MetaWindows/Plus boasts above the previous Halo implementation However it does provide for VGA graphics, and the programs for this project were modified to use this mode The reader should note that the MetaWindows documentation incorrectly identifies the graphics modes for VGA, but that they are correctly specified in the header file GRCONST HXX

3 6 Windows, Operating Systems and CommonView.

Microsoft Windows provides a method of running several programs under DOS Version 1 03 [WIND86] does not support multitasking When one application is called, an already running application is suspended till the new application is terminated, suspended, or runs to conclusion Version 2 03 [WIND87] WIN386, claims multi-tasking operation on a PS/2 or 80386 microprocessor based machine In order that a program utilises the windowing features of Windows the program must be written with the Microsoft Windows Software Development Kit (SDK) [WSDK87] Applications programs, which are not developed with the SDK can be run in *exclusive mode* only This means that the program takes complete control of the screen regardless of whether the application is text or graphics based A PIF file must be set up for the application which gives details of whether the application is text or graphics based and the amount of memory required for the application In **exclusive mode** the program appears as if it is running directly under DOS When the program is suspended or runs to conclusion the Windows shell, or the program that was running when the non-windows window was called, is returned For details on how to write programs specifically for Windows using the SDK the reader is referred to Programming Windows by Petzold [PETZ88]

The reasons why it was decided not to use Windows in this project are outlined below

The Windows environment takes up 300Kbytes of standard system memory leaving a mere 340Kbytes for the application program, device drivers and heap or free store for dynamic objects

The multitasking facility would necessitate a 80386 based machine This would represent quite a leap in hardware requirements for the eventual end user.

For applications to incorporate themselves neatly into the Windows environment and use its full potential they must be written using

the SDK It is questionable whether this would support Halo graphics or MetaWindows/Plus

The Glockenspiel CommonView Applications Framework [GLOC89] provides classes of objects for Microsoft Windows and OS/2 Presentation Manager (PM) [PMAN89] These classes can be utilised by a C++ application program to produce a Windows or PM application Due to the compatibility issue of graphics packages such as Halo graphics and the lack of support for the large memory model, it was decided not to use this application framework However CommonView was accompanied by the version 1 2E Glockenspiel C++ compiler which was adopted This implementation directly supports the Codeview debugger

There are a whole plethora of operating systems on which this application could be run, but due to cost and inavailability of OOPS tools for these platforms it was decided to attempt to run the application under DOS using terminate and stay routines (TSR) for any future background processes The mechanics of terminate and stay routines are discussed in the Waite Group's MSDOS Developer's Guide [MSDG89]

3.7 Developments under Glockenspiel C++

It was decided to use the programs written to date, to test the various packages and compilers above, consisting of a small set of graphic object classes and a CAD program as the base for the project The CAD program could call background image files to screen and was capable of drawing, moving and erasing the dynamic objects

The system design was to consist of the following parts

i The construction of a set of primitive objects, some that could effect animation and others that would be passive

ii A menu generator which could interactively use the primitive objects to generate a sub-menu of primitive parts

iii A mimic generator which could be used to interactively design icons from the primitive menu and generate an icon menu The icon menu could then be used to generate mimics based on these icons

To effect this the set of primitive classes was expanded and the programs MENUGEN and MIMICGEN were written in C++ with Halo graphics and compiled under Glockenspiel C++ compiler Version 1 2E and Microsoft C version 5 1 [MICR88] The programs used the Microsoft mouse and VGA graphics

The Halo graphics documentation implies that when using the move functions *MOVETO, MOVEFROM, MEMEXP, MEMCOM* that segment 0 should be used in segment, offset format rather than in buffer format This is in fact incorrect When memory is allocated using the *new* operator then *FP_SEG()* and *FP_OFF()* should be applied to the far pointer returned by *new*

In was seen that fragmentation of free memory or heap due to multiple *new* and *delete* operator intercalls caused significant heap

problems These problems were resolved by strictly monitoring the heap usage

The set of objects consisted of passive and active objects Active objects could be animated with an analog value or a digital switch causing their character to change The programs were capable of carrying out the design features required, and a *modulate* facility in MENUGEN when called caused animation of the active objects so that their animation features could be tested However the C++ environment did not support disc storage for classes and hence the object foreground was volatile A program was also designed to allow Media Cybernetics image files to be cut up to form bitmap files which could be used as icons

Each object was given a set of extents and methods to generate and test for overlap of extents The extents in each case represented a boundary around the object, and set this area as a protected area This meant that objects could not overlap on the screen The move facilities in all programs used the extents test to ensure a clear area before moving the object in question into that area

In order to compile the MENUGEN program discussed above without the compiler error *'INTERNAL - ERROR FREE SPACE EXHAUSTED'*, it was necessary to generate a batch compilation file by running the compiler with the *'O* switch and setting output to the batch file It was also necessary to remove all drivers from memory during the compilation stage leaving 550Kbytes available for the compilation Despite the fact that the program set was restructured memory problems remained

Memory inadequacies were also detected at run-time The elements requiring large amounts of memory in the executable program were the Glockenspiel libraries, the Halo graphics libraries and the Zortech library (ZLL LIB) used for the Microsoft mouse routines

3 8 Developments under Zortech C++

The memory problems discussed in the Glockenspiel implementation above prompted the examination of the Zortech C++ version 2 0 environment with integral Flash Graphics and Microsoft mouse support [ZORT89]

Zortech C++ version 2 0 requires that all functions declare their prototypes before use Halo Graphics does not supply any header files of function declarations, hence compilation errors occur Thus Halo Graphics is not compatible with the Zortech environment

All the programs developed under the Glockenspiel environment including the CAD objects (icons) were rewritten to use Flash Graphics functions and Zortech C++ New icon classes were written and many of the programs were upgraded at this point The details are set out in the implementation chapter of this thesis The compiler flag for large programs (*-b*) had to be set to compile some of the bigger modules

In order to link the animation of objects to analog and digital I/O events and real-time, the objects must satisfy a number of criteria before animation is allowed These criteria, which were designed into the application at this stage of the development, include a simple counter decremented on each call to animate an object, a raw value deadband and an intersample time deadband The details will be discussed in the implementation section

There were considerable differences between the Flash Graphics and Halo Graphics libraries Flash Graphics has less functions, than Halo Graphics, the flash functions available requiring more parameter passing An example was the fact that Flash Graphics provided no function for drawing a pie shape It was necessary to construct a pie from an arc and separate lines The *fg_drawarc()* function required the angles in tenths of degrees whereas the *sin* and *cos* functions required angles in radians. No radian to degree

transform function was provided While Flash Graphics provides a limited set of graphic primitives, it is the responsibility of the user to generate from these a larger set of functions Flash Graphics defines its own data types like *fg_box_t* for a box type It was also seen that whereas Halo Graphics uses a left-hand graphic system, Flash Graphics uses a right-hand graphic system In a left-hand system the lowest value coordinates are at the upper left corner of the screen and the highest value coordinates are at the lower right corner of the screen In a right-hand system the lowest coordinates are at the lower left corner of the screen and the higher coordinates are at the upper right corner of the screen

The Flash Graphics library provides the function *fg_pt_inbox()* which was very useful in the rewrite of the extents testing methods Zortech Graphics provides a very small set of text handling functions and no font functions

The *dos* and *bios* calls are named slightly differently in Zortech than in the Glockenspiel C++ environment

It was also found that there is no equivalent function in the Zortech C++ libraries of the Microsoft *stdlib* function *gcvt* which converts floats into char strings including decimal point and sign for subsequent output to screen The *ecvt* function with explicit decimal point and sign location was used instead where required

It is stated in the Zortech documentation that MetaWindows is compatible with Zortech C++ The option of including MetaWindows libraries to complement the Flash Graphics functions was not taken up because firstly it was found that when MetaWindows functions and header files were included in test programs that compilation errors were reported, indicating some incompatibility, and secondly such an implementation might suffer the same memory space problems experienced in the earlier Glockenspiel implementation.

The background image files acquired as detailed above were now in a format (PIC) that Zortech had no way of dealing with In fact the

Zortech Graphics facility provided no direct method of writing or reading the contents of the screen to or from a screen image file Also there now existed no function to read to screen the bitmap files produced by cutting a piece out of a PIC image file These bitmap files were used in bitmap icons to enable parts of a picture to be placed over the background image These problems were overcome by using partial screen reads and writes via *fg_readbox()* and *fg_writebox()* functions The use of AT&T scanner files was also researched and an image code converter was written

The Zortech debugger ZDB EXE allowed easy debugging at C++ source level Noticeable features with respect to Codeview are the display of classes and their respective data and methods, and display of all dynamic buffers and their addresses The latter feature was particularly helpful in monitoring heap usage and the allocation and deallocation of class objects

Zortech also provides a library of classes Of interest in this library were the window, ladder, directory and button classes However after testing these classes it was found that unfortunately they are for text mode only It was necessary to call *fg_term()*, thereby clearing the screen and terminating the graphics mode, before using them, and hence they were not used

3.9 OS/2 AND ZORTECH.

During the development of software under Zortech C++ considerable memory space difficulties were encountered These problems prompted the use of separately compiled modules which were subsequently linked into the EXE file However as new classes of objects were introduced and new features brought into the MENUGEN and MIMICGEN programs eventually these programs became too big to compile It was necessary to remove classes from the implementation and carry out development on an incremental basis

The idea of generating icons interactively was, taking into account the size and complexity of the project at the time, seen as not being practically realizable within the environment chosen Therefore the programs were rewritten so that the primitive classes became the icons, the MENUGEN was upgraded to become the DESIGN program and a new program ANIMATE was written for simulating animation Separately compiled subroutines modules were used wherever possible This eased the memory problems, but when it was required to use the debugger ZDB EXE then the application had to be cut down considerably to provide the required memory at run-time

Apart from the compilation time and run-time debug memory constrictions difficulties arose from insufficient heap space When more than a trivial number of objects were constructed the heap space ran out This was partly due to the fact that all the CAD objects, to facilitate moving and resizing, stored the part of screen background which they were covering in a heap buffer

Compilation and run-time memory requirements made it increasingly difficult to add new feature, such as an object for a status box indicating when files were being read, into the DESIGN and ANIMATE programs The application at this point was small and other features such as database and raw data communications to the active CAD objects would eventually need to be incorporated DOS and its constricting 640Kbyte barrier would not facilitate the project any further OS/2 was considered as an alternative

Zortech provide an upgrade pack [ZORT89b] for OS/2 [OS288] However it is an upgrade from the DOS environment rather than a new package specifically for OS/2 It requires some header and library files and the linker from the Microsoft Presentation Manager Toolkit [MSPM89] Some modifications are also required to the file OS2 H There is no debugger available from Zortech for the OS/2 environment It was noted that Flash Graphics and Presentation manager were totally incompatible

The programs were initially edited to remove calls to BIOS, DOS and Flash Graphics commands not supported in OS/2 These included *bios_timeofday()* and *fg_init_vga13()* However IOPL errors, *system trap $0D, system error* 1811, were reported on Flash Graphic initialisation calls, in spite of the fact that the IOPL was validated for applications running under OS/2 The same error was reported on exit from the ZCONFIG EXE screen configuration file After interfacing with Zortech it was decided to examine the Zortech version 2 1 Virtual Code Management system [ZORT90b] which, it was claimed, would overcome the memory problems experienced by developers with non-trivial applications

At the same time as the OS/2 option was being examined, new dynamic icons were written and the existing icons and programs were upgraded

3 10 Zortech's Revenge and Virtual Code Management

The Zortech VCM system, as shall be expounded on later in this section, overcame the memory problems intrinsic to this application However there were other reasons for choosing DOS as a platform over OS/2 which include

i Some of the VGA graphic modes are not supported by OS/2

ii There was no literature available on Flash Graphic applications under OS/2

iii DOS was a more popular platform and OS/2 represents an extra level of expense and complexity which may not be required

iv OS/2 equivalents for image and bitmap file compression and other executables would have had to be sourced or written

Zortech version 2 1 [ZORT90b] provides support for two methods of generating large applications, namely Virtual Code Management and Rational DOS Extender Technology [RDOS90] Virtual Code Management was chosen, for reasons to be illustrated in the relevant implementation section

This version provides very powerful debugging facilities that did not require large amounts of memory below the 640Kbyte mark Therefore application programs could be debugged without cutting them down to a minimum implementation, which meant in many cases that dynamic memory problems which were very difficult to trap could now be trapped with a certain amount of ease Program development could have been significantly easier if these tools had been available from the project start It should be noted that the latest version of the Glockenspiel compiler [GLOC90] is supported by CodeView version 3 0 [CODE90] and thus claims to provide C++ source level debugging

It was possible via the VCM system to bring together the DESIGN and ANIMATE programs in their entirety The DESIGN program allowed the selection and reading of a background image file from a selected compressed library of images and the subsequent selection, drawing, sizing, data editing or erasing of icons from the full available set to form a foreground The ANIMATE program simulated raw data coming in from transducers and caused animation of the dynamic icons These two programs were then amalgamated yielding a single application for the design and subsequent animation of a mimic diagram

3 11 Phar Lap 80386 Linker.

The Phar Lap linker ,386LINK EXE, [LINK89] is a linker for the Intel 80386 microprocessor Protected mode programs linked by 386LINK can execute under Phar Lap's 386 DOS Extender RUN386 EXE [RUN389] and real mode applications can be executed under MSDOS DOS 386 C++ [ZORT90a], is a Zortech development environment which uses the facilities of the 80386 microprocessor together with the Phar Lap 80386 linker, version 2 2d, to produce an executable file which may be run in protected, real or mixed mode The Zortech compiler [ZTC90a] provides the ability to generate 32 bit Phar Lap compatible object files The Phar Lap linker 386LINK EXE or FASTLINK EXE [FAST89] produces a EXE DOS or a EXP DOS Extender executable file Real mode 16-bit object files must follow the standard Intel/Microsoft OMF-86 file format definition while 32-bit protected mode object files must follow Phar Lap's OMF-386 format [OMF88] The default for the DOS 386 Zortech compiler is a flat 32 bit address space There is no use for overlays in such a model and hence VCM or other overlay systems are not supported The Phar Lap linker provides support for full symbolic debugging by appending CodeView information to the executable Zortech's 80386 debugger ZDBPH EXE recodes this information for its own use The protected mode linked program can be run via RUN386 EXE Alternatively, if a stand alone executable is required then BIND386 EXE can be used to produce a file which can be executed from the DOS command line A protected mode program has access to the virtually unlimited linear addressing range of Intel 80386 and i486 microprocessors

The tools which make up the Zortech development environment are now provided as 32-bit protected mode executables and hence can use the full 4 Gbyte address space

The programs forming the DESIGN application executable file were recompiled for protected mode and updated as necessitated by modifications carried out to the C++ compiler The resulting object files were linked with FASTLINK EXE and run via RUN386.EXE This

application is a mixed protected and real mode program The necessary linker switches were applied to reserve space for real mode elements below the 640Kbyte barrier and REALMODE OBJ linked into the program as advised by Zortech It was found that real/protected mode transfer was not carried out successfully for file access and object/memory allocation To compound the problem successful debugging was not possible because within the debug environment any Zortech file access or Flash Graphic text screen access functions caused an error thus halting any further progress

Even though the unattractive features discussed above necessitated the continued use of the VCM system, as expounded upon in section 3 10, they only represent product immaturity The general move to 386 and i486 software technology will in the very near future represent a significant step forward for large object oriented DOS applications heretofore cramped by the 640Kbyte barrier imposed by 8086 microprocessor architecture

4 Graphics

4 1 Image and Schematic Background Files.

Marshall [MARS87] discusses the vast range of sources for graphics diagrams which may be employed In this project schematic diagrams are obtained by using graphics editors or by using various screen grab facilities In addition plant pictures or schematics may be obtained by the use of a document scanner [ATTS88]

Schematic diagrams generated directly by the use of the Dr Halo III paint package [DRHA87] and plant mimics captured from specific types of proprietary systems, using the Media Cybernetics GRAB facility [GRAB87], can be converted into a format which is suitable for Flash Graphics functions The program which converts PIC files to ZPI files is called TRANSFORM

AT&T scanner files in PCX file format [SCAN88], which may be edited using PC PAINTBRUSH [PCPB88], may be converted into the Flash Graphics function compatible file structure using the program TRPCXZPI Figure 4 1 is a screen dump acquired from the system while the DESIGN EXE program was executing It shows a picture from a distillation plant scanned using an AT&T scanner and subsequently converted into a Flash Graphics file to be used as a mimic background The Flash Graphics function compatible file may also be converted back to PCX format by use of the PAINTBRUSH FRIEZE facility

The Flash Graphics library does not directly support screen file operations A full VGA screen has 640x480 pixels Hence a screen save requires 307200 bytes Under DOS the maximum space which can be allocated for any one buffer is one segment or 64Kbytes DOS operates on the basis of a 640Kbyte total address space and the 8086 microprocessor architecture, where there are 20 address lines, 16 of which are used for addressing within each segment and the remaining 4 for addressing the required segment Hence a number of buffer operations are required for each full screen read or write. Multiple calls to *fg_readbox* and *fg_writebox* are used to read the

Figure 4.1 Distillery.

screen to and from a heap buffer, which is then appended or read from a disc file with ZPI designation Thus the background image is seen to appear on the screen in strips The background disc file represents a large disc commitment This is overcome by invoking the DOS product PKZIP [PKWA89] to compress the image file and store it in a compressed library of images or a ZIP file Imploding indices of over 90% are possible with typical mimics When a background image is required it is decompressed from a library of such images and put to the screen as shown above The routines responsible for this action are BDISPLAY and SELECTBG

bdisplay (char argv1,char* argv2),*

Bdisplay requires parameters specifying the ZIP image library file and the ZPI file in this library that is to be displayed The library files A2 ZIP through A6 ZIP contain various compressed ZPI picture and schematic files The routine takes care of the ZPI file decompression via a DOS system call to the PKUNZIP utility It also allocates and deallocates dynamic memory for the buffers used and deletes the ZPI file when it is on the screen via a DOS system call to the DEL DOS utility

Error handling for ZIP and ZPI file read operations is catered for by opening a window on the screen with an error message As shall be seen later, this routine is used in the DESIGN program when the background image library and image file are given on the command line on call of the DESIGN program

select(),

Select takes no parameters but presents the user with a series of menus for selecting a library of images and then for choosing a specific image ZPI file The first menu presented on the screen takes the form

Zip Search

Zip Entry

No Zip

This gives the user the option of searching the disc for all image library files, selecting a particular image ZIP file or choosing no background image file

If a ZPI image library file is selected then the next menu is presented *Zip Expand*

Zpi Display

Zip menu

The user may choose to view the contents of a library ZIP file, as shown in figure 4 2, select a particular image ZPI file for subsequent display, or return to the first menu that was presented ZPI and ZIP file read errors are catered for and windows indicating such states are opened on the screen Status reports are also opened on the screen via windows, for example *Searching for ZIP files* is indicated in a window when the *Zip Search* option is called in the first menu Status reports are indicated in green, whilst error reports are indicated in red The entry of text for file names is protected against invalid characters including an audible warning and the normal erase facilities are provided Files do not have to reside on the current directory and menu options are vertically mouse pointer selected If no image file is selected the screen is cleared This routine is used in the DESIGN program if no parameters are specified on the command line in the invocation of DESIGN

```
PKZIP (tm)   FAST!   Create/Update Utility   Version 1.01   07-21-89
Copyright 1989 PKWARE Inc    All Rights Reserved    PKZIP/h for help

Searching ZIP  A6.ZIP

 Length  Method   Size  Ratio   Date    Time   CRC-32  Attr  Name
 ------  ------  -----  -----   ----    ----   ------  ----  ----
 307200  Implode  19938  94%  04-12-90  15:37  41f2e014  --w  FARMGAS1.ZPI
 307200  Implode  22999  93%  04-12-90  15:38  d9b02d56  --w  FARMGAS2.ZPI
 307200  Implode  72228  77%  04-12-90  15:41  b2a0f282  --w  PUMPHOUS.ZPI
 307200  Implode  32721  90%  04-12-90  15:42  59dc376a  --w  PUMPHSE2.ZPI
 307200  Implode  41931  87%  04-12-90  15:43  d90eae5b  --w  PUMPHSE3.ZPI
 ------          ------  ---                                  -------
1536000          189817  88%                                        5
```

Figure 4.2 Expanded Zip file.

4 2 Bitmaps and Object Background Files.

In some applications it may be advantageous to be able to use bitmaps as icons This allows pictures of plant or measuring equipment the flexibility of movement around the screen just like any other icon As shall be seen later, in section 4 3, it may also be possible to allow other icons to overlap or reside on top of bitmaps in order to effect animation

The program BITMAPGN is used to cut a piece of an image ZPI file from a library of image files and generate a bitmap file this allows parts of schematic or scanned files to be used as objects The bitmap file is given the designation ZMP and compressed into a library of bitmap files BITMAPGN is called from DOS with the arguments source image library file ZIP, image ZPI file in this library, destination bitmap image library ZIP and bitmap image ZMP file in this library The program allows an area on an image file to be selected and captured to form a ZMP file The image file is decompressed from its library at the start and deleted at exit The ZMP file is generated and compressed to a library file, then the ZMP file is deleted The program uses dynamic buffer allocation and deallocation DOS system calls for PKWARE compression utilities and allows subsequent viewing of the bitmap that was captured

The bitmap image files of maximum size 64Kbytes have the format

```
{int x1,y1,      // capture area lower left coordinates
int x2,y2;       // capture area upper right coordinates
[1Byte/Pixel]    // 1 byte /pixel
}
```

These bitmap image files are used by the CADSHAPE object BMA1 Figure 4 3 shows a BMA1 bitmap object of a pump obtained as a screen dump from the system during execution of the DESIGN EXE program On construction of an instance of this class, the bitmap image is unzipped and decompressed from the bitmap image library

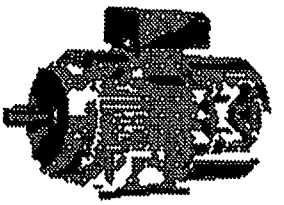


Figure 4.3 Bma1 Bitmap Object.

and placed in a dynamic buffer allocated in the constructor The size of the buffer is calculated from the capture area contained in the first eight bytes of the ZMP file The buffered bitmap pixel information is passed to a file also generated in the constructor, which is used subsequently when the *bitmap pdraw()* function is called The ZMP file and dynamic buffer are then deleted The new bitmap file name is designated BM4**** SAV The **** letters come from a random number generator This file is deleted on destructor call, or if the destructor is never called then it is implicitly deleted at the end of the DESIGN program Any file read errors are flagged to the screen via a window The BMA1 object allows pieces of image files to be overlaid on top of the background image file It can be used as any other CADSHAPE type object except that it cannot be animated in itself However it is possible to overlay active CADSHAPE objects over the BMA1 object to effect animation This is discussed in section 4 5, the animation section of this chapter

Experimental work was carried out to transform 256x256x6 bit video acquired image files to bitmap file ZMP format Programs were written to examine this Experimental programs were also written to expand out the video images with the possibility of forming background images However it was found that the quality of the original images were too poor to be of any use in this project

K Porter [PORT87] describes a method of saving the portion of a screen obscured by pop-up menus by the use of buffers and direct access to display memory In this application background files are used to store areas obscured by icons and graphic routines are used to capture the required areas Object background files designated SAV files are used to save the background behind icons and templates When an icon or template is drawn on the screen then the bitmap of the background behind the icon or template is stored in a dynamic buffer in order to preserve the background If the buffer is kept active for the life of the object then when an erase or delete of the object is required, the object is first erased

leaving a hole or black rectangle in the background, the object background buffer which has the saved bitmap is redrawn on the screen, thus the background is reconstituted When the object is being moved a new bitmap is saved in the background save buffer and then the icon is drawn over this area Keeping the background dynamic buffers open for the life of the icon or template and for all icons or templates represents quite a strain on heap memory, which will in fact eventually run out and thus limit the number of icons that can be active at the one time This problem was overcome by the use of background SAV files When the bitmap is saved to the dynamic background buffer it is transferred to the object background file and the dynamic buffer is deleted This reduces the requirements for heap memory dramatically but limits the speed or redraw for an icon due to disc access characteristics This is overcome to a very large extent by using a Virtual or RAM Disc for the SAV files which eliminated the disc access delays The background SAV file names are generated within the object constructor using two letters from the object name and four characters from the random integer number generator *rand()* For example BM6329 SAV for bitmap background save file

```
file_rand = rand(),
itoa (file_rand,sub_name,10),
```

The file is created in the *pdraw()* method and the portion of the background, that the object will cover is passed into the file via a new dynamic buffer The new dynamic buffer is then deleted, freeing up memory on the heap When the object is erased the background information is taken out of the file and the file is deleted The files scope is from *pdraw()* method to *perase()* method only

The WINDOW and WINDOWTILE classes still use background dynamic buffers for their whole lifetime Their load on the heap in quite low as windows are typically small and there are not generally more than a few open on the screen at the one time

4 3 Extents, Icon Auto-Placement and Overlapping.

An extent defines a rectangular area about a screen object The theory and practice of the use of extents is expounded on in great depth by Foley and Van Dam [FOLE84] All icon objects in the system that can form part of the mimic foreground have an extents area associated with them Other objects such as the TEMPLATE, WINDOW, WINDOWTILE and MSMENU classes are used in the generation of the foreground, but do not form part of that foreground They are drawn on the screen and save the part of the background which they write over, hence it can be said that they are written non-destructively over the background Once used they are erased and the background is restored This is in contrast to foreground icons which form part of the mimic As these icons give, for the most part, important plant information, it is crucial that they are visible and not overwritten by another icon A protective area is defined around each icon, which is known as the icon's extent The extent box enclosing any icon is defined by the extent generation method of the icon A dashed box corresponding to the extents area may be drawn and erased by extent methods of each icon This box is drawn in screen XOR mode to protect the mimic background The screen XOR and AND draw modes are discussed in more detail in section 4 5 These methods all take into account a multiplication factor in order to accommodate expansion or reduction of the icon

When an icon is to be moved, a set of extent box coordinates is generated for the proposed new position The extents box corresponding to the newly proposed position is then compared to the extents boxes of all existing objects on the screen If the new extent box intersects or overlaps with the extents of any other icon then an extent overlap condition occurs and the move is not allowed This process is shown in figures 4 6 and 4 7 There is an exception to this which shall be discussed shortly In the DESIGN program of this project an icon can be moved by moving its extents box and when the final free position is decided upon the icon itself is moved If no free position can be decided upon then no move is carried out The entire icon can also be moved and most

No extent overlap

Figure 4.6

Extent overlap 1

Figure 4.7

icons can also be resized larger or smaller. All three facilities test for extents intersection or overlap and report if extents transgression conditions exist.

When an icon is selected from the menu it is automatically placed on the screen in such a way that no extents transgression occurs. This is achieved by testing for an extents transgression with any other object. The new extents that would result from the object being placed at the required position are first tested. If this fails then the extents are tested at points about concentric squares built up about the required point till the extents test for transgression with any other objects extents, is passed. Thereupon the object is drawn. There is also a facility within the routine for automatically placing a series of objects, in which case a new set of concentric squares is utilised for each icon starting one square before that used for the last automatically placed icon. Because of the variety of icon shapes this can give a looser set of icons than if the placement started at the centre on each occasion. However there is a considerable saving in the time required for placement of a large set of icons.

Bitmap icons, as seen above in section 4.2, provide a method of using parts of captured images or schematics as icons. Their animation depends on using other icons and hence it is required that other icons can overlap them, as shown in figure 4.8. This creates two categories of icons, those that allow overlapping on themselves by other icons, and those that do not. As the animation effect is to be applied to the bitmap icon other icons are allowed to overlap it. However it is not allowed to overlap other icons. Extents testing takes into account these two categories of icons. An icon is switched from one type to the other by simply switching its extents protection on or off in the icon constructor. The order in which moving or deleting overlapping icons is carried out is important in order to ensure that the integrity of the mimic background and foreground is maintained. The DESIGN program protects against illegal icon movement and flags user illegal moves or deletions with an appropriate message. An example of an illegal

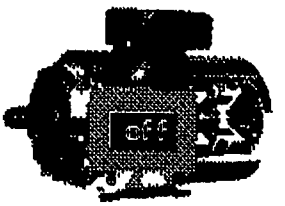


Figure 4.8 Overlapping Icons.

deletion would be trying to delete a bitmap object on top of which resides some other icon, provided for animation purposes If the bitmap icon deletion were allowed and subsequently the animation icon was moved then the background behind the animation icon would not contain the mimic background but in fact the piece of the bitmap icon which it had covered The program should only allow the bitmap to be deleted if all icons residing on top of it are deleted first, in other words operations on overlapping objects must be carried out in such a way as to support overlapping

The extents of an object being moved or expanded are also checked against the boundary of the viewport or screen, which in this case is the Flash Graphics *fg_displaybox* coordinates This is to ensure that icons always remain within the visible screen

4.4 Real-Time Considerations.

A real-time monitoring system requires that the screen information, reflecting plant status, be updated within some time increment so that the information is updated fast enough compared to the time constant of the plant under supervision The process by which the foreground icons impart information about the plant status is known as animation When an icon is animated it simulates the action of some physical device, which may be for example a measuring instrument or a piece of plant or equipment The DESIGN program animation function tests if the set of foreground icons can be animated within the increment given by the integer variable *real_time_barrier* If the animation of all the icons is within this value the system outputs the message *system wait* via a message window and waits till the increment is up before it starts another animation run of all the objects in the foreground If it takes longer than the *real_time_barrier* to complete the animation run, then the system outputs the message *system too slow* via a message window The animation function of the DESIGN program carries out a fixed number of foreground animations The percentage of times for which the animation takes longer than the *real_time_barrier* is calculated and presented on the screen via a message window The *real_time_barrier* is set at 5 ticks or approximately 275mSec If the animation cycle persistently breaks the *real_time_barrier* it may be necessary to run the system on a faster machine or alternatively, if the plant time constant allows, increase the *real_time_barrier* value Figure 4 9 is a screen dump from the system during an animation cycle It shows a mimic diagram of a pump control panel consisting of a text background and icon foreground At the instant of the screen save all icons have been updated and the system is waiting till the animation increment has elapsed before starting another animation run This is indicated by a message in a window at the lower left corner of the screen

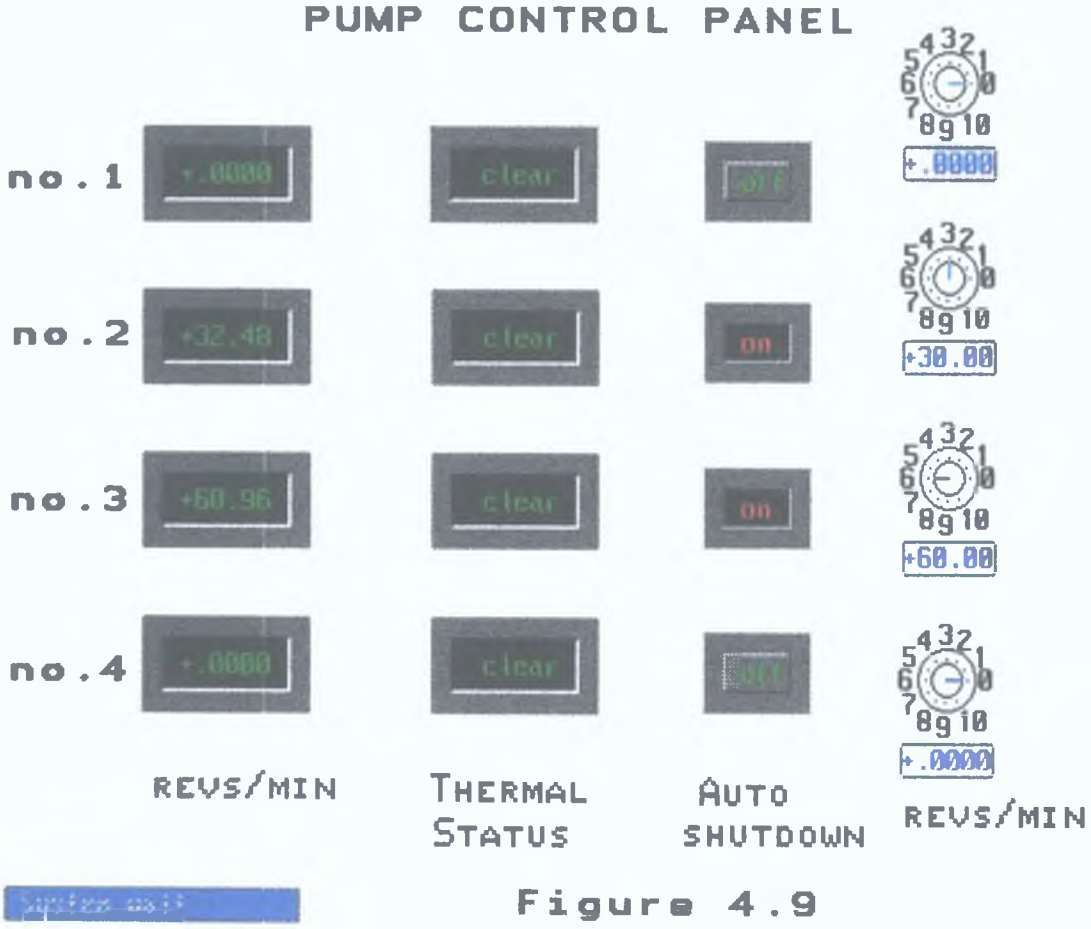

PUMP CONTROL PANEL
no.1
+.0000
clear
off
no.2
+32.48
clear
on
+30.00
no.3
+60.96
clear
on
+60.00
no.4
+.0000
clear
off
+.0000
REVS/MIN
THERMAL STATUS
AUTO SHUTDOWN
REVS/MIN

Figure 4.9

The three criteria, which are used to determine whether a particular foreground object should be animated, are

i an action counter,
ii a change of state or analog value, and
iii a time deadband

The action counter and the action counter default value, *action* and *action_default*, are declared in the CADSHAPE class The *action_default* variable is defined and the *action* variable is initialised to its default value in the constructor for each object When the animation method of an object is invoked this *action* counter is decremented If its value is zero then further animation criteria are tested, otherwise the animation is ended without any further processing till the next call to this method, when the *action* counter is once again decremented If the *action* counter reaches zero, then it is reset The criteria that are applied following the decrementing of the *action* counter to zero depend on the type of icon which is being animated

For objects that respond to a digital or analog signal change by a single non-continuous animation effect, that animation effect is carried out if there is a change in the digital or value status of the icon Examples of such objects are the DIGVALUE, STATUSTX, VALUETX, STATUSTILE and VALUETILE objects which simply output a new value or text string on the screen within their screen framework when a digital or analog value change takes place and is communicated to them by invoking their animation method Other examples include the SQUARE class, which changes colour on a digital change of state being communicated to it via its animation method Another example is the ANIMPIE object This icon mimics a pump where the rate of motion of the rotating vanes is proportional to the analog value

Some objects compare the time since their last full animation, the inter-sample time, to the analog value that they are required to

mimic giving a rate of animation that is dependant on the value In such cases the animation effect may be constant but carried out at a variable rate, or the animation effect and the rate both may be proportional to the analog value that is being simulated In the latter case the effect of analog value changes is exaggerated Examples of such icons are the ANIMPIPE object which simulates a pipe, where the liquid flow rate is proportional to the analog value that is sent to the object via the *panimate()* method Another example is the ROTSHAFT object In this case the rate of rotation is controlled by comparing the inter-sample time to the analog value to be simulated

Finally the ANIMPLOT object carries out a full animation if the time since the last full animation is greater than a debounce or deadband time Changing the deadband time changes the frequency with which full animations are allowed

4 5 Animation and Icons.

Animation is initiated via the DESIGN animate menu function, simulating plant activity as described above in section 4 4, or via user input to a digital or analog input icon

Icons which are used for digital and analog input by the user, via the mouse, carry out animation when invoked by the user When the user inputs a digital change of state by clicking on a BUTTON or BUTTONTILE type object, shown in figure 4 4, then the *digital_pinput()* methods of these animate the icon accordingly to represent the user action of pressing or releasing a button Such digital output icons are designated DOP type icons Similarly the animation for the POT and SLIDER classes, shown in figure 4 4, is carried out while the user moves the knob or slider bar via the mouse and the *analog_pinput()* method The corresponding value which the position of the slider or knob indicates is reflected in a value box in the lower part of the icon These analog input icons are designated AOP type icons

Some icons have no animation effect The bitmap BMA1 class, shown in figure 4 3, is an example of such a type As discussed above in the extents section, 4 3, this object allows other icons, which can be animated, to overlap it Hence animation can be applied to the BMA1 bitmap object indirectly Icons which have no inherent animation are designated as PASSIVE icons

The vast majority of the icons used in a system such as the one developed by this project are icons which are animated based on values or status sent to them from the plant or by simulation Those icons which reflect analog measurements are designated as analog input (AIP) icons, whereas those that reflect digital change of state measurements are digital input (DIP) icons There exist also a wide set of measurement applications where combinations of signal type are used For example a motor may have an icon which not only gives an indication of its speed as an analog quantity but

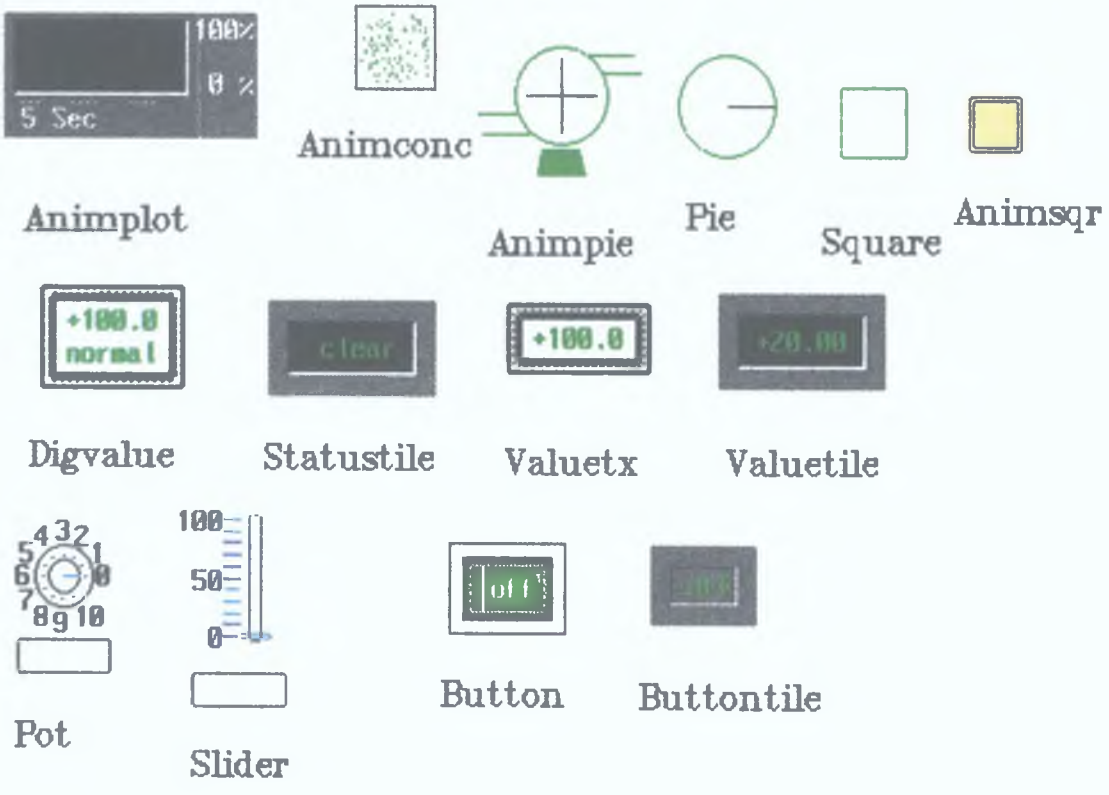


Figure 4.4 Foreground Icons.

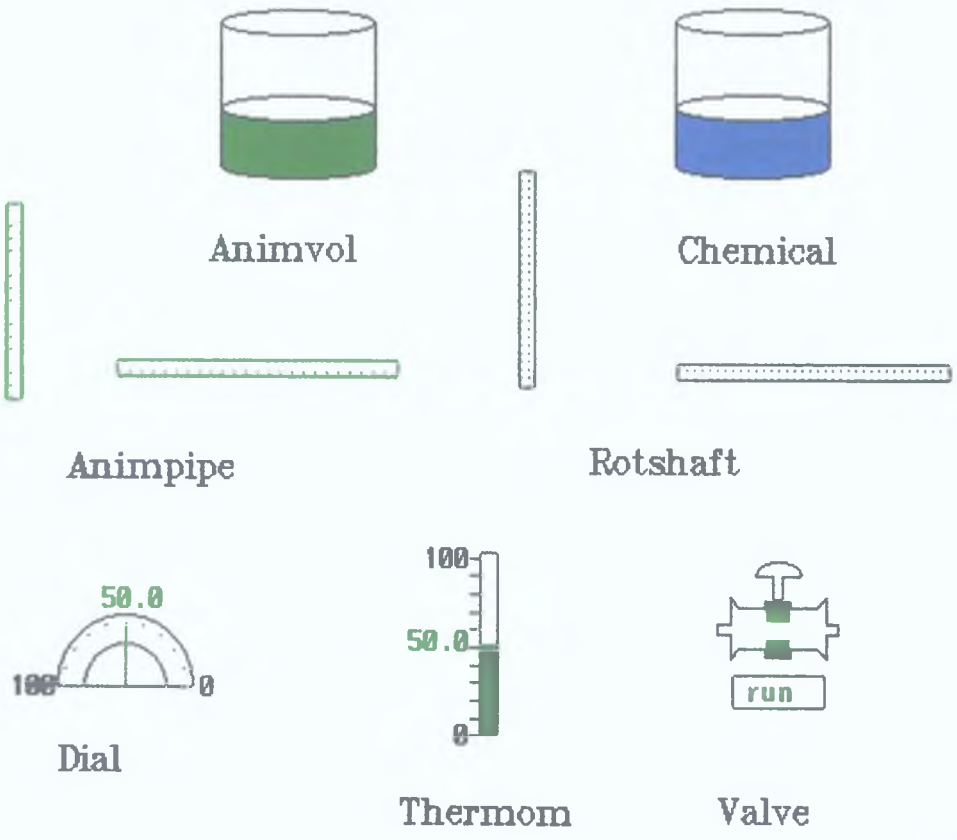


Figure 4.5 Foreground Icons.

also indicates its thermal cut-out state as a digital quantity Clearly this icon has both an analog and digital function, in which case it is an AIPDIP type Another interesting type is the analog multiple digital type (AIPMDIP) This icon accepts both analog and digital input as above, but it also accepts multiple digital levels rather than the conventional binary digital signal An example of this type is the CHEMICAL icon, shown in figure 4 5, which uses the analog value to set the level of a liquid in a tank and uses the multiple digital input to select a colour for the liquid This allows different substances to be distinguished

An icon could be drawn on top of the background picture in either **exclusive-or** (XOR) or in **and** (AND) screen write mode If the background is a fixed colour, then the interaction that occurs in XOR screen write mode between the background and the colours used in the icon, will produce predictable results The routine which draws the icon over the background could pick colours for the icon parts to suit the background in order to give the required icon presentation However if the background is in more than one colour then the resulting colour appearance of a icon drawn over the background in XOR mode will be indeterminate For the icon to be presented in predictable colours, with a multiple gray level background, the draw method would have to test every pixel and output the icon pixel by pixel, selecting the correct colour as it went along This process would represent an unreasonable length of time to draw an icon or animate some part of the icon The advantage of the XOR mode is that an icon or part of an icon can be erased simply by redrawing the specific part no longer required Therefore no background manipulation is required The PIE icon is an XOR screen write icon This icon is see-through as the background is not processed in any way The efficiency of the process of facilitating information transfer to a user is highly dependant on colour The power of using colour for status of plant, different chemicals, alarms and messages is far more important than the gain attributed to using the XOR screen write mode

The AND screen write mode obliterates any background pixels in its path Hence any icon written in this mode will retain its colour integrity However if the icon is moved or animated in any way the 'holes' in the background picture will become apparent This means that if see-through icons are required then, between each animation movement of an icon or during any movement of the icon across the screen, the background pixels that were overwritten must be replaced This would result in a large delay in the animation and movement functions However if a bitmap of the background behind the icon is saved and the background area blacked out then XOR and AND screen write modes could be applied liberally without the problem of interaction with the background When the icon is to be moved it would first be erased and then the background saved bitmap would be restored, following which the background area corresponding to the new position would be saved, the new area blacked out and the icon redrawn This is the method that was employed for all icons save the PIE icon, shown in figure 4 4, which remains an example of the XOR method for producing see-through icons In general it can be observed that the use of a black background box behind icons enhances Mimics greatly whereas see-through icons on a multiple gray scale background can be difficult to read

The simulation effects employed in this project include movement of substances in pipes, liquid level shifts, meter needle movement, random movement of particles, rotation of pump parts, light emitting diode indicators, digital meter and analog text displays, push buttons and potentiometers The animation techniques to achieve these effects are discussed below

The ANIMVOL and CHEMICAL icons, shown in figure 4 5, simulate liquid height in a cylindrical container Animation of level shifts is achieved by the use of ellipses and colour fill functions If the height in a container increases, then the ellipses which represent the surface of the liquid are redrawn at a higher level, the old surface ellipses are removed and the gap between the old

liquid level and the new level is filled using a flood fill function If the status or composition of the liquid changes then a colour change may be in order If this is the case then once the new ellipses representing the new liquid level are drawn then the old liquid level ellipses are erased and the contents of the container to the new level are flood erased to black and then flood filled to the new colour representing the new liquid composition or status If the liquid level decreases then the gap between the new level ellipses and the old level ellipses must be erased to black The area between the new level ellipses and the bottom of the container may be erased to black and refilled with another colour if a status change occurs to the liquid The colour filling process generally is between pixels of a specified colour and therefore when filling between levels represented by ellipses one or the other half of an ellipse may need to be erased during the flood fill and erase processes This is so that the correct concave shape may be produced for the liquid which rests against the inside of a cylindrical container

The ANIMPIPE and ROTSHAFT icons, shown in figure 4 5, simulate flow in a pipe and a rotating shaft A simulated flow effect is achieved by first drawing a rectangle to represent a pipe and then drawing a set of dots within the pipe Each dot is separated from the next in the horizontal and vertical directions by a fixed number of pixels In the horizontal direction each dot is erased and replaced by a dot in the next pixel space to the left or right, depending on the direction of flow required This is repeated at a rate which is proportional to the rate of flow required, as determined by the analog value, and its polarity, sent to the object. After a fixed number of movements of the dots in the one direction, the dots are erased and a new set drawn at the original pixel locations, and the process repeats For vertical flow the method is exactly the same except that the left and right shift are replaced by an upward or downward shift The flow rate is increased or decreased by controlling the rate at which the dots are erased and redrawn in their new pixel positions Rotation is achieved in exactly the same

manner, except that dot movement is perpendicular to the sides of the rectangular structure, which in this case represents a shaft, rather than parallel to the pipe walls, as in flow simulation. The rate of rotation and direction, clockwise or anti-clockwise, is controlled in exactly the same manner as flow in a pipe.

The DIAL icon, shown in figure 4.5, simulates an analog meter. Meter needle movement is animated by converting the value to be indicated into a corresponding angle, using trigonometric functions to calculate points for a line to represent this angle, and drawing the line. On value change the old line is erased and a new line redraw at a new angle. If the line representing the needle is drawn in screen XOR mode then the underlying meter will not be erased by the needle movement. The meter itself is simulated by two arcs with dots spaced between them to represent digit points.

The THERMOM icon, shown in figure 4.5, simulates a level meter or thermometer. A thermometer is simulated by a rectangle. The movement of the thermometer contents is simulated by placing a smaller rectangle within the thermometer rectangle, the height of which depends on the value to be represented. The inner box can then be flood filled with a colour indicative of some signalled condition.

The VALVE icon, shown in figure 4.5, simulates the opening and closing of a valve. The valve body, being a complex shape, is drawn using a polygon draw routine. However this type of shape could also be derived from a bitmap, this would alleviate the complex polygon draw otherwise required. Expansion of the valve body shape however necessitates a polygon draw routine. The animation centres on the opening and closing of the valve and is simulated by two boxes inside the valve body that expand towards one another or away from one another as the valve closes or opens, the analog value sent to the object representing the percentage open or closed via the size of the internal valve boxes. The boxes inside the valve are flood filled to represent some valve condition. When the two boxes within

the valve are touching the shut condition is represented, when they are furthest apart the open condition is represented, while in-between conditions represent the run condition A fail condition is represented with an appropriate colour change of the valve object if an active digital status is sent to the valve Conditions are also indicated by text in a text box below the valve

The ANIMPLOT icon, shown in figure 4 4, simulates a chart recorder The body of the chart recorder is simulated by a rectangle enclosing an active plot area The active plot area is offset from the main object by drawing it as an inset into the screen so that it appears to be set back from the main rectangle which forms the chart recorder (The inset effect is established by drawing a boundary about the active plot area in two sections, the upper and left side in a dark shade and the lower and right in a light shade This method of insetting areas, and the complimentary method of making areas stand out, in such a way as to give the impression of depth by having two physical levels or layers appear on the screen is used in other objects other than the plot object) Another scheme also employed, although not in the plot object, is to establish many layers or levels and hence simulate a gradual sloping back or sloping out towards another layer on the screen

To animate a plot of points representing analog data being sent to the plot object, it is necessary to call for a redraw of the plot points on the time increment for which the plot is required, for example one plot point per five seconds, or one plot point per minute The analog values sent to the object are kept in a matrix as are the times associated with these analog events When a plot is required then the values are scaled to the pixel range of the plot to produce a vertical or y coordinate position within the plot range An attribute matrix is loaded with information regarding the status of each value, for example whether the value is out of range or whether no analog value was recorded for the corresponding time The y coordinate positions, coming from the value matrix, are matched with a matrix of x coordinate positions. The x coordinate

positions are produced from scaling the total time for a full set of points in a plot evenly over the horizontal range available for the plot This accommodates a variable timebase and expansion of the plot object A plot is then drawn by drawing lines between the x,y coordinates The colour of these lines is controlled by the attribute matrix in order to signal various conditions such out of range values When a new value is to be plotted, on the next time increment, the old lines are erased by redrawing them, the matrices are shifted one place to accommodate the new data, the x and y scaling is recalculated and finally a new set of lines is drawn The PLOT object, designed in this project, allows both the number of samples in a plot and the minimum inter-sample time to be set

The ANIMCONC icon, shown in figure 4 4, simulates Brownian motion A number of dots, which is proportional to the analog value sent to the object, are drawn within a rectangle, which represents a container The x,y coordinates for the dot or pixel are generated using a random number function and scaled to be within the object boundary If the value sent to the object increases then new dots are drawn and all the previous dots are erased and redrawn at new locations If the value sent to the object decreases then the dots representing the difference are erased and each of the remaining dots are erased and redrawn at new locations in turn The colour of the dots represent some condition for the substance being simulated The dots or pixels are erased by redrawing them in the same colour as the background The rate of movement of the dots depends on how often the animation is carried out It is possible to have the animation carried out more or less often and this is controlled by a data element of the ANIMCONC object as discussed in section 4 4 above

The ANIMPIE icon, shown in figure 4 4, simulates a pump with rotating vanes The body of the pump is drawn by drawing a circle with lines for inflow and outflow pipes and a polygon plinth The vanes are represented by crossed lines The colour of everything

except the crossed lines can be changed to indicate the pump status The object is animated at a fixed rate, however the angle that the crossed lines are drawn at depends on the previous angle and the new value sent to the pump Hence on each animation the crossed lines move to a new angle depending on the analog value and its polarity With animation at a fixed rate the effect is that circular movement is faster for larger analog values and slower for smaller analog values The angles for the crossed lines are calculated using trigonometric functions and a constant which controls the angular movement per analog unit input

The PIE icon, shown in figure 4 4, simulates a pie chart The pie shape is generated by drawing an arc between zero degrees and an angle calculated from the analog value sent to the object Line drawing routines are used to close off the ends of the arc The XOR mode is used exclusively for this object The animation is effected by redrawing the pie for the previous value, thus erasing it, and then drawing the pie for the new value This object is not required to simulate movement and therefore a deadband is applied so that indiscernible angle changes are not animated No background save files are used for this object and the object is drawn over the background in XOR mode, thus producing a see through icon As mentioned previously, for XOR screen writes, interaction occurs between the icon colours and that of the background This effect manifests itself clearly when a multiple gray scale background is used

The SQUARE icon, shown in figure 4 4, simulates a binary digital status indicator A digital status signal change causes the colour of this object to change Such a colour change is employed in many of the icon objects It is implemented by redrawing the object in XOR mode, thus erasing it, and then drawing the object in a new colour

The ANIMSQR icon, shown in figure 4 4, simulates a variable voltage light emitting diode. A variable voltage LED gives out a different

colour depending on the voltage applied to it. Similarly this object processes the analog value sent to it to give a different colour depending on the value. This is achieved by using a colour fill of a box within an outer box. Additionally an active digital status sent to the object causes the object to blink. This is achieved by refilling the box using the XOR mode on every animation of the object, thus erasing and redrawing the coloured central portion of the object. If a value change occurs which warrants a colour change, then the colour of the central portion is examined by a pixel read. If it is blacked out then the box is filled with the new colour, whereas if it is in the old colour it is erased and refilled with the new colour.

The DIGVALUE, VALUETX, and STATUSTX icons, shown in figure 4.4, simulate various analog and digital text indicators. They all use the same type of screen framework within which to display their values or text. The effect is a meter or text indicator sunk into the screen from a frame attached to the front of the screen. The frame is generated by filling a rectangular area between two rectangles, one of which is enclosed within the other. Then lines are drawn between the corners of the rectangles to shape the corners of the frame. The sunk-in effect is achieved by drawing a number of ever decreasing enclosing rectangles in increasingly darker shades of gray. Again black lines are drawn in the corners across the gray shading to help achieve a depth effect. The animation simply consists of erasing the old text associated with analog or digital signals and redrawing the new text, with associated colour change for status changes.

The STATUSTILE and VALUETILE icons, shown in figure 4.4, simulate digital and analog text indicators. They use the same type of screen framework within which to display their information, as is used for the WINDOWTILE message and text input icon. This framework gives the effect of a meter or indicator panel inset into the screen. This is achieved by drawing a rectangle filled in a light gray shade, drawing within this another rectangle in a darker gray shade with a border. The top and left sides of this border are

drawn in a darker shade again, whilst the bottom and right sides are drawn in white The border gives the impression of a shadow associated with the inset panel As above the animation simply consists of erasing the old text associated with analog or digital signals and redrawing the new text, with associated colour change for status changes

The POT icon, shown in figure 4 4, simulates a potentiometer Animation of this icon is initiated by the user via mouse action Based on the user action an analog value is passed to the system by the user The icon consists of two concentric circles with ten dots equally spaced mid way between the circles and at equal angle increments between two angles at 0 and 300 degrees, as set in the object Numbers zero to nine are drawn opposite each dot outside the outer circle A line simulating the wiper of the potentiometer is drawn in XOR mode from the centre of the inner circle with a radius of the inner circle The user via the mouse can rotate the wiper to any desired angle Only angles within the potentiometer range, in this case 0 to 300 degrees, are accepted The icon also has a rectangle which displays the value selected via the wiper position The wiper and value are drawn in blue to indicate that they represent user analog input rather than simulated plant information

The SLIDER icon, shown in figure 4 4, simulates a slide potentiometer As for the icon above, this icon is utilised for user input via mouse action The icon is constructed from a long rectangle with ten equally spaced horizontal dashes to the left and running the length of the central rectangle or central column Text strings containing '0', '50' and '100' are drawn to the left of the dashes A small horizontally elongated rectangle drawn across the central rectangle represents the slide of the potentiometer A rectangle below the slide column contains the actual value selected by the slide position The user can move the slide up and down the slide column via mouse action The slide moves across the central column non destructively by the use of the XOR screen write mode

and by erasing the slide before redrawing it for each movement required As with all icons, which allow analog or digital input via mouse action, the mouse cursor must be on the icon or within an active field or active distance of the icon to effect an input The slide and value are drawn in blue to indicate that they represent user analog input rather than simulated plant information

The BUTTON and BUTTONTILE icons, shown in figure 4 4, simulate single throw buttons The animation of these icons is initiated by the user rather than by analog or digital signals from the plant or via simulation The user via the mouse selects an option to change the status of the icon The former of the two uses a framework similar to that employed by the DIGVALUE icon, whereas the latter uses a framework similar to that used by the STATUSTILE icon There are two states associated with these icons which are accompanied by two corresponding screen presentations of the icon The BUTTON icon in its off state is drawn as an outer white rectangle enclosing a black filled rectangle which in turn encloses a number of rectangles of increasingly lighter shades of gray and finally an inner green filled rectangle which contains the status word 'off' The overall effect is that of a curved sided button sitting up from the screen within a border with a centrally illuminated square indicating the status of the button The 'on' state, on the other hand, has a white outer rectangle enclosing rectangles alternating between black and ever increasingly darker shades of gray Within all of this is the same curved sided button simulated by enclosing rectangles of ever increasingly lighter shades of gray with a red central rectangle indicating the status as 'on' In this case the central button is smaller The effect of switching from the 'off' to 'on' state is that the button is perceived to move down into the screen The movement of the central button inwards is delayed while the outer shaded rectangles enclosing the smaller inner 'on' button are drawn in order to give the effect of spring loaded friction to the inward movement This is further helped by accompanying the movement with an elongated click sound The effect of switching from the 'on' to the 'off' state is that the button is perceived to

pop out from the screen There are no delays associated with this movement so as to give the impression that the button is spring load assisted while being released into the 'off' state Again a click sound accompanies the movement

The BUTTONTILE icon in its 'OFF' state is drawn as a filled outer light gray rectangle enclosing a smaller inner light gray rectangle The inner rectangle has a border of which the top and left sides are drawn in white, while the bottom and right sides are drawn in black The inner rectangle contains the status text 'OFF' The overall impression is that of two layers or tiles with the inner smaller tile or layer sitting on top of the larger outer tile In its 'ON' state the inner rectangle is reduced in size and filled with a dark gray shade, the colours of the sides of the border are swapped and the text is changed to 'ON' The inner filled rectangle appears at a level behind the large outer filled rectangle The 'on' and 'off' text are in the colours red and green respectively The effect of switching from one state to another is that the inner button is perceived to click into the large outer tile for the 'ON' state and click out to reside on top of the outer tile for the 'OFF' state Both movements are accompanied with a button click

4.6 Text, Menu and Mouse I/O

Information may be input to an application in a number of ways In this project information is input to the application via simple 'y/n' keyboard input, the WINDOW, WINDOWTILE and TEMPLATE object icons, and via the MSMENU mouse menu object

The simple 'y/n' input is called for by the PKUNZIP DOS utility [PKWA89] invoked via the *system* function call This type of input method expects a specific key or keys to be inserted to answer a query It is important to clear the input stream buffer before accepting new input in order that any extra input from the previous input session is not used as input for the next session

Another way of providing for text input and output from the application is via a window The WINDOW and WINDOWTILE icon objects provide such a framework The WINDOW icon allows a text string or message of some colour to be put up on the screen in a box which is background filled with selective colour and surrounded by a user defined coloured boundary The WINDOWTILE icon provides a two layer or tile framework, similar to the BUTTONTILE object above, where the central section appears as an inset into the screen The text appears on the inset panel The colours can be selected for inset tile or layer and text Both icons accommodate keyboard input When the window or layered window is to be drawn tests should ensure that the required vertices are not twisted, that the window will not reside outside the violable screen and that the maximum area for the window is less than the maximum data size for dynamic buffer allocation The latter requirement stems from the fact that these icon use a dynamic buffer to store the background which they write over For text input, deleting past the beginning of the input area on the icon, inputting illegal characters or giving text strings greater than the window length should be protected against and an audible warning sounded Some of the applications of windowed input in this application cover providing status reports to the user on disc access functions, input of icon identification numbers and input of background image or schematic file names

A similar framework for inputting or displaying text is via a TEMPLATE icon This style of icon provides a number of windows within the one overall icon box As for the window type inputting past the end of the template window, deleting past the beginning of the input buffer and illegal input should be protected against and include an audible warning This type of window structure is particularly useful for displaying and editing sets of data Template style icons write non-destructively to the screen This is implemented in this application by using a background save file for the TEMPLATE object, as for the other icons

Mouse input is used for selecting menu options, changing the state of digital output via digital output icons such as the BUTTON and BUTTONTILE icon, changing the value of analog output via analog output icons such as the POT and SLIDER icons, or picking icons for processing from a set on the screen

When used for analog or digital input via icons the mouse cursor must be on the icon selected or within some active field defined about the icon Dragging of the mouse cursor may be used, and an example of this is seen in the SLIDER icon which allows the mouse to slide the wiper of a potentiometer up or down For digital input a single mouse button can be used to change a button state

When picking an icon from a field of icons on the screen the distance from the mouse cursor position to each icon must be calculated and the nearest icon selected A further check can be applied to ensure that this is in fact the icon that was meant to be selected When the mouse cursor is placed near an icon and a mouse button is depressed, then the icon is erased and redrawn continually, thus indicating that the icon is to be selected Now simultaneously depressing the second mouse button indicates that the icon is to be processed If the second mouse button is not depressed then the mouse can be moved to select another icon. In this application the TEMPLATE icon and information option of the DESIGN program use this method to ensure the TEMPLATE object is created for the correct icon.

A menu can be used to select one of many options or indeed to select other menus Menus can be highlighted, as in this application, by colour filling their background When used with the mouse, a single mouse click beside a menu option can be used to select the particular option The menus in this application return an option number which is used in a *switch* statement to select the appropriate action To make selection easy the mouse movement should be limited to the direction of the options in the menu In this application, the DESIGN program, the mouse movement rectangle is the width of the cursor and the length of the menu The rectangle sits at the side of the vertically arranged options Movement is restricted to this area, and an option is invoked by clicking beside it Where more than one menu is used the topology of the menu structure depends on the application It is usually convenient to have one main menu, only one exit point back to the operating system and an intuitive flow through the menus in a system

When the mouse is used for several different types of actions utilising different cursor styles can avoid confusion In this application the pointer style mouse cursor is the default style and is used when selecting icons for processing from the screen The cross-wire cursor style is used when moving icons and the arrow style is used for menu input

5 Implementation Details

5.1 Classes and Icons.

Thirty one classes have been developed for use in the main program, DESIGN EXE, of this project These classes are shown in the hierarchy diagram of figure 5 1 The vast majority of these classes are icon style classes known as *cadshape* classes However there are other classes including the *window* type classes used for information I/O, the *shapelst* classes for generating linked lists of *cadshape* classes, and the *msmenu* class for creating interactive mouse menus

The *cadshape* category are icon type objects These icons are representations of pieces of equipment such as pumps, valves and pipes, representations of measuring instruments such as thermometers, digital and analog dial meters, representations of input devices such as buttons and potentiometers, and pictorial representations of equipment from plant schematic diagrams or photographs The *cadshape* classes provide the dynamic and static icons which make up the mimic foreground The animation of these cadshape icons intuitively imparts the plant status to the user The full set of *cadshape* type object classes which are used as foreground icons are given below All can be animated with the exception of the Bitmap icon This icon allows other icons to be overlaid on top of it so that animation can be performed

Design program Menu elements	Class CPP HPP Files	Description
Plot	ANIMPLOT	Plot of analog values
Bitmap	BMA1	Picture or schematic Bitmap
Gas	ANIMCONC	Brownian Motion gas container
Pump	ANIMPIE	Pump schematic.
Pie	PIE	Pie chart
Square	SQUARE	Two colour status square
Led	ANIMSQR	Multi colour Light Emitting Diode.
Text	DIGVALUE	Value and status meter

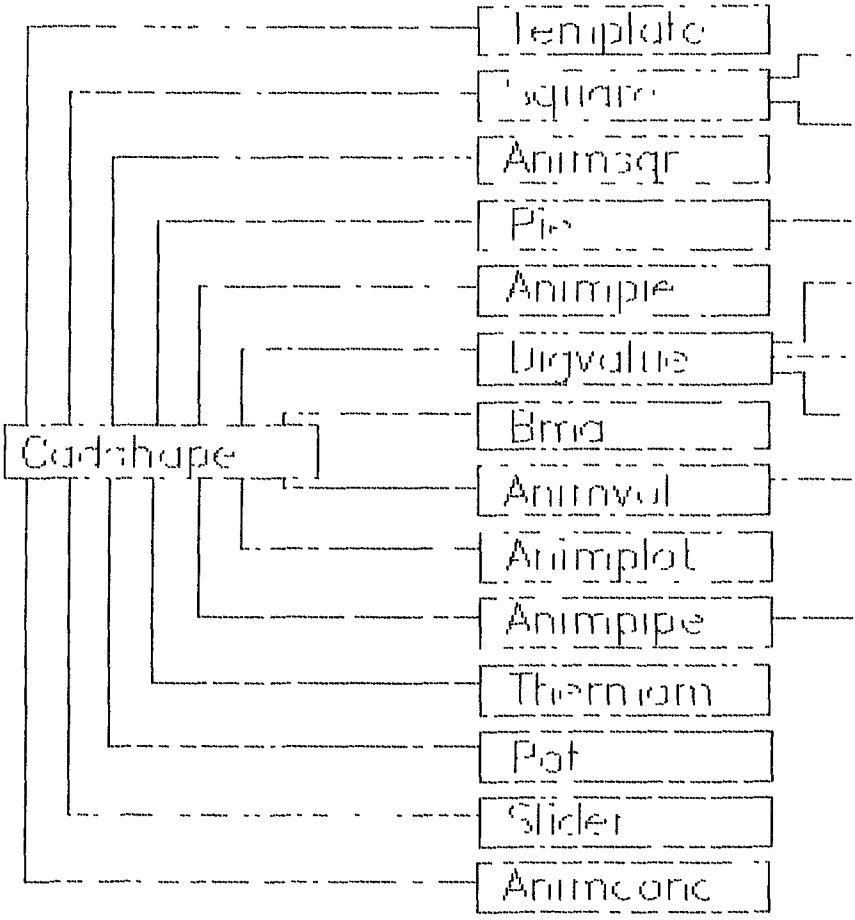
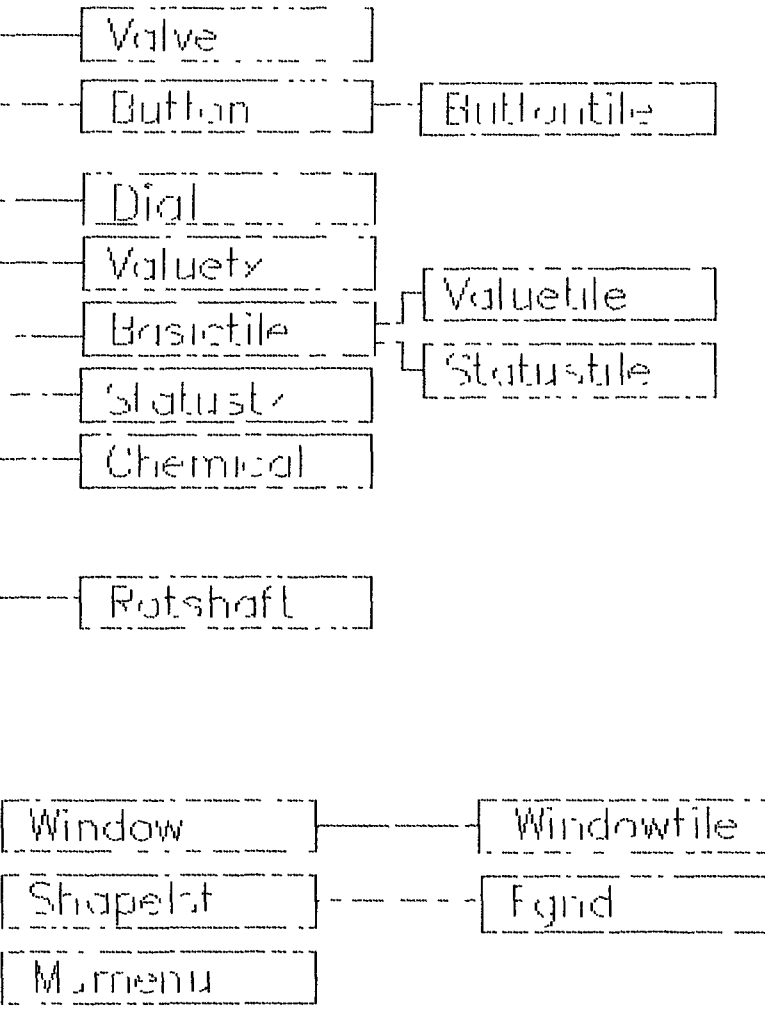


Figure 5.1 Class


Valve
Button
Buttontile
Dial
Valuety
Valuetile
Basictile
Statustile
Statusty
Chemical
Rotshaft
Window
Windowtile
Shapelst
Fgrid
Mainmenu

Hierarchy.

Design program Menu elements	Class CPP HPP Files	Description
Status	STATUSTX	Status meter
Status3D	STATUSTI	Inset status meter
Value	VALUETX	Value meter
Value3D	VALUETIL	Inset value meter
Pot	POT	Linear Potentiometer
Slider	SLIDER	Linear slide potentiometer
Button	BUTTON	Push button with click
Button3D	BUTTONTI	Inset push button with click
Tank	ANIMVOL	Tank of liquid, two colour
Chemical	CHEMICAL	Tank of liquid, multi colour
V_Pipe	ANIMPIPE	Vertical dual direction pipe
H_Pipe	ANIMPIPE	Horizontal dual direction pipe
V_Shaft	ROTSHAFT	Vertical dual direction shaft
H_Shaft	ROTSHAFT	Horizontal dual direction shaft
Valve	VALVE	Valve
Level	THERMOM	Level indicator
Dial	DIAL	Analog meter

The *cadshape* class is the base or parent class for all the icon type classes above The data members of the base *cadshape* class represent the common data elements for all *cadshape* type classes and include pointers for file names, the background save file pointer, the coordinates for the object position, the object multiplication factor and limits, extents information, current and previous digital status, current and previous analog value, counters for animation timing, *cadshape* type and identifier, and conversion factors for engineering units

The *cadshape* class contains the virtual functions indicated below These virtual functions have an empty definition in the base class and most of them are defined in each child class

virtual void pdraw() {}

This method is used to draw the *cadshape* type icon object

virtual void panimate() {}

This method is used to animate the *cadshape* type icon object

virtual void perase() {}

This method is used to erase the *cadshape* type icon object

virtual void pextent_pgen() {}

This method is used to generate a set of simulation extents based on some proposed new screen position for a *cadshape* type icon object

virtual void pextent_pdraw() {}

This method draws the extent box of a *cadshape* type icon object as a dashed box

virtual void pextent_perase() {}

This method erases the dashed extent box of a *cadshape* type icon object

virtual void analog_pinput() {}

This method allows the user input analog information to a *cadshape* type icon object

virtual void digital_pinput() {}

This method allows the user input digital information to a *cadshape* type icon object

virtual ~cadshape() {}

The class destructor If, as in this project, a new derived class object is created and assigned to a base class pointer, then a virtual base class destructor must be defined. In this case the following type of class construction is often used, *square* being a derived class from the base class *cadshape*

```
cadshape* cshape= new square();
```

The arguments for the *square* class constructor are left out for simplicity When *new square()* is performed it is impossible to be allocating anything but a *square* object Once the address of this object has been placed in *cshape*, that information has been lost and *cshape* might be pointing at a *cadshape* object or an object of any class derived from the *cadshape* object Subsequently the statement *delete cshape* will cause a problem because the actual type of the object addressed by *cshape* will have been lost, and the result will be an attempt to delete a *cadshape* object unless class *cadshape* has a virtual destructor [KOEN90]

The non virtual member functions of the *cadshape* class, which are derived by all *cadshape* child classes are detailed below

```
cadshape() {}
```

An empty constructor Each child class has its own constructor

```
void idmod (int new_id){
     /        /
}
```

This method allows the *cadshape* type object identifier to be modified

```
int idinq() {
    /    ..  /
}
```

This method returns a *cadshape* type object's identifier

```
unsigned* getcoords() {
          /.....   /
}
```

This method returns the current coordinates of a *cadshape* type object

```
void pextent_set (int extent_onoff){
     /   ...../
}
```

This method allows the *status_extn* flag to be set or reset This flag is used to determine whether extents are to be checked or not for a *cadshape* type object Relinquishing extent testing for an object allows overlapping to occur

```
int pextent_view() {
    /       /
}
```

This method returns the status of the *status_extn* flag of a *cadshape* type object

```
int pextent_test (float* extn_sim_ptr) {
    /       /
}
```

This method tests whether the simulation extents, pointed to by *extn_sim_ptr*, intersect or overlap with the extents of the *cadshape* type object for which this method is called The method returns an integer flag with value

0 if no intersection or overlapping has taken place,
1 if overlapping has taken place, and
2 if overlapping has taken place on an object that has extent protection switched off

```
int boundary_test (float* extn_sim_ptr){
     /       /
}
```

This method tests if the extents pointed to by *extn_sim_ptr* intersect with, or are outside, the viewport *fg_displaybox[]* The method returns an integer flag set to

0 if no boundary transgression has occurred, and
1 if a boundary transgression has occurred

```
float* pextent_gen (int mul_switch, float m_sim,
                    unsigned new_x_sim, unsigned new_y_sim){
        /. .... ./
}
```

This method generates a set of simulation extents, based on a proposed new position *(new_x_sim,new_y_sim)* for a *cadshape* type object, by calling *pextent_pgen()* for the *cadshape* type object for which this method is called The method returns a pointer to this set of simulation extents A switch, *mul_switch*, is provided to enable the argument *m_sim* to be utilised as an increment to the object's multiplication factor to be used in the calculation of the extents Alternatively the switch can be set so that the object's multiplication factor is not modified This enables extent testing to be carried out when the size, rather than the position, of an object is being modified

```
int expandp (float m){
    /       /
}
```

This method allows a *cadshape* type object to be expanded or reduced to the limits imposed by its data members *max_mul* and *min_mul* respectively The data members *max_mul* and *min_mul* are initialised on object construction The argument *m* is added to the objects current multiplication factor, and the method returns an integer flag to indicate whether the multiplication or reduction has reached its limits The method calls the object member function *pdraw()* to draw the expanded or reduced object

```
void modulatep (float raw_value, int state, time_t timestamp,
                long run_time){
    / .... ../
}
```

This method converts the raw transducer value *raw_value* to its engineering units equivalent by the *assignment*

*value = (raw_value*conv_m) + conv_c,*

where *conv_m* and *conv_c* represent the conversion factors for the

object It also transfers the status of a digital transducer and time information to the objects appropriate member data elements The method calls the object to animate via the member function *panimate()*

```
float analog_inputp (unsigned new_x, unsigned new_y){
      /        /
}
```

This method allows a user input an analog value to an object via the mouse The user input is calculated by the object's method *analog_pinput()* and the coordinates *(new_x, new_y)* The float value is returned to the calling program This method is functionally the opposite of the animation method for digital icons It allows the user control some piece of plant with the mouse

```
int digital_inputp (unsigned new_x, unsigned new_y){
    /       ./
}
```

This method allows a user to input an digital value to an object via the mouse The user input is determined by the object's method *digital_pinput()* The digital input state is returned to the calling program This method is functionally the opposite of the animation method for analog icons It allows the user control some piece of plant with the mouse

```
void pextent movep (unsigned new_x, unsigned new_y){
     /    .   /
}
```

This method allows movement of the extent box of a *cadshape* object

```
void movep (unsigned new_x, unsigned new_y){
     /. ....  /
}
```

This method moves the *cadshape* type object, for which it is called,to the new position *(new_x, new_y)* It uses the member function *pdraw()*

```
void moveprel (unsigned new_x, unsigned new_y){
     /.. ...../   }
```

This method moves the *cadshape* type object, for which it is called, to a new position which is relative to its old position by the

coordinates *(new_x, new_y)* It uses the member function *pdraw()*

```
unsigned long prange (unsigned xr, unsigned yr){
        / .     /
}
```

This method returns a measure of the distance between the point *(xr,yr)* and a *cadshape* type object's position This method is used when it is required to find the closest *cadshape* type object on the screen to the mouse cursor position

The *cadshape* icons in general redefine the functions indicated below in addition to their own constructor and destructor

void pdraw()
void panimate()
void pextent_pgen()
void pextent_pdraw()
void pextent_perase()
void perase()

Each class defines a private or protected data set The protected set allows data inheritance to any derived class

There are exceptions to the redefinition of functions as described above and typical examples from this set of exceptions are now detailed The *basictile* class defines a new function, *void basic_pdraw*, and an empty *perase()* function only The *basic_pdraw* function draws the frame of an inset meter or switch The *button* class defines the function *void digital_pinput()* instead of the function *panimate()* This is for digital input via the mouse and the subsequent changes in the icon, instead of animation The *buttontile* class defines the functions *void pdraw()*, *void digital_pinput()* and *void perase()* only The *valve* class does not define the *void pextent_perase* function The *digvalue* class redefines all the *cadshape* virtual functions and in addition defines the function *virtual void basic_pdraw()* The *statustile*, *valuetile*, *statustx* and *valuetx* classes only redefine the *void perase()*, *void panimate* and *void perase()* methods The *pot* class defines the method *void analog_pinput()* instead of the *void*

panimate() method, because an object of this class allows analog input via the mouse and subsequent icon changes in place of conventional animation which occurs on a plant status or value change

In many of the cases above the classes which don't define functions that are defined by their parent classes rely on the use of parent class methods and the inheritance mechanism

Other *cadshape* type classes include the *template* class, which is described below, and the *basictile* class The *basictile* class is the base class for the *valuetile* and *statustile* classes and is derived from the *digvalue* class It has the following member functions

basictile () (),

The constructor creates an instance of the class *basictile* and generates a random file name, BT**** SAV, which is used to save the background behind inset or tile style objects

~basictile(),

The destructor calls the *digvalue perase()* member function and deletes the object of class *basictile*

basic_pdraw();

This method saves the background, where the inset or tile style object will reside, to the background save file It then constructs and draws the frame for inset type objects

The *template* class is also a *cadshape* class An object of this class allows the user access to a *cadshape* class object's private data A *template* object can be created for any cadshape icon object The *template* object has a set of fields which hold and display, within an overall box, selected data items from the *cadshape* These items can be edited, thus editing the private data associated with a *cadshape* This provides a means of changing

status textwords, engineering conversion units and icon identifiers of objects in the mimic foreground The member functions of the *template* class are detailed below

*template (cadshape * cad, unsigned x, unsigned y),*

The constructor creates an instance of a *template* class The constructor establishes the number of fields in the *template* object based on the type of *cadshape* object, (e g DIP,DOP,AOP, or AIP) for which the *template* is generated The area of screen required to display the *template* object consisting of the various field boxes is calculated and a random file name, TP**** SAV, is created This file is used to store the background behind the *template* when the *template* is drawn

~template(),

The destructor deletes the instance of the *template* class and explicitly erases the template, restores the background to the screen and deletes the background storage file

void valuefld (int vm,float conv),

This method converts the float *conv* to a null terminated string and places the string in the *template* field *field [vm][]*

void valuefld_s (int vm,float conv);*

This method leaves the float pointer *conv* pointing to a float which has been generated from the null terminated string in the *template* field *field [vm][]*

void textfld (int vm,int vn,char text),*

This method transfers the null terminated string starting at *text[vn]* to a null terminated string starting at *field [vm][vn].*

void textfield_s (int vm,int vn,char text);*

This method transfers the null terminated string starting at *field [vm][vn]* to a null terminated string starting at *text [vn].*

char test_input (cadshape cad, int fld),*

This method is used to ensure that user input type, during the edit of a *template* object field, matches the *template* object type If it does not then the user input character is replaced by an invalid character with the result that the user must retype the character

The *template* object field which contains the *cadshape* object *type* field is used in a switch statement with the *template* object field number and macros to test if ASCII and digit type characters are used for the appropriate fields

void pload (cadshape loadcad),*

This method loads the selected *cadshape* object's data into the *template* objects fields First the *cadshape* object type and identifier are converted to strings and placed in *template* object fields Then depending on the *cadshape* type the *textfld()* and *valuefld()* member functions are used to convert the *cadshape* object data into strings and place the strings in the appropriate *template* object fields

void pstore (cadshape storecad);*

This method stores the data from the *template* object fields to the *cadshape* object First the *cadshape* identifier *template* object field is converted to an integer and stored back to the *cadshape* object identifier Then using *textfld_s()* and *valuefld_s()* member functions the *template* object fields are converted from strings to the appropriate data types and stored back to the appropriate *cadshape* object member data elements

void pdraw();

This method draws the template and its fields First the coordinates of the box containing the *template* object are modified if necessary so that the *template* resides within the viewport The background, which occupies the area where the *template* object is to be drawn, is stored to the background save file The field boxes

for *cadshape type* and *identifier* are drawn The appropriate fields are displayed for the *cadshape* object for which the *template* object is invoked Finally the *exit* field is drawn

```
int pedit (cadshape * storecad,fg_coord_t current_x,
          fg_coord_t current_y);
```

This method allows the user to edit the template field contents First the member function allows a particular field to be chosen by mouse cursor action No editing is allowed for either the *exit* or *cadshape* object *type* fields On all other fields editing of the field strings is allowed with the usual text input facilities These edits are applied to the *template* fields and could be easily modified to be stored back directly to the *cadshape* object data members On choosing the *exit template* field the *pedit()* method is exited and an integer flag returned to the calling program

void perase(),

This method causes the *template* object to be erased from the screen and the contents of the background save file to be restored

The *window* type class objects and the *windowtile* class objects, derived from the *window* type, provide for display of status or error messages and input information in a window In addition to the constructor and destructor the WINDOW class provides the following methods

```
void  open (unsigned llx, unsigned lly, unsigned urx,
           unsigned ury, fg_color_t b_color,

           fg_color_t l_color);
```

This allows a window to be opened on the screen with lower left coordinates *(llx,lly)*, upper right coordinates *(urx,ury)* with boundary of colour *l_color* and background within the window of colour *b_color* Errors such as twisting of the window coordinates, box area less than the size of a single character, vertices outside the *fg_displaybox* coordinates and excessively large windows are all

handled The background behind the window is saved to a dynamic buffer which is allocated within this method

*void text (char * msg, fg_color_t t_color),*

This method allows a message in colour *t_color* to be displayed in the window

fg_coord_t text_position(),

This method returns the current available text position within the window

void erase(),

This method erases the text within the window

void close(),

This method erases the window and replaces the background that was overwritten by the window when it was opened This method is not the destructor The dynamic background buffer is deleted It should be noted that no background save files are used in this class and that a dynamic buffer is created for the purpose of saving the background behind the window This buffer exists for the duration that the window is open

The *windowtile* class is derived from the *window* class and provides for a 3D style window

```
class windowtile . public window {
/*          */}
```

Shapelst type objects contain a linked list of pointers to *cadshape* type objects and allow pointers to be added to, or deleted from this list Other functions or methods to search the list, move to the next pointer and go back to the start of the list are included The facility to draw and erase all cadshapes in a *shapelst* list is also included. The *shapelst* class creates within itself a *shapelst_el* class An object of this class holds a pointer to a cadshape object and a pointer to the next *shapelst_el* class object. The *shapelst* class is defined as a *friend* of the *shapelst_el* class

so that it can have access to *shapelst_el* private data Virtual functions must have a definition in the base class *Shapelst* class methods include

```
virtual void draw() {}
virtual void erase() {}
virtual void move (unsigned x, unsigned y) {}
virtual void expand (float m) {}
```

shapelst(),

The constructor resets the current and head pointers of a linked list of *shapelst_el* objects to an *new* instance of the *shapelst_el* class All pointers are NULL

~shapelst(),

The destructor frees the memory used for the linked list

*void insert (cadshape * s);*

This method inserts a new *shapelst_el* into the list with a pointer to a *cadshape* object

void reset(),

This method resets the current *shapelst_el* pointer to the head pointer

*cadshape * headsh(),*

This method returns the *cadshape* pointer corresponding to the head *shapelst_el*

*cadshape * test();*

This method tests that there exists a *cadshape* pointer entry in the *shapelst* object and returns this pointer, otherwise a *NULL* pointer is returned

*cadshape * next();*

This method returns the *cadshape* pointer corresponding to the current *shapelst_el* object and moves the current pointer on to the next *shapelst_el* pointer

*cadshape * findid (int a),*

This method finds the *cadshape* pointer in the list which points to the *cadshape* with identifier *a* and returns this pointer

*void remove (cadshape * s);*

This method finds the *shapelst_el* with the cadshape pointer *s*, removes this *shapelst_el* link from the list and relinks the list

*cadshape * nearest (unsigned x, unsigned y);*

This method hunts through the list of *shapelst_el* objects to find the *shapelst_el* object whose *cadshape* coordinates are closest to coordinates *(x,y)* and returns the pointer to this *cadshape* object

The *fgnd* class is a *shapelst* type class which is derived from the *shapelst* class and forms the structure to hold the mimic foreground of icons In addition to the methods it inherits from the *shapelst* class as above, it provides other methods which include

fgnd (unsigned x, unsigned y),

This is the constructor for an object of class *fgnd*

~fgnd (),

In addition to its role of *fgnd* destructor the destructor causes each *cadshape* object to be removed from the foreground and subsequently deleted

void draw (unsigned x, unsigned y);

This method invokes the *cadshape·:pdraw()* function of each *cadshape* object pointed to, via the linked list entries of the *fgnd* object As detailed in sections 4 3 and 5 3, precedence is given for objects which allow other objects to be overlaid on top of them.

void erase(),

This method invokes the *cadshape·:perase()* function of each *cadshape* object pointed to, via the linked list entries of the *fgnd*

object As detailed in sections 4 3 and 5 3, precedence is given for objects which allow other objects to be overlaid on top of them

void move (unsigned x, unsigned y),

This method is designated for future use where the coordinates *(x,y)* may be used to shift the coordinates of each object in the FGND object and hence move the entire foreground At present it simply redraws the foreground and does not effect the positions of the *cadshape* objects within the foreground

void expand (float m),

This method expands all objects in the foreground *fgnd* object by calling *cadshape expandp(mul),* for each object, with increasing values for *mul*

void modulate (int id, float percent, unsigned state,
time_t timestamp, long run_time);

This causes the *cadshape modulate(),* method to be called for each cadshape in the foreground, effectively animating the foreground with the given arguments

int extents_test (float extents_sim_ptr,*
*cadshape * moving_shape);*

This method tests if the extents associated with a proposed new position of the *cadshape* object *moving_shape* intersect or overlap on the extents of any other object in the foreground which has extent protection switched on If this object overlaps with an object that has extents testing switched off (an overlapping type) and is itself an overlapping type, then an extents violation is flagged by the returned integer value A violation is also flagged if it overlaps with any object that has extent testing switched on

unsigned long range (unsigned x, unsigned y);

This method, intended for future use, returns a measure of the distance from the point *(x,y)* to the foreground objects root point, *(x_center,y_center).* This could be utilised, if several foregrounds

were scaled and placed on the screen simultaneously, to find the nearest foreground to a mouse cursor position in order to select a particular foreground for further processing

The *msmenu* class provides for the construction, destruction, drawing and erasing of a menu It also saves the background behind it when it is drawn, maintaining the background integrity, and allows selection of a menu item via the mouse and the Microsoft mouse routines included by virtue of the include file MSMOUSE H

There are also methods to convert from the Microsoft mouse coordinate system to that used by the Flash Graphics system, to change mouse cursor styles, and to interpret mouse button commands There is also a boundary check method to ensure that the entire menu is always within view on the screen

msmenu (struct menu_s mp),*
The constructor creates a menu based on the *struct menu_s.* This structure gives the menu text and an *item_number* for each text item in a menu There are some *menu_s* structures defined at the start of the program DESIGN CPP The pixel size of the menu is calculated and a dynamic buffer, used to save the background behind the menu, is allocated Finally the mouse initialisation routine is called

~msmenu(),
The destructor explicitly deletes all dynamic buffers allocated by the *msmenu* constructor and terminates the mouse

void drawmenu (int x, int y),
This is a private method of the *msmenu* class and is called by the member function *get_selection (unsigned x, unsigned y).* It outputs each of the menu element names to the screen

int get_selection (unsigned x,unsigned y);
This method ensures that the menu will reside within the viewport *fg_displaybox[],* saves the background behind the menu to a dynamic

buffer and draws the menu via the member *drawmenu()* The mouse cursor is changed and a menu element can then be chosen via the mouse The mouse cursor style is reset, the screen behind the menu is restored and the item number of the menu element is returned to the calling program

void translate_coords (unsigned x, unsigned* y);*

This method translates the mouse coordinates to Flash Graphics coordinates The mouse coordinate system sees the lowest y coordinate at the top of the screen, while the Flash Graphics system sees the highest y coordinate at the top of the screen

*void boundary_check (unsigned * x, unsigned* y int type),*

This method is called by the *get_selection* method to ensure that the menu, regardless of where it is called to be drawn on the screen, always remains viewable on the screen in its totality The constants *BUFFER_INNER* and *BUFFER_OUTER* can be used to set the closeness of the menu to the extents of the viewport *fg_displaybox[]*

void default_cursor(),
void menu_cursor(),
void cross_cursor(),

These methods are used to select a particular cursor style for the mouse Static integer arrays are defined in MSMENU CPP, and are used within these methods with the Microsoft mouse method *msm_setgraphcur (int hot_spot_x, int hot_spot_y, int* style_array)*

void wait_left_pressed (unsigned x,unsigned* y),*
void wait_right_pressed (unsigned x,unsigned* y);*
void wait_left_released (unsigned x,unsigned* y);*
void wait_right_released (unsigned x,unsigned* y);*

These methods use the Microsoft mouse method, *msm_getstatus()*, within a while loop to wait for mouse button actions

5.2 The Class Hierarchy

The set of classes designed and used in this project may be divided up into four categories based on their parent base class. The first and largest category is made up of the *cadshape* type classes. The second category is the *window* type class category. The third category is made up of the *shapelst* type classes, and the final category consists of the *msmenu* class.

As shown in the class hierarchy diagram of Figure 5.1 the *msmenu* class has no child classes, the *window* and *shapelst* classes both have a single inheritor, whereas the *cadshape* base class is the parent for many child classes, some of which have their own child classes and some of which act as parent bases for more than one child class.

An example of an inheritance tree in the *cadshape* type category is the *digvalue* class. The method inheritance structure of this tree is shown in figure 5.2. The *digvalue* class inherits member functions and data from the *cadshape* class, but is also a parent class for the *valuetx*, *statustx* and *basicti* classes. *Basicti* is in turn the parent base class for the *valuetile* and *statustile* classes. The *digvalue* class has its own definition of the following virtual functions, which are defined as empty functions in the *cadshape* class. The arguments are omitted.

```
virtual void pdraw(),
virtual void panimate();
virtual void pextent_gen(),
virtual void pextent_pdraw(),
virtual void pextent_perase();
```

Additionally the *digvalue* class defines the member function.

```
virtual void basic_pdraw();
```

The classes *valuetx* and *statustx* only redefine virtually the member functions *pdraw()* and *panimate()*. These *pdraw()* functions call the

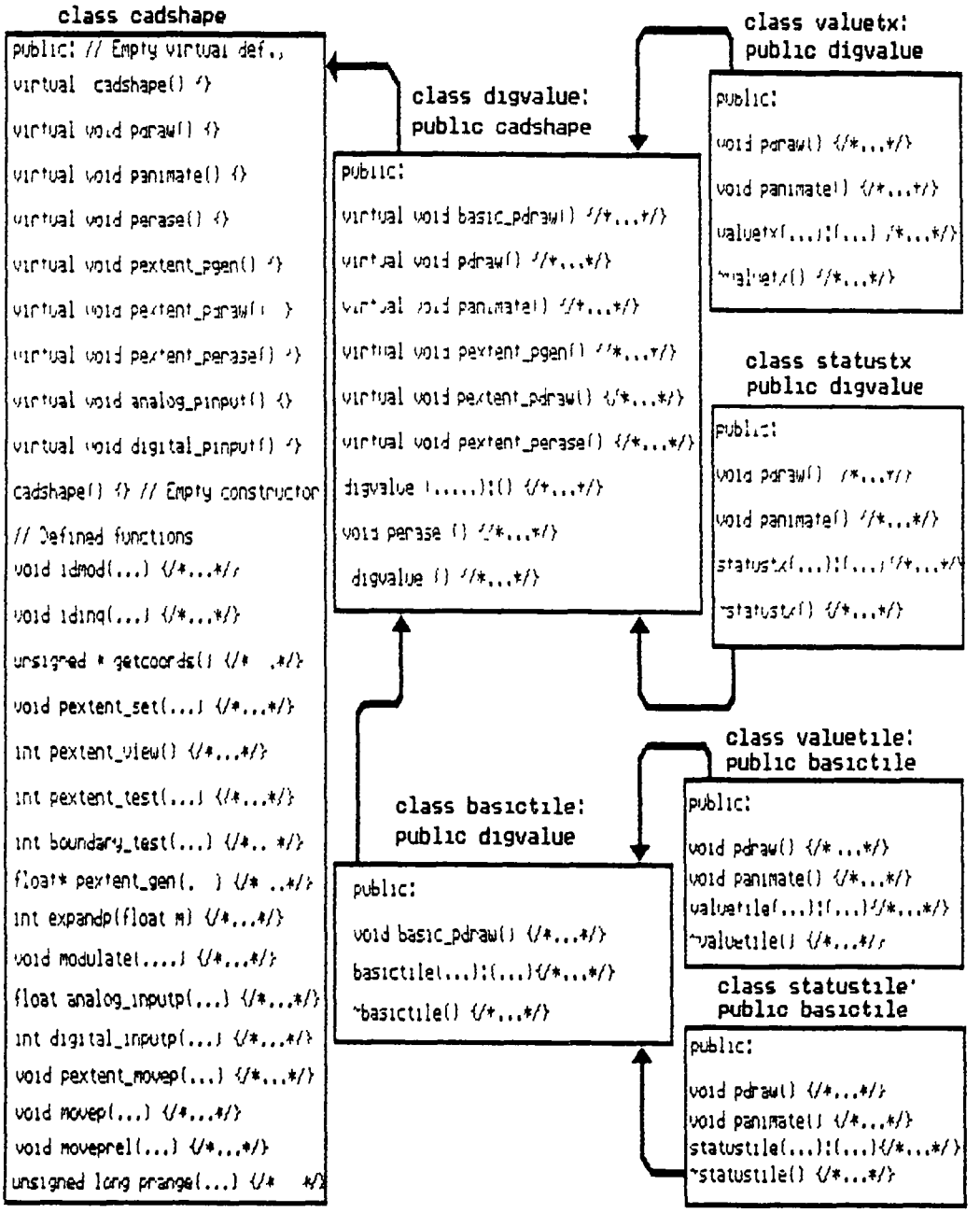


Figure 5.2 Method Inheritance.

function *basic_pdraw()*, and as this is not redefined for this class the function which is called is *digvalue .basic_pdraw()* The data member inheritance from the *digvalue* class *protected* data section is so strong that no new variables are defined in the *valuetx* and *statustx* classes, and the inherited data members are initialised in the respective constructors The *basic_pdraw()* function draws on the screen a framework for the presentation of values or digital status of a transducer There are two frameworks, the *digvalue* box type and the *basictile* 3D type, which are drawn on the screen by their respective *basic_pdraw()* functions The class *basictile*, derived from the *digvalue* class, redefines the virtual function *basic_pdraw()* This class does not define any new data members for itself The *valuetile* and *statustile* classes derived from the *basictile* class, inheriting all the class members and functions before them, do not define any new data members but do redefine the *pdraw()* and *panimate()* virtual functions They also use the *basic_pdraw()* function, in which case the function called is *basictile basic_pdraw()*

There are child classes of the *cadshape* class that do not have child classes themselves These are the *template*, *animsqr*, *animpie*, *bma*, *animplot*, *thermom*, *pot*, and *slider* classes All of these classes, with the exception of the *template* class, define the functions listed below There is a heavy dependence on data member inheritance from the *cadshape* class

pdraw()
panimate()
pextent_gen()
pextent_draw()
pextent_erase()
perase()

However the *pot* and *slider* classes also define the function *void analog_pinput()* and, like the *bma* class, define the *panimate()* as an empty function The *template* class redefines the virtual functions *pdraw()* and *perase()* and defines some functions of its own. It also uses inherited data members from the *cadshape* class

The *square* class inheritance tree is the second largest inheritance structure in the project The *square* class inherits data and function members from the *cadshape* class The *valve* and *button* classes in turn inherit data and functions from the *square* class Finally the *buttontile* class inherits data and function members from the *button* class The valve class has its own data members and inherits data members from the *square* class However it redefines all *cadshape* virtual functions apart from the function *void pextent_perase()* which it inherits from the *square* class The *button* class also inherits this function from the *square* class and defines, for the first time in this inheritance string, the virtual function *void digital_pinput()* *Button's* own data members, as for many of the classes in the project, are defined in a *protected* section so that they may be available to any classes derived from this class in the future The *buttontile* class inherits data and function members from the *button* class It defines its own private data and only redefines the *pdraw(), digital_pinput()* and *perase()* virtual member functions This is because the only differences between the *button* and *buttontile* class objects are the look of the object and the effect on the object when the user, via the mouse, enters a digital input The *buttontile* class is strongly dependent on inherited data members

The *window* category has one child class, the *windowtile* class The *window* class has the functions detailed below The arguments are omitted

void open()
void text()
void erase()
void close()

The windowtile class inherits data and function members *erase()* and *text()* from the *window* class These classes facilitate the input and output of messages to the screen The screen style of the framework surrounding the text is different and so the only derived functions are those relating to outputting and erasing of the actual text

The *shapelst* class has one child class, the *fgnd* class As described in section 5 1 the *shapelst* class defines a linked list of *cadshape* type classes, with linked list processing functions The *fgnd* class inherits the linked list structures from the *shapelst* class and defines functions to allow the *cadshape* objects pointed to by the linked list entries to be draw on the screen, erased, moved, expanded, reduced and animated as a set of objects The foreground class *fgnd* with its *shapelst* inheritance allows a foreground of dynamic icons to be generated and manipulated in terms of screen functions via the *fgnd* functions and in terms of a linked list via the *shapelst* functions The *fgnd* class defines some of its own data members as well as inheriting data members from the *shapelst* class

5 3 The Design Program.

The DESIGN EXE program allows a search for all background image or schematic libraries and the subsequent selection of a library and image, or schematic, from that library as a background for a Mimic The program then allows chosen icons to be constructed, as objects on the heap, forming the mimic foreground Icons can be drawn, erased, moved, expanded or reduced Icons can be moved by extent movement or by moving the icon in its entirety Expansion or reduction of an icon is bounded by a minimum and maximum size The chosen set of icons can be erased and redrawn on mass When the foreground is chosen the active icons can be animated, thus animating the mimic The program uses a mouse for selection of menu options and the keyboard for text input via windows

The program includes various header files for function declarations These include Flash Graphics, BIOS, DOS, and mouse functions, which are linked into the application via the libraries The *fgnd*, *window*, *template* and *windowtile* object header files are also included External menu structs and enumerations are defined Menu and foreground constructors are called generating external objects

The *main(argc,argv)* function is called followed by various data declarations, definitions and function declarations Menu constructors are called for menus used within the scope of *main()*

The graphics mode is initialised by a call to the routine *fg_init_palette()* The program FGINIPAL CPP was written to provide for palette manipulation and offers routines to initialise the palette for various resolutions and colour assignments The palette decided upon for this application has eight shades of gray with eight colours Essentially the grays are used for the background and the colours for the icon foreground The *fg_init_palette()* routine initialises this graphics mode and sets the resolution at the standard VGA specification of 640x480 pixels

The boundaries for mouse cursor movement are then set for the application to within the screen coordinates, as defined by Flash Graphics *fg_displaybox* coordinates

A message window is opened by constructing a window object, and by invoking one of its methods with a suitable argument, and an introductory windowed message is output to the screen as shown in figure 5 3 This message includes the version number of the program After a suitable length of time the message is erased and the window is closed

The background ZPI file is chosen from a selected library ZIP file and subsequently displayed by either specifying these files on the command line on calling DESIGN, which in turn calls and passes its arguments to BDISPLAY, or within the DESIGN program via the routine SELECTBG and its menus as described in section 4 1 The ZIP and ZPI menus are shown in figure 5 4 The WINDOWTILE objects used for entry of the required library ZIP file and background ZPI file are shown in figure 5 3

The program now enters its main processing loop with a menu presented on the screen This menu is called the function menu, shown in figure 5 4, and its options are

New Icon A	Construct and draw a new icon
New Icon B	" " " " "
New Icon C	" " " " "
New Icon D	" " " " "
Move	Move entire selected icon
E_Move	Move selected icon by moving its extents
Size	Expand or reduce a selected icons size
Delete	Erase and free memory used by selected icon
Wipe	Erase and free memory used by all icons
Clear	Erase all icons
Show	Draw all erased icons
Info	Display and edit of data for selected icon
Animate	Animate all active icons
A_Input	Allow analog input on selected AOP icon.

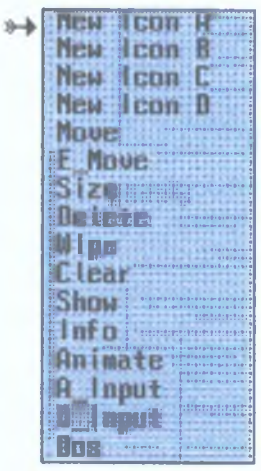
Program Introductory Message.

Text I/O Windowtile objects

Figure 5.3 Window and Windowtile Objects.

Zip Menu.

Zpi Menu.



Main Menu.

Figure 5.4 Menus.

D_Input	Allow digital input on selected DOP icon
Dos	End application and return to DOS

On selecting *New Icon A, B, C* or *D* the corresponding *menugns*()* function is called which displays one of the icon menus shown in figure 5 5 The MENUGNS* CPP program includes the header file MENUGNS* HPP which itself includes the header files for the icons which can be selected within this routine, the foreground class header and the header files for the window style classes There are also various data and functions declarations within MENUGNS* HPP As an example of this set of routines the *menugns4()* routine will be expounded upon The routine first sets up data for the icon which will be selected This includes for example a default *raw_value* for a transducer analog input and default *onoff* status for a digital transducer or some status associated with limits processing for an analog transducer input This achieved, a menu is presented to the user In this case the menu has the following options

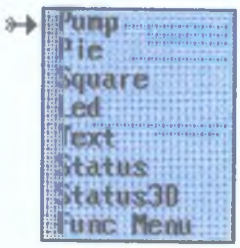
Plot	Select a PLOT icon
Bitmap	Select a BITMAP icon
Gas	Select a GAS icon
Func Menu	Return to DESIGN function menu

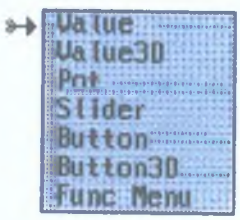
On selection of any one of the icons, the constructor is called and an object is generated via the new operator on the heap For example on selection of the *Plot* option

```
cadshape* s;        //Declared in MENUGNS$ HPP
s=new animplot(x,y,id,mul,raw_value,onoff,,,extent_onoff),
```

Then a function from MENUGNFU CPP is called

```
cadshape_insert(unsigned x,unsigned y,cadshape *s,int series){
    foreground insert(s),
    place (x,y,s,series),
    id=id_input(),
    s->idmod(id);
}
```





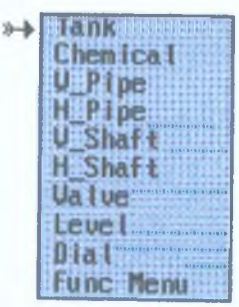


Figure 5.5 Icon Menus.

The routines *place()* and *id_input()* are also defined in MENUGNFU CPP In *cadshape_insert()* the icon is inserted into the foreground object as a pointer linked list entry, the icon is autoplaced on the screen and the icon is given a user specified identifier The auto placement of the new icon so that the extents of any other object on the screen or the boundary areas are not overlapped is discussed in section 4 3 The function *id_input()* allows user input via the keyboard of an integer value ≤ 65535 for the icon identifier The input is effected via a *windowtile* object as shown in figure 5 6

On exit from the *cadshape_insert()* function the application returns to the *menugns4()* function and the options to either select an icon or return to the function menu of DESIGN are presented On choosing the *Func Menu* option the routine is terminated and returns to the DESIGN program function menu loop

On selecting the *Move* option of the function menu in DESIGN CPP, if there exist icons in the foreground as tested by *fgnd .test()*, the mouse is used to select a particular icon via *fgnd ·nearest()* and mouse button press detection routines An extents test is carried out so that icons allowing overlapping to occur on themselves by other icons conform to ordered movement of these overlapping icons and themselves, as seen in the section 4 3 which deals with extents Provided this test is passed, the icon is moved by successive calls to the functions *cadshape::pdraw()*, *cadshape_movep()* and *cadshape_perase()* Mouse coordinate *msm_getstatus()* and coordinate translation *mouse translate()* functions are used to move the icon about the screen Throughout the icon movement extent testing is carried out as detailed in the extents section The effect of this option is to drag, using the mouse, an icon across the screen When extent overlapping occurs movement of the icon is suspended till the mouse is at a position at which the extents of the selected moving object clears the stationary icon The exception is stationary objects that allow overlapping In this case an object can be moved over and placed on top of the stationary object

Enter Cadshape Identifier

Figure 5.6

The *E_Move* option in the function menu is similar to the *Move* option except that in this case *cadshape pextent_pdraw(), cadshape. pextent_movep()*, and *cadshape pextent_perase()* are used instead of *cadshape pdraw(), cadshape movep()* and *cadshape perase()* This results in the extents rectangle being dragged across the screen by the mouse instead of the entire icon The second difference between this option and the *Move* option is that this option provides a window on the lower left corner of the screen which gives the overlap status The extents move technique used in this option means that the icon need only be redrawn when the destination site is decided upon, unlike the *Move* option where the icon is continually being redrawn When an icon is drawn on the screen then the bitmap of the background behind the icon is stored in a dynamic buffer in order to preserve the background If the buffer is kept active for the life of the object then when an erase or delete of the object is required, the object is first erased leaving a hole or black rectangle in the background, the object background buffer which has the saved bitmap is redrawn on the screen, thus the background is reconstituted When the object is being moved a new bitmap is saved in the background save buffer and then the icon is drawn over this area Keeping the background dynamic buffers open for the life of the icon and for all icons represents quite a strain on heap memory, which will in fact eventually run out and thus limit the number of icons that can be active at the one time This problem was overcome by the use of background SAV files as seen in section 4 2 When the bitmap is saved to the dynamic background buffer it is transferred to the object background file and the dynamic buffer is deleted This reduces the requirements for heap memory dramatically but limits the speed of redraw for an icon due to disc access characteristics This is overcome to a very large extent by using a Virtual or RAM Disc for the SAV files which eliminated the disc access delays The speed of dragging an icon in the *Move* option is to some extent slower than the extent drag in the *E_Move* option.

The *Wipe* option in the function menu first calls the *fgnd::erase()* method which erases all icons in the *foreground* instance of the

FGND class The order of erasing is such that all objects which are overlapping on top of other objects are erased first and then the objects which were underneath are erased

```
// Precedence for extent overlap types
reset(),        // Start at top of list
while ((cadsh = next()) != 0) {
    if (cadsh->pextent_view()) cadsh->perase(),
    }

reset(),
while ((cadsh = next()) != 0) {
    if (!cadsh->pextent_view()) cadsh->perase();
    }
```

Then, while there are still icon pointers in the *foreground* object, each icon entry is removed and the memory initially provided by the *new* operator when the icon was constructed, is freed

```
while ((cl=foreground test())!=(cadshape*) 0) {
   foreground remove(cl),
   free (cl),
   }
```

It is interesting to note that the *free* operator was used and that the *delete* operator was not This is because the *delete* operator calls the object destructor explicitly, whereas *free* does not but still frees up the heap used by the object when it was allocated using the *new* operator Because each destructor contains a call to the *perase*() function for the particular object, and some objects are first level and second level derived, then explicitly invoking the destructor for an object will call the *perase()* not only for the object in question, which has already been erased, but for all its parents in line back to the base object from which they were derived. This clearly is not required as the destructor will be called on scope resolution anyway and memory can be freed at this stage by a simple call to *free()*

The *Clear* option tests if the foreground is clear and if not simply calls a *fgnd erase()* for the *foreground* instance of the *fgnd* class and sets a flag to indicate that the foreground is now clear

The *Show* option tests for a clear foreground and if clear draws all icons on the screen

```
if ('clear) {
   foreground.draw(x,y);
   clear=1, }
```

The the *fgnd draw()* method draws objects which allow overlapping of other objects over themselves first and then draws the other objects Extracts from this method show how this is achieved

```
reset(),
while ((cadsh=next()) '=0) {
   if ('cadsh->pextent_view()) cadsh->pdraw(), }
reset(),
while ((cadsh=next()) '=0) {
   if (cadsh->pextent_view()) cadsh->pdraw();}
```

The *delete* option of the function menu first tests to ensure there are icons in the foreground which can be deleted, using the method *fgnd ·test()* If icons exist the mouse cursor is set to a cross and can be used to select a particular icon

```
mouse cross_cursor(),
mouse wait_left_pressed(&x,&y),
mouse translate_coords(&x,&y),
cadshape* rm = foreground nearest(x,y),
```

The particular icon pointer is removed from the foreground list and is erased Memory allocated by the *new* operator is subsequently freed for further use using the *free* operator

```
foreground remove(rm);
rm->perase();
free(rm),
```

The discussion for using the *delete* and *free* operators is as seen above for the *wipe* option The mouse cursor is set back to the pointer style

mouse default_cursor(),

The *A_Input* option of the function menu allows a particular icon, given that there are icons in the foreground, to be chosen The method *cadshape analog_inputp()* is applied for the duration that one of the mouse buttons is pressed The effect of this method on the icon depends on the particular method implementation For example the method *slider analog_inputp()* allows the user position the slide bar of a slide potentiometer The mouse cursor must be within a certain distance of the slide bar for mouse movement to have any effect on its position The overall effect is that the mouse is used to push up or pull down the slide bar causing the input to the icon of an analog value proportional to the position of the slide bar A similar arrangement is used in the method *pot analog_inputp()* However in this case the knob of a normal circular potentiometer is mimicked

The *D_Input* option of the function menu allows a particular icon, given that there are in fact icons in the foreground, to be chosen The method *cadshape digital_inputp()* is applied for the duration that one of the mouse buttons is pressed After each call to this method a timer is invoked so that the user can toggle continuously between states for a digital output type icon The effect that *digital_inputp()* has on the icon depends on its implementation For example when this option is invoked for a button object the effect is that when the mouse is activated on or near the icon then the status of the button toggles This status change is indicated by a text change and the button appears to have been pressed in or released While the mouse is activated on the icon, the status changes and the button pops in and out continuously

The *Size* option of the function menu allows a particular icon to be chosen from the foreground, given that icons exist in the

foreground The chosen icon is expanded or reduced by the use of the *cadshape· expandp()* method Extents checking for overlap with other objects or the screen boundary are tested for, as in the *E_Move* option The resizing of an icon will not take place if it will result in an extents or boundary infringement The extents status is shown in a window at the lower left corner of the screen The method *int expandp()* returns an integer flagging whether or not expansion or reduction has reached its limiting values The limiting values are declared in CADSHAPE HPP, inherited by all the icons and initialised in their respective constructors No further resizing is carried out on the object if the limits are reached unless there is a change in the status of the mouse buttons to cause resizing in the opposite direction If one of the mouse buttons is pressed then, depending on the button, the icon is either expanded or reduced in size If one button is pressed, say expanding the icon, then holding this button down while the other is pressed causes the expansion to continue but at a much slower rate The same type of operation occurs for reduction of the icon

The *Animate* option of the function menu causes the *panimate()* method for each icon in the foreground to be called The animation effect in each icon depends on the implementation of the method for the icon as discussed in section 4 5 The animation is carried out by a call to the method *fgnd. modulate()*

```
void fgnd. modulate(int id, float percent, unsigned state,
                    time_t time_stamp, long run_time) {
    /*        */
    value=percent,
    status=state;
    timestamp=time_stamp;
    run_time=runtime,
    reset();
    while ((cadsh=next()) !=0){
          cadsh->modulatep(value,status,timestamp,
                           runtime);}}
```

The full set of icons are animated a predefined number of times with a set of simulation analog, digital and time data Before an animation of the full set of icons (an animation cycle) in the foreground takes place, the time is acquired via the BIOS function *_bios_timeofday()* Thus the time taken for one cycle is calculated If this time is greater than or equal to a predefined constant, known as the real-time barrier, then the message '*system too slow*' is indicated via a window at the lower left corner of the screen If the time taken is less than the real-time barrier then the message '*system wait*' is indicated in this window and the program loops until the real-time barrier period has elapsed The real-time barrier in this case is 5 clock ticks, which approximates 275mSec The screen must update itself within some reasonable time with respect to the time constant of the system under supervision as seen in section 4 4 When the total number of animation cycles has finished, or the animation run is aborted by clicking one of the mouse buttons, the program takes the number of times the animation was not fast enough (number of '*system too slow*' occurrences) divides by the appropriate total number of animation cycles and thus forms a percentage of effectiveness of the screen update for the particular set of icons and real-time barrier chosen

The *Info* option of the function menu allows a particular icon to be chosen from the foreground, if icons exist in the foreground This option requires that a particular icon which has been chosen must be acknowledged by changing the mouse button status if *Info* operations are to be carried out on it Prior to acknowledgement the icon is repeatedly erased and redrawn This acknowledgement system ensures that the correct icon is chosen for *Info* operations An instance of the *template* class, a direct inheritor of the *cadshape* class, is then created, which holds particular data for the type of object which has been chosen

```
cadshape * infoets=foreground.nearest(x,y);
/*          */
template template1(infoets,x,y);
```

The template with the information is then displayed via the INFORMATION HPP function *info_display()*

info_display (template_ptr, infoets, x, y),

This function calls the *template pload()* and *template: pdraw()* methods The *pload()* method converts particular icon data into ASCII strings and loads it into its own field arrays for subsequent screen output It can access *cadshape* base class protected data inherited by icons because the *template* class is a *friend* of the *cadshape* class

Any one of the windows within the template can then be chosen via the *info_edit()* method This function in turn calls the *template pedit()* method The number of windows in the template depends on the icon type Regardless of the icon type three windows are always implemented, the icon identifier, the icon type number and the exit window A window is selected by placing the cursor on the window and clicking a mouse button Figure 5 7 shows *template* objects being used to change the status textword corresponding to the off condition for a *statustile* object, and the cadshape identifier for a *valuetile* object

The icon type field cannot be edited and the user is returned to select some other field if an edit is attempted By editing the icon identifier field the identifier, given to the icon when it was chosen from the list of icons, can be changed The number of windows or fields in a template for a particular icon type is established in TEMPLATE HPP, as is the size of windows and the resulting template size The processing of these fields is carried out by the *template* class methods in TEMPLATE.CPP The number of fields in a template for any icon class and the processing of these fields is fully expandable If a digital input or output type is chosen then its status '0' and '1' texts can be edited If an analog input type is chosen, then its conversion factors can be changed For an analog input the conversion factors are used to convert the raw data from the transducer to engineering units,

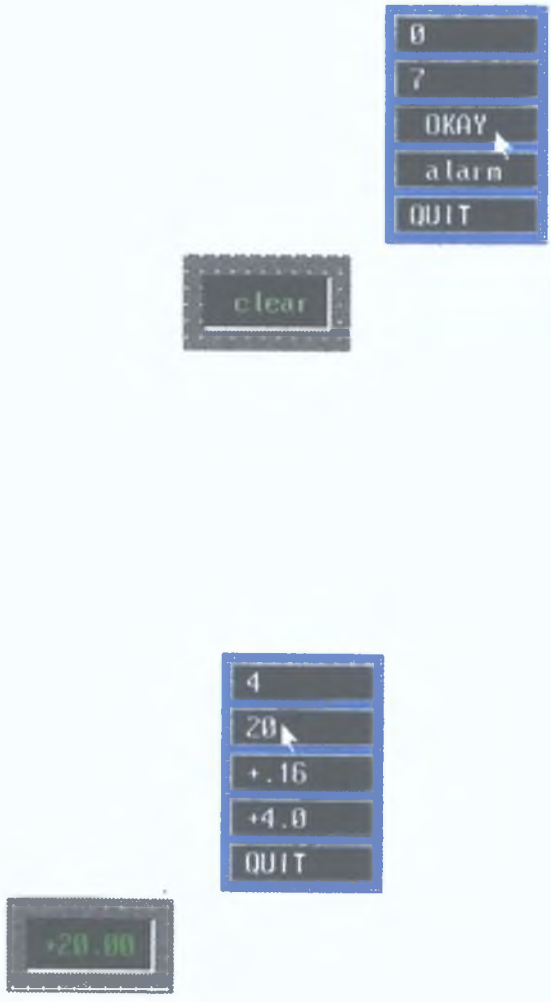


Figure 5.7 Template Objects.

whereas for an analog output type the conversion factors convert the user input, via the *A_Input* option, to a raw value to be fed out to some actuator The *pedit()* method ensures that only the correct data type is accepted for particular fields For example only digits are allowed when editing the identifier field, but all alphanumerics are allowed when editing digital status fields

Selecting the exit window causes the *template·.pstore()* and the *template perase()* method to be called, the edit loop to terminate and the *Info* option to end

The *pstore()* method takes the *template* fields, converts the ASCII strings to the correct data type for the icon members where necessary, and stores the data back to the icon The member function and call to convert a particular *template* array member to a float for storage back to the *storecad* icon with data member *conv_m* looks like

```
valuefld_s (m,&(storecad->conv_m)),
void template .valuefld_s (int vm,float* conv),
        *conv=(float)atof(&(field[vm][0])),    }
```

The *pstore* method ensures that if parameters for analog conversion are changed for a particular icon then the stored values in the icon will be recalculated on the basis of the new parameters and restored

On selection of the DOS option in the DESIGN function menu, the while statement enclosing the menu option choice is terminated Thereafter any remaining BM**** SAV constructor generated files are deleted, the Flash Graphics mouse modes are terminated and control is returned to DOS

6 The Way Forward

6 1 Introduction.

The class hierarchy developed during this project represents a subset of what would be an evolving class library to be utilized in an object-oriented workstation The design program highlights the ease with which realistic mimic diagrams could be assembled by a plant operator The simulation function within the design program acknowledges the power of realistic mimic diagrams coupled with dynamic icons

This chapter looks at the idea of **persistence** and object-oriented databases It highlights future enhancements that could be carried out on this development It points to the future of object-oriented languages and C++, and finally it focuses on the important issue of the management and organisational impact of Object-Oriented development

6 2 Object-Oriented Databases and Persistence.

In many object-oriented applications an object-oriented database will be required to store to disc instances of classes as they are generated or as their internal data is dynamically modified An object which resides on disc but which can be called to memory and written back to disc transparent to the user program is known as a **persistent** object In other words the transient in-memory objects map automatically and isomorphically to disc storage Such an object is non-volatile The user program utilizes the object as if it were in memory all the time

Object-oriented databases are still at an early stage of development A set of rules for object-oriented databases "The Object-Oriented Database System Manifesto A consensus from Academia" [ATW090] was formulated late in 1989 The role of this manifesto is not to lead a standardisation effort However it does represent a convergence of research opinion on the direction of object-oriented database management systems (OODBMS) The manifesto consists of thirteen rules, eight covering the object-orientation of a OODBMS, and the other five covering the database operations of the system An additional optional five rules are included The set of object-orientation rules deal with support for classes, for the notion of building complex objects from other objects, for the principle of separation of object interface and object implementation, for the inheritance mechanism, and for no distinction between system and user defined types The DBMS rules provide for persistence of data, concurrent users and the ability to recreate the database in some coherent state after a software or hardware, processor or disc failure A coherent state is a state in which the contents of the recovered database reflect the database status at some point before the failure "The authors are aware that the Golden Rules will not provide an easy choice between OODBMS capabilities, because the Golden Rules are likely to be supported in some fashion by every offering In contrast, the five optional features will provide more of a basis for comparing OODBMSs " [ATW090] Those optional features deal with such topics

as multiple inheritance, compile time type checking and the possibility of simultaneous access to users running on different machines

There are few products available on the market at this point in time However there is a consensus of opinion that the area of CAD is an ideal application for OODBMS "CAD/CAM applications with their complex data requirements and engineering drawings are natural candidates for this technology" [BOCH89] "The majority of the object databases are not commercial products, but proprietary systems built into CAD products by companies like Mentor Graphics " [STON90]

The first high performance, commercially available object database supporting C++ was the ONTOS system from Ontologic Inc , [ONTO89] This database can be used directly with C++ applications and is designed to support object data management between memory and disc It claims support for multi-user database access in a homogeneous network environment and runs under OS/2 Distributed access is important particularly if the object database can be modified at workstation level For future releases it claims support for clients and servers distributed across platforms of different types

The database architecture is based on the storage of objects rather than records Objects that are related are stored in such a way that they can be retrieved by a single database access Because of this and the way in which complex data structures like trees can be read in a single database access, performance is greatly improved over relational databases Caching techniques are also used to improve performance The system allows multiple users concurrent access to the database

ONTOS offers a C++ programmatic interface, an object SQL interface and, in the future, a Fourth Generation graphical programming environment

The C++ programmatic interface is of particular relevance in this case The reference facilities of this interface give the user program direct access to the entire database, containing the user-defined object instances, in a transparent manner This facility allows the user to reference persistent objects directly and transparently, without regard to whether they are on disc or in memory already ONTOS supports GET/PUT object and GET/PUT CLUSTER semantics

ONTOS claims much higher performance characteristics than relational databases This is due to the fact that data elements are stored as objects and are presented to the program in directly usable C++ object form with no conversion needed, objects can be stored on disc in a programmer defined grouping (clustered) thus allowing objects frequently used to be stored and retrieved in a single database access, and memory caching is used The particular implementation of C++ indicated in the ONTOS documentation is Glockenspiel C++

The recently released Versant Object Technology OODBMS, VERSANT, is to be ported to OS/2 and Microsoft DOS environments sometime within the end of 1991 and the middle of 1992 [VERS90]

6 3 Operating Systems.

DOS was designed as a single user, single tasking operating system OS/2 was designed for single users with multitasking capability, while UNIX provides for multi users and multitasking UNIX can now be run on 80386 and 80486 microprocessor machines OS/2 requires an 80286 or better, while DOS runs on 8088 upwards

The OS/2 option was examined in this project in order to overcome the 640Kbyte DOS barrier However it was found that the object-oriented tools required did not support OS/2 satisfactorily It was also found that, given that this project was essentially a DOS application, the availability of utilities, the ease of transfer to and compatibility of DOS overlay and extender technology was far superior to an OS/2 migration Given the requirement for a low cost implementation, the use of an overlay system under DOS requiring only 640K memory and a minimum of a PC represents an ideal solution

UNIX provides for multiple users sharing information and given its multitasking capabilities could provide a fine platform for a graphics application such as this However the financial implications of using UNIX tools must be taken into account

The Windows environment provides for a flexible user-friendly system interface Zortech C++ version 2 1 claims support for Windows versions 2 1 and 3 0 Further research could determine the level of compatibility between C++ and Windows for non-trivial systems where extensive use is made of dynamic object allocation and deallocation For an OOPS graphic project such as this the level of support for graphics provided within Windows would also be an important issue as would the compatibility of Windows and the Flash graphics library provided under Zortech C++.

6.4 Towards the Object-Oriented SCADA Workstation

It is envisaged that the set of classes provided in this project may be the starting point for SCADA projects and that new derived and new base classes may be created using these classes during the course of future projects. Above and beyond the wide range of classes supplied here, special types of graphical dynamic objects would typically be created for different applications.

A number of enhancements and additions could be made to the software developed in this project.

A real-time I/O system needs real raw data. This project may eventually form part of a development which may include a network card and associated software to transfer raw data from data collection stations via a network to the workstation and onto the dynamic objects which mimic the behaviour of the machinery and/or the transducers on site.

There is a requirement for a object-oriented database which would store the particular set of objects chosen for a mimic foreground. The coordinates for each object are part of the data of the object as are the other object features which the user can change, such as the conversion values for raw data to engineering units for analog transducer objects. It would also be a feature to be able to couple together the particular mimic background with the cluster of dynamic icons which form the mimic foreground, thereby allowing a mimic name to be called which would select both the background and the dynamic foreground to appear automatically. The concept of a distributed database is important if the user can change data relating to a particular icon. An example of this would be if a user changed the raw value to engineering units conversion factors on one workstation and it was not changed on another workstation which was in the same telemetry network. Clearly the one transducer could have different values at different points in a networked system.

It would be a very powerful enhancement if dynamic icons could be created graphically and interactively on screen by a user This would be achieved by supplying a set of primitive graphic objects which could be put together to form an icon similar to the dynamic icons given in this project Some of the primitive graphic objects would be passive, while others could be active They would all be classes An example of a passive element could be a square An example of an active element could be a square whose colour changes on receiving a particular method The new icon would also be an instance of a class Clearly such a development as seen from this project would require excellent software tools and memory management including dynamic memory associated with classes It is apparent at this point in time that C++ would not support such a development on a PC except for trivial applications However it does remain an interesting development for the future, when it could be supported by more advanced development and run-time environments and would allow users to construct their own high level dynamic icons

6.5 Object-Oriented Languages and C++

"There is no language/system in existence that can serve the range of applications/needs that is currently being served by various OOP languages" [STRO90b] There is a plethora of OOP languages available depending on the application, from Smalltalk and multi-inheritance **Eiffel** to **Flavors** for Artificial Intelligence, Turbo Pascal version 6.0 and various implementations of C++. The important issue is not how many features does each language have but does each language have the features required for object-oriented programming in the required application area.

Zortech C++ now has direct competition from Turbo C++ Professional 1.0. Turbo C++ at present does not support Windows or OS/2 giving Zortech the clear lead. Turbo C++ has an overlay system to break the 640Kbyte DOS memory barrier as does Zortech's Virtual Code Manager (VCM). It claims compliance to AT&T C++ 2.0 and uses the new AT&T iostreams library which is not available in the Zortech implementation. Both environments provide debuggers and the set provided by Zortech allowing source level debugging of extremely large programs under VCM was found invaluable during this project.

According to Spicer "Zortech has managed to avoid many of the bugs found in *cfront* (AT&T's C++-to-C translator) versions of C++ 2.0" [SPIC90]. According to King of Microsoft UK, Microsoft is developing a C++ professional development system which will be offered as an upgrade to existing C customers [KING90]. The system is to provide mechanisms to support persistent objects and object libraries for graphical environments.

C++ is now undergoing standardisation, as previously mentioned [STRO90a] is the ANSI base document for this standardisation. As regards implementations, Zortech C++ version 2.13 should be upgraded to conform to the AT&T C++ version 2.1 specification. Pointers to members are implemented using a different method to AT&T's CFRONT. However Zortech claims that its implementation is more efficient [SMIT90].

Criticisms of C++ alluded to in chapter 2 show some of the directions C++ has to move in now According to Stroustrup there is more work required on the streams "the concept of a stream is too underdeveloped I think we need a system that allowed a uniform mechanism for I/O of both user defined and built-in types There is a model, and it works well for a lot of cases and it can be developed further " [STRO90c]

Another improvement would be a virtual function check For example if in a base class has

func (int),

and a derived class has

func (char),

and it's *virtual*, then the compiler will ask is this really what is required

Parameterised types, which should make the writing of container classes easier, have been introduced informally into the language in the form of the *template* class The *template* design is experimental at this point in time Finally **Exception Handling** has been introduced, again informally, into the language as an experimental design

Up to very recently as seen in this project OOPS software tools lagged significantly behind the language An example of this was the inability to debug programs of any significant size This made development particularly difficult when extensive use was made of dynamic memory Another example was the lack of any overlay system to beat the 640Kbyte DOS barrier Even to-date as seen above there are very few object databases available

One of the most important issues is the usability of object libraries The vast majority of screen I/O object classes provided for Zortech C++ are for text mode Building object libraries for graphics environments that are well designed, reusable and portable is one of the greatest challenges facing the OOPS community today

6.6 The Impact of OOPS on Technical Management

Object-Oriented technology requires a different way of thinking about system development. This is required not only for the developers but also for their managers, as management issues will ultimately determine the success or failure of a development. The key issues include management of human resources, budgeting, project time estimation, system development methods, the OOP learning curve, and planning for class reuse.

"We believe that during the next decade, wholesale migration to object-oriented applications throughout the software market is inevitable" [BOCH89]. As seen from above it is generally agreed that there will be a massive move to object-orientation in the near future. The question is whether it is time to move to the object-oriented world. A prudent strategy might be to let others pioneer the technology and monitor their progress. However those in first are likely, considering the enormous advantages of OOPs, to gain a significant lead on their competitors. Other critical factors in the OOPS decision include whether the current system is meeting the demands of its users or not, and the status of OOP tools for the particular environment in question.

When budgeting for OOPs projects it is important that there is planning for startup costs such as training, external support expertise and software tools. When OOPS projects are being developed successfully maintenance costs are not likely to decrease, because it is likely that more systems will be produced faster. Perhaps the most important asset that a prospective OOPS developer needs is the ability to change, particularly if coming from a procedural development environment.

The close binding between real world objects and classes will inevitably lead to closer interaction between designer and user. Task assignment should be clearer due to the coding and testing modularisation that OOPS facilitates.

Training in OOPS is required for every member of a project team including managers It is important that managers develop an understanding of OOPS so that they can work knowledgeably with the project developers It is natural that faced with the problem of understanding OOPS that, at difficult points, developers may slip into procedural design This is compounded by the ability to write C and use C library calls within C++ This can be avoided by sufficient technical support on initial projects from external expertise followed by internal support consultants Project time estimates should be elongated to take into account the learning process

It typically takes several projects before the library of reusable classes begins to grow Classes may also have to be reworked Thus this is a long term process [LENZ90]

6.7 Conclusion.

This project shows how an interactive object based graphical user interface for a SCADA application may be implemented using state-of-the-art software languages and tools

It shows how mimic diagrams can be transformed from archaic character based diagrams to realistic schematics and/or plant pictures with dynamic icons It shows the power, flexibility (and weaknesses) of C++

It illustrates the object-oriented paradigm

It expounds on how object-oriented programming and graphics can be combined and the on dangers of product incompatibility

It establishes that the OOPS revolution is a reality today whose future is balanced on the availability of object libraries

It points to the future, to the enhancements that could be applied to this project's software, to changes imminent in C++ itself and to the learning and mind-set changes that are inevitable if the object-oriented paradigm is to be used to its full potential

APPENDIX 1 THE C++ LANGUAGE

A1 1 Introduction.

The purpose of this appendix is to illustrate the differences between the C++ and C programming languages, to show where C++ enhances C and to set forth the key elements that make C++ the language that it is Criticisms directed towards the C++ language and new features that have been implemented recently are also examined

A1 2 C Enhanced

The most significant difference between C and C++ is the support for object-oriented programming that C++ provides There are also many ways that C++ enhances C by the introduction of non-object-oriented features [WIEN88] A typical example is the way in which C++ compilers performs type checking to ensure that the number and type of parameters sent into a function match the number and type of formal arguments defined for the function The return type for a function is also checked to ensure it matches the variable used in the calling expression It is interesting to note also that C++ functions with the specifier *overload* can have the same name within the same scope if they are distinguishable by the number and type of its parameters The parameters for any function can have default values and hence a function may be called with fewer parameters than in its formal definition, the missing parameters assume the default value C++ presents strict parameter type checking, although this can be relaxed by using the ellipsis ( ) in a function call

While on the subject of function calls it is possible within C++ to have formal parameters of a function declared as **reference** parameters using the ampersand operator For example

```
void decrement (int& value) {value--;}
int i;
decrement (i),
```

Value is defined as a reference parameter and hence the address of the parameter *value* is assigned to the address of *i* when the function *decrement* is invoked The value of *i* that is passed is decremented within the *decrement* function and returned to the variable *i* outside the function Therefore it is not necessary for the address of *i* to be explicitly passed to the function *decrement*

Other improvements include the *new* and *delete* operators, provided to handle memory allocation, and the ability to make declarations within blocks An example of in-block declaration is given by

for (int q=5,q>0,q--){}

C++ boasts a scope-qualifier operator This operator is used to resolve scope conflicts and is also used in connection with classes For example, if an automatic variable *sum* is declared within a function and a global variable *sum* also exists, then · *sum* allows the global variable *sum* to be specified within the scope of the automatic variable The reverse is not true in that an automatic variable cannot be accessed from outside its scope

The type *void* in C++ is used to indicate that a function returns nothing A pointer variable which is declared to point to void can be assigned to any other pointer that points to an arbitrary base type

The *inline* specifier can be used to instruct the compiler to provide inline substitution of a given function at the location where the function is called Function call overhead may be saved at the expense of increased code size

As an alternative to cast conversion the name of a type can be used as a function to convert data from one type to another

The *const* specifier can be used to freeze the value of an entity within its scope For example the parameters of a function can be declared as constant to freeze the value of those parameters within the function

The object-oriented features of C++ include

Classes, structs and data **Encapsulation,**
Constructors and **Destructors,**
Private, Protected and **Public** sections
Friends,
Overloaded and **Virtual** functions,
Streams

The *class* is the basic structure in object-oriented programming The data for an abstract *object*, an instance of a class, and the associated set of routines called **methods** can be encapsulated in a class definition "Each instance of a class is an independent object with its own data (and methods) but it shares identical methods or procedures with all other objects in the same class" [KIER90] The object thus contains its own set of private and public data and methods Communication to a class object is via its methods The struct in C++ is a special case of a class with no private or protected sections, but contains not only data as in C but also has functions

Constructors and **destructors** provide guaranteed initialisation and clean-up of data and memory allocation within an object declared to be of a given class The declaration of an object calls the initialisation specified in the class constructor A destructor provides for automatic deallocation of heap associated with an object when it goes out of scope

The class has up to three sections *private, protected and public* The private section of a class contains data and/or functions and is accessible only to a classes own methods and friend classes or functions The protected data and/or methods are accessible to the class and child classes which are derived from the class The public data and/or methods of a class are available inside and outside the class It is these methods that form the communications protocol by which the object is manipulated A method to draw an

object may take the form

obj draw(1,10),

to draw the object at coordinates (1,10) The implementation details are of no interest to the user The object may be a square or circle or some other geometric shape The particular shape typically will not have been decided at compile time and will be decided at run time

The *friend* construct allows the data encapsulation of the private section of a class to be broken One or more outside functions or an entire class may be declared to be a friend of a class and thus have access to the private functions and data of a given class

A large set of existing operators can be given new meanings by the **overloading** facility The responsibility for overloading functions in an intuitive way is that of the user For example the operators + and - could be overloaded to cater for a complex number class However re-designing + to mean modulus of a complex number is obscure and should be avoided The definition of new operators is not allowed

A hierarchy of sub, derived or child classes can be established Derived classes have access to the protected and public data/methods of the parent class The derived class may have its own data and methods which the parent class does not have access to

A *virtual* function or method is a method which appears more than once within a class hierarchy with the same name but with a different implementation If the object *draw()* function above were a virtual function, then for the different geometrical shapes a different *draw()* implementation could be called thus allowing squares or circles or other shapes to be drawn This feature is called **polymorphism** This helps the software design process, because it allows a high degree of abstraction The designer need only worry about the action, not the implementation

The **stream** classes *cin*, *cout* and *cerr* are provided in C++ implementations for terminal and file I/O The operators in these classes can be overloaded for newly defined classes thus facilitating their I/O

A1.3 Fundamentals and Macros.

A **definition** allocates memory for a variable or constant The data element may or may not be explicitly initialised For example

```
double pi,
double pi=3 14159,
```

whereas

```
extern double pi,
```

is **a declaration** and *pi* must be defined elsewhere The *extern* keyword is a flag to the linker that *pi* is defined elsewhere

In the following definition the value is permanent

```
const double pi=3 14159,
```

For a variable or constant declared in a function the scope of the variable or constant extends from the point of declaration to the end of the block in which the declaration occurs For a variable or constant not declared in a function the scope extends from the declaration to the end of the file in which the declaration occurs If the same name is used for a local and a global variable ,then within the scope of a function, which defines the local variable, the local variable may mask the global variable This is called name hiding The scope resolution operator may be used to overcome this problem For example

```
int a.          // global a

func() {
        int a=2, // local a assignment
        · a=3;   // global a assignment
        }

int *ptr=&a;    // ptr points to address of global a
```

Static variables that are not explicitly initialised are implicitly initialised to zero

In C++ the length of a name, consisting of letters and digits, is not limited, however the particular compiler implementation may limit it For example in Zortech C++ the maximum name length is 127 characters Certain keywords cannot be used Examples include

class const double sizeof while virtual

Upper and lower cases are distinct, therefore *Square_width* is not the same as *square_width*

C++ has a set of fundamental types

Integers	*char*
	short int
	int
	long int
Floats	*float*
	double
unsigned	*unsigned char*
	unsigned short int
	unsigned int
	unsigned long int

Characters, integers and enumerations are called integral types Integral and floating types are called arithmetic types The actual sizes of the types depends on the hardware implementation All that can be said is that

sizeof(char)<=sizeof(short)<=sizeof(int)<=sizeof(long)

and

sizeof(float)<=sizeof(double)

Implicit type conversion can be carried out in that the fundamental types can be mixed freely in assignments and expressions, however assignment of a variable of one type to another with fewer bits is potentially a source of error Care should be taken

The declaration operators

** pointer,*
& reference,
[] vector, and
() function

can be used to derive other types from the fundamental types For example

```
int* a,          // pointer to integer a
char* arg[10],   // vector of 10 character pointers
```

The type *void* is used to specify that a function does not return a value or as a base type for pointers to objects of unknown type (at compile time) For example

```
void func(),      // func doesn't return a value
void* ptr_void,   // pointer to object of unknown type
```

A program must contain a function called *main*() which designates the start of the program This function cannot be overloaded and its type is implementation dependent

```
int main() {}
int main (int argc, char* argv[]) {}
```

Argc is the number of parameters passed to the program If it is non-zero these parameters are supplied as zero terminated strings in *argv[0]* through *argv[argc-1]*, where *argv[0]* is the name used to call the program

If, switch, while and for statements take the form.

```
char ch;
if (ch=='a'){}
```

```
else {}

switch (ch){
     case 'a'
        /*        */
     break,
     default
        /*        */
     break,
}

while (ch!='a'){}

for (int i=0,i<5,i++){}
```

A function declaration may take the form

```
int erase(cadshape*),
```

returning an integer and taking a pointer to some user defined type cadshape

The function definition may take the form

```
int erase (cadshape* cad_object){
     int all_erased,
     /*       .*/
     return all_erased,
}
```

Typically a C++ program will include many **header** files containing declarations and definitions The keyword *extern* indicates that a declaration of a name is only a declaration and that a definition exists in some other file An example of an extern declaration is given below

```
extern int look,
```

A **macro** allows a string to be replaced by a token Macros know nothing about C++ types or scope rules, they simply manipulate strings

```
#define NOT_DIGIT(io) (!isdigit(io))
```

An error in a macro will be reported when the macro is expanded not when it is defined This can lead to very obscure error messages

For a type cadshape

```
cadshape*          // is a pointer to cadshape and
                   // holds the address of cadshape
```

Other pointer examples include

```
char** ch,         // pointer to pointer to a character
float (*v)[5],     // pointer to a vector of 5 ints

// pointer to a function taking a pointer to a character, a
// float and returning an integer
int (*func)(char*,float)
```

An **array** is an aggregate of elements of the same type, whereas a *struct* may be an aggregate of elements of different types It is not possible to compute the size of an object of a structure type simply as the sum of its members, because some machines require certain types to be allocated on architecture dependent boundaries For example an integer must typically be allocated on a word boundary

A reference is an alternative name for an object For example

```
int b=1;
int& a=b, // a and b now refer to the same thing
```

A **reference** always refers to the object it is initialised to denote, and it cannot be changed after its initialised To get a pointer to an object referenced one can write for example

```
int* a_ptr=&a;
```

On many hardware architectures an object can be accessed faster if placed in a *register* For example

```
register int a,
```

Register declarations should only be used when efficiency is paramount It is not possible to take the address of a register object nor can it be global

The keyword *const* can be used to make an object a constant rather than a variable Since a constant cannot be assigned to, it must be initialised There is an alternative to declaring type *const* For example

```
const DRAW  = 0, DRAW = 0,
const ERASE = 1,
const END   = 2,
```

could be written

```
enum choice {DRAW, ERASE, END},
```

where the name of the *enumeration* is a distinct integral type Values can also be explicitly given to enumerators Note that the type for *DRAW*, when declaring as *const* above, is not given If the type is missing the default is taken as *int*

A *struct* can be used with fields to put more than one small object into a byte For example

```
struct xreg {
unsigned enable   1;    // 1 bit
unsigned page     3,    // 3 bits (not used)
unsigned          1,    // 1 bit padding
unsigned mode     2;    // 2 bits
unsigned          1,    // 1 bit (not used)
};
```

This may not save space as the code to manipulate the struct fields may be larger than if a char were used for each object

The objects in a *union* use the same space at different times All the members of a union only take up the space of the largest member of the union The members of a union have the same address For example

```
union un{
        char *ch,
        int a:
};
```

A *union* can be named and thus becomes a type in its own right

A full list of operators for C++ is given in [STRO87]

A14 Operator and Function Overloading.

Most operators can be **overloaded** An overloaded operator may be declared to accept class objects as operands Overloading allows an operator to be redesigned as a method within a class or as a function taking at least one argument of a class or a reference to a class The precedence of operators cannot be changed, nor can the meaning of operators applied to non-class objects or the number of operands of operators Relationships between operators applied to basic types need not hold for operators applied to class types For example

--a ≡ a-=1, is not necessarily true for class types

The class member access operator ., scope resolution operator , conditional operator ? , and preprocessor symbol # cannot be overloaded This is because the class operators above have a predefined meaning for all class types This makes C++ extensible, at the same time protecting the programmer from obvious abuses of operator overloading such as the redefinition of + on floats to mean subtraction

A typical example is overloading operators for complex numbers

```
// Class complex file  COMPLEX HPP
   class complex {
         float re,im;
   public
        complex (float r, float i) {
               re=r, im=i,
               }
        // Redefine + and - as methods of this class
        complex operator+ (complex,complex);  // Definition in
                                              //   cpp file
        complex operator- (complex,complex);  // Definition in
                                              //   cpp file
        };
```

```
complex a = complex (1,2),
complex b = complex (3,4),
complex c = complex (5,6),

a = b + b;        // same as   a = operator+(b,c),
a = b - c,        // same as   a = operator-(b,c),
```

It is not possible to define new operator tokens Function call notation should be used when the standard operator set is not adequate

A number of classes have been designed for **stream** I/O in C++ and although they do not form part of the language they are strongly bound to it These classes provide for I/O of basic and user defined types The standard input stream cin, standard output stream cout and standard error stream cerr are provided by the stream library STREAM HPP For example

```
#include <stream hpp>
main()
     {
     long l,
     double d;
     cout <<"Streams",
     cout << "l=" << l,
     cout << "double=" << d << "\n",
     }
```

where the overloading of the left shift operator is given in the class *ostream* For example

```
class ostream {
    /*          */
    public
```

```
        ostream& operator<<(char*),
        ostream& operator<<(long),
        ostream& operator<<(double),
        ostream& operator<<(      ),
      /*          */
  },
```

Because the operator<< functions return an *ostream* reference the left shifts can be stacked as above

```
cout <<"1=" << 1,       is interpreted as
(cout operator<<("1=") operator<<1),
```

A similar mechanism exists for cin and the class *istream* with overloading of the *right shift* operator >>

A function is overloaded when there are several different declarations for the function The compiler chooses the relevant function by matching the number and type of call arguments with that of the formal arguments Each function that is overloaded must be uniquely and unambiguously distinguishable An overload declaration may precede declarations for an overloaded function, except when the overloaded function is a member or operator function For example

```
overload abs,
int abs (int);
double abs (double);

abs (10),      // calls int abs (int);
abs (10 0),    // calls double abs (double);
```

and

```
class complex {
    double re,im;

    public:
```

```
        // Overloaded constructors
        complex(),
        complex (int),
        complex (int,int),
        complex (double),
        complex (double r,double i) {re=r, im=i,}

        double real (complex& z) {return z.re;}
        double imag (complex& z) {return z.im;}
};
```

Since the complex class has constructors it has a public copy constructor even though none is explicitly declared

```
complex A,          // Using complex()

complex A(10),      // Using complex (int)

complex A = 10 0,   // Construct complex (10 0) using
                    // complex (double) and copy it into A

complex A = complex (1,1),  // Construct complex (1,1) using
                            // complex (int,int) and
                            // copy it into A
```

The left shift operator can be defined for a user defined type such as this complex class as follows

```
ostream& operator<<(ostream& s, complex z)
  {
  return s << "[" << real(z) <<"," << imag(z) <<"]",
  }
```

and used simply as.

```
complex A(1 0,10 0),
cout<< "A = " << A;    to produce    A = [1 0,10 0];
```

A15 C Compatibility, New Features and Criticism.

C++ is an evolving language, into which new features have been introduced since the publication of [STRO87] Although a particular implementation of a C++ compiler does not constitute a language definition, it is common for the features to be referred to according to the release of the AT&T C++ translator in which the feature was first introduced Until such time as a standard like the ANSI C standard becomes available for C++, the various features will continue to be measured against AT&T compilers [STRO90a] has been selected by ANSI as a starting point for the formal standardisation of the C++ language

"C++ is (very nearly) a simple superset of the C language" [SCOT90] It is based on C [KERN88] and has adopted most of the changes given by the ANSI C standard A few noticeable differences may prevent an ANSI C program from being compiled with a C++ compiler

The additional C++ keywords cannot be used as C identifiers, examples include

asm class delete friend virtual protected public

In C++ a function must be declared before it is called Whereas ANSI C provides an automatic declaration of all undeclared functions as a function returning an int and with a variable number of arguments For example

int f( ),

The function *f()* in C++ means that the function f takes no arguments, hence in C++ it is seen as *f(void)* Whereas in C it means it can take any number of arguments of any type However this use is deemed obsolescent in ANSI C

In ANSI C a *void** may be used as the right-hand operand of an assignment to or initialisation of any pointer type, in C++ it may not.

If a variable of type *const* is declared outside a function in C++ it has static linkage, whereas it has extern linkage in C To force external linkage of const variables in C++ the *extern* keyword must be used in the declaration

The type of a character constant is char in C++ and int in C
For example

```
sizeof ('a'),  // Equals sizeof(int) in C
sizeof ('a'),  // Equals sizeof(char) in C++
```

In assignment of a char to an int For example

```
char ch = int i,
```

The most significant bits are lost

In ANSI C an external name may be defined several times whereas in C++ is must be defined exactly once

[HANS90] gives some indications as the requirements of a compiler implementation which may ease the use of C programs as C++ programs

In C++ a structure name declared in an inner scope can hide an outer scope object, function, enumeration or type For example

```
int A[10],
void func() {
    struct A  {/*        */};
    sizeof(A),  // size of integer array in C
                // size of struct in C++
    }
```

"The primary driving force in the evolution of C++ between 1985 and 1990 has been the desire to produce elegant and efficient libraries and to allow easy and safe composition of programs out of separately developed and separately compiled parts " [STRO90a]

The keyword *template* was reserved for future use [HANS90] It has just been introduced by [STRO90a] as an experimental design The template class provides the layout and operations for an unbounded set of related types It provides a way of providing general container types such as *list*, and *array* where the specific type of elements is left as a parameter For example a List class might provide a common definition for a list of ints, a list of floats, or a list of cadshape classes A single template function Sort could provide a common definition for sorting all types defined by the List class template

The **exception handling** design proposed in [STRO90a] is experimental It does not as yet form part of the C++ version 2 0 reference manual

Multiple inheritance introduced in 1989 allows a class to inherit data and/or methods from more than one base class For example

```
class W {/*        */}
class X {/*        */}
class Y {/*        */}
class Z   public W, public X, public Y {/*        */}
```

Protected class members were introduced in 1986 A protected class member can be used only by member functions and friends of the class in which it is declared, and by member functions and friends of any class derived from the declaring class
At the same time pointer to member functions was introduced For example

```
class X {
   public.
     int mem(char*),}
```

The declaration for a pointer to function within X taking a *char** argument and returning an *int*

```
int (X··pmem)(char*);
```

The assignment of the address of X. *mem()* to this pointer

```
pmem = &X  mem;
```

Now the member can be indirectly invoked, if Y is an object of class X

```
int p = (Y *pmem) ("string"),
```

or directly invoked by

```
int p = Y mem("string"),
```

In 1989 the use of *this* for control of space in constructors and destructors was replaced by the capability of user defined operators *new* and *delete* for classes The use of *this* was considered very error prone and is to be phased out of the language When *X operator new* and *X operator delete* are defined for a class X, then all calls to *new* and *delete* for that type will invoke the defined version of *new* and *delete* instead of the global versions

Objects can now be explicitly destroyed This has been expanded on in the section above dealing with constructors and destructors

The keyword *overload* is now redundant but may still be used. The class member operator ( ) and class member pointer operator (->) may now be overloaded The restriction on having two versions of an overloaded function differing only in a parameter which is unsigned in one version and signed in the other was lifted in 1987 Similarly, following the ANSI C change that a *float* parameter can now be passed on the stack without being automatically promoted to a *double*, it is now possible to declare two versions of an overloaded function differing in that one has a parameter as a *float* and the other as a *double* Likewise the restriction on overloading *char* and *short* parameters was lifted in 1989

Following the lead of ANSI C, constants of type *float* may now be declared by placing the letter *f* after the number The number must have a decimal point For example 1 7*f*
Also the type *long double* has been introduced, and may be declared

for a by placing the letter *l* or *L* after the number which must include a decimal point

The size of a *long double* is guaranteed to be at least as long as a double Its size depends on the hardware types available on the particular machine

The **abstract** class idea was also recently introduced An abstract class is a class that can be used as a base class for other classes only An example of this would be a general *shape* class, from which *square*, *line* and *circle* classes are derived Virtual functions declared but not defined in the shape class, for example *draw()* and *erase()* would be defined in the derived classes Defining a variable of type *shape* does not make sense since nothing can be done with it It would be better if the compiler forbade the declaration of a type *shape* This is possible by giving null definitions to at least one of *shape's* virtual functions Such a virtual function with null definition is known as a pure virtual function An abstract class is one which has at least one pure virtual function The compiler now prevents any variable of type *shape* from being declared A mixture of pure virtual functions and error definitions could be used All classes derived from an abstract class are required to provide a definition for the pure virtual functions or redeclare the functions as pure virtual functions The cadshape class in this project is very similar in its construction to the shape class discussed above An example of an abstract class and a class derived from an abstract class is given below

```
class shape {
    unsigned x,y,
    /*        */
public:
   // Declaration and definition
   void move (unsigned x_coord,unsigned y_coord){
        x = x_coord; y = y_coord;
        draw(),}
```

```
    // Pure virtual declarations
    virtual void draw(unsigned,unsigned) = 0,
    virtual void erase() =  0,
   /*       */
},

class circle   public shape {
    int radius,
public
    void draw() {
        /*        */
    }
    void erase() {
        /*        */
     }
},
```

If no definition is given for either of the pure virtual functions of the parent class then the derived or child class inherits the respective pure virtual function from the base or parent class and becomes itself an abstract class Alternatively the pure virtual class may be redeclared as pure virtual in the derived class and the derived class becomes an abstract class If all the pure virtual functions are declared normally (not pure) in the derived class then the derived class is not abstract

If a class *Y* is derived from a class *X* write

```
class Y   X  {    },
```

then by default the base class *X* is private to *Y* and not accessible to users of the derived class *Y* However in future versions of C++ it may be required to make *X* explicitly private to *Y* by using

```
class Y . private X {    };
```

On the other hand, to allow the base class public status then use

```
class Y : public X {   .};
```

This section has given examples of some of the important recent updates and modifications to the C++ language For examples of further changes the reader is directed to [HANS90]

Criticisms relating to the current lack of object libraries and object interface/implementation dependency are discussed in detail in chapter 2

A number of interesting points were discussed at a question and answers session attended by leading object-oriented experts and Stroustrup [STRO90c]

On mixing of C++ data structures and C library functions it was made clear that although one of the primary goals of C++ was to retain its compatibility with C, data types cannot be passed across the interface expecting to get C++ protection C++ guarantees cannot be squeezed out of C

Other questions revolved around the use of operator overloading According to Stroustrup it was better to provide operator overloading rather than the capability of creating new operators Operator overloading should only be used when there is a conventional use for it, otherwise program obscurity can be increased quite dramatically The level of obscurity becomes critical if some new operator is introduced with a totally obscure name Hence operator overloading was introduced and defining new operators was not In view of this philosophy of only overloading operators intuitively one might ask how did the overloading of the left shift operator to *stream* operator come about ? [STRO87]

A MAKE utility typically accompanies a C++ implementation This utility automates program maintenance Instead of explicitly typing in commands to compile and link programs in a project, it is possible to write all of the steps required in a text file called a makefile Date and time stamping is used so that only updated files are recompiled. A problem regarding the makefile was reported

" when you start writing large C++ systems, the makefile is more complicated than the program, and is more difficult to write " During the development of this project the makefile became so large that eventually it was more of a hindrance than a help A dependency analyser that can hook up an incremental compiler so that the cascading of recompilation does not have to take place is on the way, but will not be ready in the near future according to Stroustrup [STR090c]

In response to a query relating to formatting functions and I/O Stroustrup stated that the tight focusing of I/O on built-in types as opposed to a uniform mechanism for I/O of both user defined and built-in types with good formatting functions, is just immaturity and could be developed further

Appendix 2. Compilation, Link and Debug Details.

A2.1 Compilation, Linking and Memory Management Systems.

The programs were edited using the ZORTECH editor ZED COM [ZED89] and from within the ZORTECH C++ Workbench ZWB EXE version 1 03 [ZWB90b] and version 1 12 [ZWB90a]

The programs were compiled with the Zortech C++ Compiler version 2 1 ZTC COM [ZTC90b] The full compilation was carried out via the batch file ZOR2COMH BAT The make facility was not used due to the complexity of the make file resulting from the large number of programs An example from the compilation batch file is shown below

```
ztc   -c  -mV -R -o+space -b -C   animvol cpp
ztc   -c  -mV -R -o+space -b -C   animplot cpp
```

The flags may be seen from the compiler manual [ZORC90] to be as follows

-c	Compile only,
-mV	Select VCM mode memory model,
-R	Place switch tables in code rather than data segment, this is recommended for VCM [ZORC90]
-O+space	Optimise for Space,
-b	Compile large program, and
-C	Prevent *inline* function expansion to aid debugging

The linker used is BLINK COM version 4 06 [ZBLI90] This is called from the compilation batch file and takes a linker response file as an argument

```
BLINK @zor2h rsp
```

The linker response file gives the names of all the compiled object files for linkage with a + between them, the name for the linked

executable and any libraries used Since Zortech C++ embeds the standard library names into the object files, it is not required to specify the names of the libraries explicitly The libraries used for the large memory model are the standard C library *zll lib* and the C++ library *pll lib* The flash graphics library *fg lib* is specified in the linker response file and the linker is given the flags as specified below

/PAC/F Required for VCM to convert certain types of function calls as described in the Zortech V2 1 update guide [ZORC90]

The VCM system requires that each separate Virtual Code Segment be contained within parenthesis in the linker response file An example is given below

(design obj)+(msmenu obj)+(fgnd obj)+(shapelst.obj)+
(square obj)+(animsqr obj)+(animpie obj)+(pie obj)+

The two methods generally used to generate large DOS applications are Virtual Code Management (VCM) and DOS Extender Technology Virtual code management is essentially an overlay system which swaps virtual code segments in and out of memory from the disc executable file as they are required It does not use extended memory or 80286 or 80386 microprocessor technology Hence an application developed on the basis of VCM could run on a 8086 microprocessor based machine At link time the modules which are required to reside in the same virtual code segments are indicated as shown above The amount of thrashing depends on the number of virtual code segments that are defined and on the mix of modules within each virtual code segment Fine tuning of the application's performance can be achieved by the selection of modules within the virtual code segments and the number of segments chosen

The Rational DOS Extender Technology requires the use of extended memory and a 80286 or 80386 target machine for the application with the Rational technology installed For this reason the VCM system was chosen because of its target system advantage

Subsequently the Phar Lap DOS Extender system [PHAR89] was tested with the Zortech C++ version 2 18 compiler for 80386 microprocessor applications [ZTC90a] Due to incompatibilities in the technology, as discussed in section 3 11, the VCM implementation was retained

It was found that the *farcalloc()* memory allocation function does not work with the Zortech VCM system This problem was isolated down to a minimal program, ALLOCATI EXE This problem was consistent with problems encountered with the same function in some other Zortech versions The Zortech VCM system only claims compatibility with the *malloc()* function and hence all *farcalloc()* calls were replaced by *malloc()* function calls Furthermore it was found that, within this application, the VCM system was not capable of managing memory dynamically allocated from within class objects Hence the dynamic buffers used to store the background behind icons were replaced by disc files This was carried out for CADSHAPE type objects The *window* and *windowtile* type objects still use dynamic buffers, but as the number of such objects constructed and destructed during this application is small the memory loss resulting from memory mismanagement by the VCM system is negligible Moreover these object's draw and erase member functions are called with high frequency and the corresponding file accesses, if a file buffer system was used, would thus consume a lot of time

If it is required to draw a large number of objects on the screen, and these objects all use files to store the background over which they will reside, then there will be very heavy disc access and the application will slow down considerably. This problem is overcome by installing a RAM or Virtual disc on the machine This is established by using the *DEVICE* command and the VDISK SYS file in the system CONFIG SYS file The result is that the thrashing effect disappears and the operation of the application is quite smooth without any noticeable VCM originated delays

Tuning the performance of a VCM application requires that programs or objects that are heavily interdependent be in the same virtual

code segment However in order that enough memory exists to call DOS utilities from the application it was found that it was necessary to make the VCM segments a small as possible The VCM system does not manage memory for DOS utilities spawned or called by the application via the *spawn()*, *system()* or *exec()* functions The DOS system and VCM system do not mesh in that if there is not enough memory to run a DOS system call from the application program then the VCM system does not unload segments to accommodate this call and even if forced to do so in a debugging session, via the *vcm_maxres* variable, then the DOS system still does not know that this has occurred and cannot use the newly released memory The result is that the DOS call fails This was seen on the call of the DEL command and other DOS utilities, such as PKUNZIP EXE [PKWA89], via the *system()* and *spawn()* functions within the application Hence the virtual code segments are kept small and this alleviates the problem to a large extent

The performance of an application under VCM can be evaluated by only allowing one VCM segment in memory at any particular time, and by subsequently monitoring the disc access frequency This is achieved by defining the variable *unsigned __vcm_maxres* = 1, in one of the VCM segments

A2.2 Debugging.

The standard Zortech C++ debugger ZDB COM [ZDB90] requires approximately 250Kbytes of memory and therefore could not fit in memory while the DESIGN.EXE program was also resident Hence the virtual debugger ZDBV86 COM [ZDBV90] was used. The virtual debugger calls the standard mapper ZMAP EXE [ZMAP90], but this could not cope with the number of symbols To overcome this the DOS Extended mapper ZMAP286 EXE [ZMP290] was used by first invoking the DOS Extended debugger ZDB286 COM [ZDB290] which in turn calls ZMAP286 Subsequently the virtual debugger ZDBV86 COM or the DOS Extended debugger ZDB286 com can be used

The extended and virtual debuggers have the facility to show dynamic buffer memory allocation and deallocation Hence memory allocation and deallocation, and object construction and destruction can be monitored This is a very powerful feature when compared to the debugging facilities available for a large time during this project when the debugger could not fit in memory with the program under test or when dynamic allocation could not be monitored

It was noted that on running certain code, which ran perfectly without the debugger, that when single or functioned stepped in the debugger, the debugger ZDBV86 failed with *'fatal internal error'* messages In accordance with the Zortech Update Guide [ZORC90] such occurrences were seen as debugger implementation errors

All programs for debugging must be compiled as detailed in section A2 1 above, with the additional *-g* flag specified on the ZTC command line Additionally the Flash Graphics library FGDEBUG.LIB may be linked into the application in place of FG.LIB. Debugging tests are performed on the function calls in this library

The VCM system is fully supported by the virtual debugger ZDBV86 COM [ZDBV90] and the extended debugger ZDB286 COM [ZDB290]

While debugging, a window is provided which shows the segments, which segments are loaded or not, the number of loads for each segment and the number of times the each segment is used Fine tuning of the VCM system can be performed by comparing the number of loads of a particular segment to the number of times the segment in question is used A VCM type system imposes a time overhead on debugging due to the loading and unloading of segments

The Extended debugger uses extended memory and requires a 80286 microprocessor based machine, while the virtual debugger uses the facility of the 80386 microprocessor to set up a virtual 8086 environment in which to run the program being debugged Both of these debuggers use a minimum of standard memory and make extensive use of extended memory Hence a 80386 microprocessor based machine is required to debug the programs in this project However, because of the VCM system the target system need only be a 8086 based machine

APPENDIX 3 BIBLIOGRAPHY

[ATTS88] AT&T OVERVIEW Scanner, Model No 640 HGS
AT&T, Audiographic Communications Systems Service,
Room 3E-123,
185 Monmouth Parkway,
West Long Branch, New Jersey 07764, USA.

[ATWO90] The Object-Oriented Database System Manifesto
A consensus from Academia"
T Atwood
HOTLINE on Object-Oriented Technology Vol 1, No 3
January 1990

[BOCH89] OBJECT-ORIENTED CELLS BRING NEW LIFE TO DBMS
B Bochenski
Software Magazine (International Edition), June 1990

[BOOC86] Object-Oriented Development
G Booch
IEEE Transactions on Software engineering, Vol SE-12,
No 2, February 1986

[BROK87] No Silver Bullet Essence and Accidents of Software
Engineering
F Brooks
IEEE Computer, April 1987

[CFXX88] CFXX EXE AT&T Translator, release 1 2 Msoft 1 (beta 2)
February 1988, Copyright 1984 AT&T, Inc and
Glockenspiel Limited, Lr Dominick Street,
Dublin, Ireland

[CHIR87] Unix for the IBM PC An Introduction
P M Chirlian, Stevens Institute of Technology
Merrill Publishing Co , 1987
0-675-20785-1

[CIEX89] Control and Instrumentation Exhibition 1989
National Exhibition Centre Birmingham

[CODE90] Microsoft CodeView Version 3 0, 1990
Microsoft Corporation,
16011 NE 36th Way,
Box 97017,
Redmond, WA 98073-9717, USA

[CODE89] Introducing CodeView
A Denning
EXE Magazine, Vol 3, Issue 8, February 1989
EXE Magazine, Vol 3, Issue 9, March 1989

[CODE87] Microsoft CodeView Version 2 10, 1986,1987
Microsoft Corporation,
16011 NE 36th Way,
Box 97017,
Redmond, WA 98073-9717, USA

[COX90] There Is a Silver Bullet
B J Cox
BYTE, October 1990

[DANI90] OOP is more than using C++
J Daniels
EXE Magazine, Vol 5 Issue 3, August 1990

[DATA89] Datac LS900/VME Central System Project
Datac Control Ltd ,
IDA Centre, Pearse St ,
Dublin 2

[DEIN90] Making Computer Behavior Consistent
The OSF/Motif Graphical User Interface
A O Deininger, C V Fernandez
Hewlett-Packard Journal, June 1990

[DOS88] International Business Machines Corporation Disk
Operating System, Version 3 x
IBM United Kingdom,
International products Limited,
PO Box 41, North Harbour,
Portsmouth, PO6 3AU, England

[DRHA87] Dr Halo Version 2 26a
Media Cybernetics,
8484 Georgia Ave ,
Silver Spring, MD 20910, USA

[EXE89] The PCX and PCC file formats
EXE Magazine, Vol 3, Issue 11, May 1989

[FAST89] FASTLINK EXE Phar Lap DOS Extender system
386|Link Linker,
386 DOS-Extender Software Development Kit Version 2 2d
1989
Phar Lap Software, Inc
60 Aberdeen Ave ,
Cambridge, MA 02138, USA

[FIED89] Object-Oriented Unit Testing
S P Fiedler
Hewlett-Packard Journal, April 1989

[FLEX88] FlexOS 286 Operating system
Digital Research Inc
Box DRI
Monterey, CA 93942 USA

[FOLE84] Fundamentals of interactive Computer Graphics
J D Foley, A Van Dam
Addison-Wesley Publishing Company 1984
0-201-14468-9

[FORD90] Object-oriented programming will it work for real time systems ?
W R Ford
Computer Design March 1990

[GERO90] Personal computers in control applications
M Gerow, R Henley, Action Instruments
Control & Instrumentation, February 1989

[GLOC90] Glockenspiel C++ version 2 Oc compiler , 1990
Glockenspiel Limited, Lr Dominick Street,
Dublin, Ireland

[GLOC89] Glockenspiel C++ Programming System 1 2E, 1989
Glockenspiel C++ SDK release 1 2E with CommonView
Applications Framework for Presentation Systems
Glockenspiel Limited, Lr Dominick Street,
Dublin, Ireland

[GLOC88] Advantage C++ (tm) Programming System, release 1 2
Msoft 1 (beta 2) February 1988
Glockenspiel Ltd , Dublin, Ireland
Published by Lifeboat Associates, Inc , Tarrytown, NY, USA

[GLOC87] Advantage C++ (tm) Programming System, version 1 1, 1987
Glockenspiel Limited, Lr Dominick Street,
Dublin, Ireland

[GCPP88] GCPP EXE Glockenspiel System V 3 compatible C pre-processor, release 6 (beta 1), February 1988
Glockenspiel Limited, Lr Dominick Street,
Dublin, Ireland

[GRAB87] GRAB EXE The Halo Frame grabber
Copyright (c) Media Cybernetics Inc 1986,1987
Media Cybernetics,
8484 Georgia Ave ,
Silver Spring, MD 20910, USA

[HALO88] HALO'88 version 1 00 04, and Device drivers Version 1 00 13, 1988
Media Cybernetics,
8484 Georgia Ave ,
Silver Spring, MD 20910, USA

[HALO87] HALO Graphics Kernel and Device Drivers
Version 2 26a, May 1987
Media Cybernetics,
8484 Georgia Ave ,
Silver Spring, MD 20910, USA

[HANS90] The C++ Answer Book
T Hansen
Addison Wesley Publishing Company, 1990
0-201-11497-6

[HOPG83] Introduction to the Graphical Kernel System (GKS)
A P I C studies in Data Processing No 19
F R A Hopgood, D A Duce, J R Gallop, D C Sutcliffe
Computing Division, Rutherford Appleton Laboratory,
Didcot, UK
Academic Press, Inc 1983
0-12-355570-1

[JONE91] C++ or C9X?
D Jones
EXE Magazine, Vol 5, Issue 8, February 1991

[KERN88] The C Programming Language
B W Kernighan, D M Ritchie
Prentice Hall, 1978 & 1988

[KIER90] Object-Oriented Programming Systems - an Overview
P Kiernan
Irish Computer, July 1990

[KING90] Microsoft Speaks!
A King
EXE Magazine, Vol 5, Issue 6,
November 1990

[KOEN90] Looking Around
A Koenig
Journal of Object Oriented Programming Vol 2, No 5,
January 1990

[KURT89] An Object-Oriented Methodology for Systems Analysis and Specification
B D Kurtz, D Ho, T A Wall
Hewlett-Packard Journal, April 1989

[LENZ90] Adopting Object-Oriented Techniques Management and Organizational Impacts
M A Lenzi
HOTLINE on Object-Oriented Technology Vol 1, No 3,
January 1990

[LILL89] Keeping Control of Telemetry
J Lilley
Communications Engineering International April 1989

[LINK89] 386LINK EXE Phar Lap DOS Extender system
386|Link Linker,
386 DOS-Extender Software Development Kit Version 2.2d
1989
Phar Lap Software, Inc
60 Aberdeen Ave , Cambridge, MA 02138, USA

[MARS87] Computer Graphics in Application
G R Marshall
Prentice-hall International, Inc 1987
0-8359-0856-9

[META89] MetaWindows/Plus version 3 4b, 1989
Metagraphics Software Corporation
4575 Scotts Valley Drive
PO Box 66779
Scotts Valley, CA 95066, USA

[MICR88] Microsoft C Optimising Compiler Version 5 10, 1988
Microsoft Corporation,
16011 NE 36th Way,
Box 97017,
Redmond, WA 98073-9717, USA

[MICR87] Microsoft C Optimising Compiler Version 5 00, 1987
Microsoft Corporation,
16011 NE 36th Way,
Box 97017,
Redmond, WA 98073-9717, USA

[MSDG89] Waite Group's MSDOS Developer's Guide 2nd edition, 1989
Howard W Sams & Company
0-672-22630-8SSRA

[MSPM89] Microsoft OS/2 Presentation Manager Softset Version 1 1
Microsoft Corporation,
16011 NE 36th Way,
Box 97017,
Redmond, WA 98073-9717, USA.

[OCON89] Computer Graphics - An Object Oriented Approach
P O'Connell
M Sc Thesis 1989
School of Computer Applications
Dublin City University

[OMF88] Object Module Formats
Appendix C
386|LINK Reference Manual 2nd Edition June 1988
386 DOS-Extender Software Development Kit Version 2 2d
1989
Phar Lap Software, Inc
60 Aberdeen Ave ,
Cambridge, MA 02138, USA

[ONTO89] ONTOS
Ontologic Inc
Three Burlington Woods
Burlington MA 01803, USA

[OS288] IBM Operating System/2 Extended Edition Version 1 1E
IBM United Kingdom,
International products Limited,
PO Box 41, North Harbour,
Portsmouth, PO6 3AU, England

[PCPB88] PC-Paintbrush+ Version 1 54, 1988
ZSoft Corporation,
1950 Spectrum Circle,
Suite A - 495,
Marietts, GA 30067, USA

[PERR90] Adequate Testing and Object-Oriented Programming
D E Perry, G E Kaiser
Journal of Object-Oriented Programming
January/February 1990

[PETZ88] Programming Windows
Petzold
Microsoft Press

[PFOR87] PFORCE++ Libraries version 1 04, 1987
Phoenix Technologies Ltd ,
320 Norwood Park South,
Norwood, MA 02062, USA

[PHAR89] 386 DOS-Extender Software Development Kit Version 2 2d
1989
Phar Lap Software, Inc
60 Aberdeen Ave ,
Cambridge, MA 02138, USA

[PKWA89] PKUNZIP EXE FAST ! Extract Utility Version 1 01,
July 1989
PKWARE Inc ,
7545 N Port Washington Rd,
Glendale, WI 53217, USA

[PMAN89] Microsoft OS/2 Presentation Manager Softset Version
1 1, 1989
Microsoft Corporation,
16011 NE 36th Way,
Box 97017, Redmond, WA 98073-9717, USA

[PORT87] A Graphics Toolbox for Turbo C - Part 2
K Porter
Dr Dobb's Journal, December 1987

[RDOS90] Rational Systems DOS Extender DOS/16M
Rational Systems Inc ,
P O Box 480,
Natick, Massachusetts 01760, USA

[QNX88] QNX Operating System
Quantum Software Systems Ltd ,
175 Terrence Matthews Crescent,
Kanata, Ontario, Canada, K2M 1W8

[RUN389] RUN386 EXE 386|DOS Extender, 1989
386 DOS-Extender Software Development Kit Version 2 2d 1989
Phar Lap Software, Inc
60 Aberdeen Ave ,
Cambridge, MA 02138, USA

[SCAN88] AT&T SCANWARE
AT&T, Audiographic Communications Systems Service
Room 3E-123,
185 Monmouth Parkway,
West Long Branch, New Jersey 07764, USA

[SCOT90] Rolling Your Own Arithmetics in C++
M Scott
DCU School of Computer Applications Working Paper CA-2090
Dublin City University

[SHOW90] An Object-Based User Interface for the HP NewWave Environment
Hewlett-Packard Journal, August 1989
P S Showman

[SMIT90] The Key to Success
P Smith
EXE Magazine Vol 4, Issue 9, March 1990

[SPIC90] C++, Plus
S Spicer
BYTE July 1990

[STON90] Database Wars Revisited
C M Stone, D Hentchel
BYTE, October 1990

[STRO90a] The Annotated C++ reference Manual
M A Ellis, B Stroustrup
AT&T Bell Telephone Laboratories, Incorporated
Addison-Wesley Publishing Company, 1990
0-201-51459-1

[STRO90b] On Language Wars
B Stroustrup
HOTLINE on Object Oriented Technology Vol 1, No.3 , January 1990

[STRO90c] Around the Table with Bjarne Stroustrup
.EXE Magazine, Vol 5, Issue 6, November 1990.

[STRO88] What is Object-Oriented programming ?
B Stroustrup AT&T Bell Laboratories
IEEE Software May 1988

[STRO87] The C++ Programming language
B Stroustrup AT&T Bell Laboratories
Addison Wesley Publishing Company, 1987
0-201-12078-X

[TAZE90] Object Lessons
J M Tazelaar
BYTE October 1990

[TINH88] PC graphics are key to low cost automation
B Tinham
Control & Instrumentation, February 1988

[VERS90] Versant Object Technology
4500 Bohannon Drive,
Menlo Park, CA 94025, USA

[WIEN88] An introduction to Object-Oriented Programming and C++
R S Wiener, L J Pinson
Addison Wesley Publishing Company, 1988
0-201-15413-7

[WILS90] The Object-oriented approach
S Wilson Praxis Electronic Design
IEE Review, July/August 1990.

[WIND87] Microsoft WINDOWS/386 Version 2 03
Microsoft Corporation,
16011 NE 36th Way,
Box 97017,
Redmond, WA 98073-9717, USA

[WIND86] Microsoft WINDOWS Version 1 03
Microsoft Corporation,
16011 NE 36th Way,
Box 97017,
Redmond, WA 98073-9717, USA

[WSDK87] Microsoft Windows Software Development Kit (SDK)
Version 2 0
Microsoft Corporation,
16011 NE 36th Way,
Box 97017,
Redmond, WA 98073-9717, USA

[ZBLI90] BLINK COM Version 4 06, 1990
Zortech Incorporated, 1165 Massachusetts, Arlington, MA 02174, USA

[ZDB90] ZDB COM Zortech C++ debugger Version 2 10A, 1990
Zortech Incorporated, 1165 Massachusetts, Arlington, MA 02174, USA

[ZDB290] ZDB286 COM DOS Extended debugger Version 2.10A, 1990
Zortech Incorporated, 1165 Massachusetts, Arlington, MA 02174, USA

[ZDBV90] ZDBV86 COM Zortech Virtual debugger Version 2 10A, 1990
Zortech Incorporated, 1165 Massachusetts, Arlington, MA 02174, USA

[ZED89] ZED COM Version 3 00 Zortech Limited, 1989
Zortech Incorporated, 1165 Massachusetts, Arlington, MA 02174, USA

[ZMAP90] ZMAP EXE Zortech Standard mapper, 1990
Zortech Incorporated, 1165 Massachusetts, Arlington, MA 02174, USA

[ZMP290] ZMAP286 EXE DOS Extended mapper, 1990
Zortech Incorporated, 1165 Massachusetts, Arlington, MA 02174, USA

[ZORC90] Zortech C++ Version 2 1 Update Guide 28 May 1990, 1st print
Zortech Incorporated, 1165 Massachusetts, Arlington, MA 02174, USA

[ZORC89] Zortech C++ Version 2 0 compiler reference manual
Zortech Incorporated, 1165 Massachusetts, Arlington, MA 02174, USA

[ZORC89b] Zortech C++ Compiler Version 2 0 Upgrade for guide OS/2
Zortech Incorporated, 1165 Massachusetts, Arlington, MA 02174, USA

[ZORC88] Zortech C++ version 1 07 release notes
Zortech Incorporated, 1165 Massachusetts, Arlington, MA 02174, USA

[ZORT90a] DOS 386 Zortech C++ version 2 18 for 80386 microprocessor applications, 1990
Zortech Incorporated, 1165 Massachusetts, Arlington, MA 02174, USA

[ZORT90b] Zortech C++ Version 2 1, 1990
Zortech Incorporated, 1165 Massachusetts, Arlington, MA 02174, USA

[ZORT89] Zortech C++ Version 2 0, 1989
Zortech Incorporated, 1165 Massachusetts, Arlington, MA 02174, USA

[ZORT88] Zortech C++ version 1 07, 1988
Zortech Incorporated, 1165 Massachusetts, Arlington, MA 02174, USA

[ZTC90a] ZTC COM DOS 386 Zortech compiler version 2 18, 1990
Zortech Incorporated, 1165 Massachusetts, Arlington, MA 02174, USA

[ZTC90b] ZTC COM Zortech C++ Compiler version 2 1, October 1990
Zortech Incorporated, 1165 Massachusetts, Arlington, MA 02174, USA

[ZWB90a] ZWB EXE Zortech C++ Workbench version 1 12, 1990 Zortech Incorporated, 1165 Massachusetts, Arlington, MA 02174, USA

[ZWB90b] ZWB EXE Zortech C++ Workbench version 1 03, July 1990 Zortech Incorporated, 1165 Massachusetts, Arlington, MA 02174, USA

Appendix 4 Source Code Listings.

```
// file  animconc cpp
// Vr 5 0 03 07 90 11 30 PK
// Stores background behind object in dynamic buffer
#include "animconc hpp"
#include <msmouse h>
#include <fg h>
#include <stdlib h>

void animconc  pdraw() {
     msm_hidecursor(),

     anim_conc_outer_box [FG_X1]=x_center,
     anim_conc_outer_box [FG_Y1]=y_center,
     anim_conc_outer_box [FG_Y2]=y_center+height+(a*mul),
     anim_conc_outer_box [FG_X2]=x_center+width+(a*mul),

     anim_conc_fbox [FG_X1]=anim_conc_outer_box [FG_X1]+3,
     anim_conc_fbox [FG_Y1]=anim_conc_outer_box [FG_Y1]+3,
     anim_conc_fbox [FG_Y2]=anim_conc_outer_box [FG_Y2]-3,
     anim_conc_fbox [FG_X2]=anim_conc_outer_box [FG_X2]-3,

     x_min_extn=anim_conc_outer_box [FG_X1]-1,
     y_min_extn=anim_conc_outer_box [FG_Y1]-1,
     x_max_extn=anim_conc_outer_box [FG_X2]+1,
     y_max_extn=anim_conc_outer_box [FG_Y2]+1,

     background_box[FG_X1]=x_min_extn,
     background_box[FG_Y1]=y_min_extn,
     background_box[FG_X2]=x_max_extn,
     background_box[FG_Y2]=y_max_extn,

     pixel_buffer_length=(sizeof(fg_color_t)*fg_box_area(background_box)),
     pixel_buffer=malloc (pixel_buffer_length),
     fg_readbox (background_box,pixel_buffer),
     bytez=(sizeof(int))/2,
     bitmap=total_name,
     fptr=fopen (bitmap,"wb"),
     fwrite (pixel_buffer,bytez,pixel_buffer_length,fptr),
     fclose (fptr),
     free (pixel_buffer),

     fg_fillbox (FG_BLACK,FG_MODE_SET,~0,background_box),
     fg_drawbox(FG_LIGHT_WHITE,FG_MODE_SET,~0,FG_LINE_SOLID,anim_conc_outer_box
                                                                    ,fg_displaybox ),

     fg_fillbox (FG_BLUE,FG_MODE_SET,~0,anim_conc_fbox),

     if (previous_value>100) previous_dot_value=100,
     else previous_dot_value=previous_value,
     while (previous_dot_value>0){
          scolor=10+previous_status*2,
          randxy(),
          xd[previous_dot_value]=dot_x,
          yd[previous_dot_value]=dot_y,
          fg_drawdot(scolor,FG_MODE_SET,~0,xd[previous_dot_value]
                                          ,yd[previous_dot_ value] ),
```

```
                    previous_dot_value--,
                    }

                if (dot_value>previous_dot_value) {
                    scolor=10+status*2,
                    randxy(),
                    xd[dot_value]=dot_x,
                    yd[dot_value]=dot_y,
                    fg_drawdot(scolor,FG_MODE_SET,~0,xd[dot_value],
                                                        ,yd[dot_value]),
                    dot_value--,
                    }

                if ((dot_value==previous_dot_value)&&(dot_value>0)) {
                    scolor=10+previous_status*2,
                    fg_drawdot(FG_BLUE,FG_MODE_SET,~0
                      ,xd[previous_dot_value],yd[previous_dot _value ]),
                    scolor=10+status*2,
                    randxy(),
                    xd[dot_value]=dot_x,
                    yd[dot_value]=dot_y,
                    fg_drawdot(scolor,FG_MODE_SET,~0,xd[dot_value]
                                                        ,yd[dot_value]),
                    previous_dot_value--,
                    dot_value--,
                    }
                }
            previous_status=status,
            msm_showcursor(),
        }
}
void animconc  randxy() {
            in_box=0,
            while (in_box==0){   // non-zero if pt in box
                    test=rand(),
                    test=test/32767,
                    dot_x=anim_conc_fbox[FG_X1]+(test)*
                          (anim_conc_fbox[FG_X2]-anim_conc_fbox [FG_X1 ]),
                    test=rand(),
                    test=test/32767,
                    dot_y=anim_conc_fbox[FG_Y1]+(test)*
                                    (anim_conc_fbox[FG_Y2]-anim_conc_fbox [FG_Y1 ]),
                    in_box=fg_pt_inbox(anim_conc_fbox,dot_x,dot_y),
            }
}


void animconc  perase() {
        msm_hidecursor(),

        fg_fillbox (FG_BLACK,FG_MODE_SET,~0,anim_conc_fbox),
        fg_drawbox (FG_LIGHT_WHITE,FG_MODE_XOR,~0,FG_LINE_SOLID
                                    ,anim_conc_outer_box,fg_disp laybox ),
            pixel_buffer=malloc (pixel_buffer_length),
            bitmap=total_name,
            fptr=fopen (bitmap,"rb"),
```

```
          min_mul=0,
          conv_m=1,
          conv_c=0
          value=(raw_value*conv_m)+conv_c,
          status=state,
          status_extn=extent_onoff,
          previous_value=value,
          dot_value=value,
          previous_dot_value=dot_value,
          previous_status=status,
          a=1,
          height=40,
          width=40,
          action_default=4,
          action=action_default,
          pextent_set(status_extn),
          max_extent_box[0]=x_center-1,
          max_extent_box[1]=y_center-1,
          max_extent_box[2]=x_center+width+(a*max_mul)+1,
          max_extent_box[3]=y_center+height+(a*max_mul)+1,
//        pixel_buffer_length=(sizeof(fg_color_t)*fg_box_area(max_extent_box))
          size=sizeof(fg_color_t),
//        pixel_buffer=farcalloc (pixel_buffer_length,size),
//        pixel_buffer=farmalloc (pixel_buffer_length)
          file_rand=rand(),
          itoa (file_rand,sub_name,10),
          total_name[0]='a',
          total_name[1]='c',
          q=2,
          while (sub_name[q-2]!='\0'){
                    total_name[q]=sub_name[q-2],
                    q++, }
          total_name[q]=' ',
          total_name[q+1]='s',
          total_name[q+2]='a',
          total_name[q+3]='v',
          total_name[q+4]='\0',
          bitmap=total_name,
     }

     void perase(),
     ~animconc() {
//        farfree (pixel_buffer),
          perase(),
/*        source[0]='d',
          source[1]='e',
          source[2]='l',
          source[3]=' ',
          for (alpha=4,*bitmap!='\0',alpha++){
                    source[alpha]=*bitmap++,}
          source[alpha]='\0',
          system (source),
*/        }
},
#endif ANIMCONC_HPP
```

```
// set radius
x_radius=x_raddef+(a*mul),
outer_radius=x_radius+3,

x_min_extn=x_center-1+(x_radius*(cos(4 0)))-10-a*mul,
y_min_extn=y_center-outer_radius-ooly_const-a*mul/2,
x_max_extn=x_center+1+(x_radius*(cos(1 0)))+10+a*mul,
y_max_extn=y_center+outer_radius,

background_box[FG_X1]=x_min_extn,
background_box[FG_Y1]=y_min_extn,
background_box[FG_X2]=x_max_extn,
background_box[FG_Y2]=y_max_extn,

pixel_buffer_length=(sizeof(fg_color_t)*
                                    fg_box_area(background_box)),
pixel_buffer=malloc (pixel_buffer_length),
fg_readbox (background_box,pixel_buffer),
bytez=(sizeof(int))/2,
bitmap=total_name,
fptr=fopen (bitmap,"wb"),
fwrite (pixel_buffer,bytez,pixel_buffer_length,fptr),
fclose (fptr),
free (pixel_buffer),

fg_fillbox (fgcolor,FG_MODE_SET,~0,background_box),

// status colour
scolor=10+(status*2),
fg_drawarc(scolor,FG_MODE_SET,~0,x1,y1,x_radius,angl,ang2,
                                                    fg_displaybox),

port_botline_right[FG_X1]=x1+1+(x_radius*(cos(0 5))),
port_botline_right[FG_Y1]=y1+(x_radius*(sin(0 5))),
port_botline_right[FG_X2]=port_botline_right[FG_X1]+5+a*mul,
port_botline_right[FG_Y2]=port_botline_right[FG_Y1],
fg_drawline(scolor,FG_MODE_XOR,~0,FG_LINE_SOLID,port_botline_right)

port_topline_right[FG_X1]=x1+1+(x_radius*(cos(1 0))),
port_topline_right[FG_Y1]=y1+(x_radius*(sin(1 0))),
port_topline_right[FG_X2]=port_topline_right[FG_X1]+10+a*mul,
port_topline_right[FG_Y2]=port_topline_right[FG_Y1],
fg_drawline(scolor,FG_MODE_XOR,~0,FG_LINE_SOLID,port_topline_right),

port_topline_left[FG_X1]=x1-1+(x_radius*(cos(3 5))),
port_topline_left[FG_Y1]=y1+(x_radius*(sin(3 5))),
port_topline_left[FG_X2]=port_topline_left[FG_X1]-5-a*mul,
port_topline_left[FG_Y2]=port_topline_left[FG_Y1],
fg_drawline(scolor,FG_MODE_XOR,~0,FG_LINE_SOLID,port_topline_left),

port_botline_left[FG_X1]=x1-1+(x_radius*(cos(4 0))),
port_botline_left[FG_Y1]=y1+(x_radius*(sin(4 0))),
port_botline_left[FG_X2]=port_botline_left[FG_X1]-10-a*mul,
port_botline_left[FG_Y2]=port_botline_left[FG_Y1],
fg_drawline(scolor,FG_MODE_XOR,~0,FG_LINE_SOLID,port_botline_left),
```

```
                    // redraw
                    scolor=10+(status*2),
                    fg_drawarc(scolor,FG_MODE_SET,~0,x1,y1,x_radius,ang1,ang2
                                                              ,fg_displaybox),
                    fg_drawline(scolor,FG_MODE_XOR,~0,FG_LINE_SOLID
                                                            ,port_botline_right),
                    fg_drawline(scolor,FG_MODE_XOR,~0,FG_LINE_SOLID
                                                            ,port_topline_right),
                    fg_drawline(scolor,FG_MODE_XOR,~0,FG_LINE_SOLID
                                                             ,port_botline_left),
                    fg_drawline(scolor,FG_MODE_XOR,~0,FG_LINE_SOLID
                                                            ,port_topline_left),
                    fg_fillpolygon(scolor,FG_MODE_XOR,~0,4,poly,fg displaybox),
                    previous_status=status,
                          }
                fg_drawline(color,FG_MODE_XOR,~0,FG_LINE_SOLID,line),
                fg_drawline(color,FG_MODE_XOR,~0,FG_LINE_SOLID
                                                      ,right_angle_line),
                ang3=ang3-((value/100)*delta_angle),
                if (ang3 <= 0) ang3=3600+ang3,
                ang4=ang3-900,
                if (ang4 <= 0) ang4=3600+ang4,

/*              ang3=ang3+((value/100)*delta_angle),
                if (ang3>=3600) ang3=ang3-3600,
                ang4=ang3+900,
                if (ang4>=3600) ang4=ang4-3600,
*/
                angle=(ang3/fullcir),
                angle=angle*2*3 1415926,
                angle2=(ang4/fullcir),
                angle2=angle2*2*3 1415926,

                line [FG_X1]=x1-((0 75*x_radius)*(cos(angle))),
                line [FG_Y1]=y1-((0 75*x_radius)*(sin(angle))),
                line [FG_X2]=x1+((0 75*x_radius)*(cos(angle))),
                line [FG_Y2]=y1+((0 75*x_radius)*(sin(angle))),
                fg_drawline(color,FG_MODE_XOR,~0,FG_LINE_SOLID,line),

                right_angle_line [FG_X1]=x1-((0 75*x_radius)*(cos(angle2))),
                right_angle_line [FG_Y1]=y1-((0 75*x_radius)*(sin(angle2))),
                right_angle_line [FG_X2]=x1+((0 75*x_radius)*(cos(angle2))),
                right_angle_line [FG_Y2]=y1+((0 75*x_radius)*(sin(angle2))),
                fg_drawline(color,FG_MODE_XOR,~0,FG_LINE_SOLID
                                                      ,right_angle_line),
                mod=1,
                msm_showcursor(),
        }
}

void animple  perase() {
        msm_hidecursor(),

        // set radius
        x_radius=x_raddef+(a*mul),
```

```
float poly_const,
fg_coord_t poly[10],
int degrad,
public

void pdraw(),
void panimate(),
void pextent_pgen(),
void pextent_pdraw(),
void pextent_perase(),

   animpie(unsigned  x,unsigned  y,int  id,float  m,float  raw_value,unsigned
                                                        state,int extent_onoff)    () {
        identifier=id,  // cadshape id
        cad_type=4,      // DIPAIP type
        color=15,        // default colour
        fgcolor=0,        // default colour
        degrad=1,        // degrees - radians falg
        delta_angle=250,  // default value for pie increment
        fullcir=3600,
        x_raddef=10,     // default pie radius
        poly_const=5,    // triangular pedestial constant
        a=2,               // multiplication factor
        ang3=3600,           // running pie angle for animation
        arg2=3600,       // default value for pie stop angle
        ang1=0,           // default value for pie start angle
        mod=0,            // animation flag
        // pass parameters
        x_center = x, y_center = y
        conv_m=1,                  // Conversion multiplier
        conv_c=0,                  // Conversion constant
        value=(raw_value*conv_m)+conv_c,
        status=state,
        previous_status=status,
        status_extn=extent_onoff,
        mul=m,
        min_mul=0,
        max_mul=10,
        action_default=2,
        action=action_default
        pextent_set(status_extn),

        // set radius
        x_radius=x_raddef+(a*max_mul),
        outer_radius=x_radius+3,

        max_extent_box[0]=x_center-1+(x_radius*(cos(4 0)))-10-a*max_mul,
        max_extent_box[1]=y_center-outer_radius-poly_const-a*max_mul/2,
        max_extent_box[2]=x_center+(x_radius*(cos(1 0)))+10+a*max_mul,
        max_extent_box[3]=y_center+outer_radius,

//      pixel_buffer_length=(sizeof(fg_color_t)*
                                          fg_box_area(max_extent_box)),
        size=sizeof(fg_color_t),
//      pixel_buffer=farcalloc (pixel_buffer_length,size),
//      pixel_buffer=malloc (pixel_buffer_length),
```

```
// file  animpipe cpp
// V4 3   Flash graphics 19 02 90 18 50
// V4 8   05 04 90 17 00 PK
// Pushes background 'behind' object to dynamic buffer on draw
// Pulls  background back to screen on erase
// V5 0   14/06/90 12 00
// Horizontal and verticle
#include "animpipe npp"
#include <msmouse h>
#include <bios h>
#include <fg h>

void animpipe  pextent_pgen() {

          extents_sim[0]=x_center-1,
          extents_sim[1]=y_center-1,
          extents_sim[2]=x_center
                      +((length+1+(a*mul))*horiz)+((diameter+1)*vert),
          extents_sim[3]=y_center
                      +((diameter+1)*horiz)+((length+1+(a*mul))*vert),
                    }

void animpipe  pextent_pdraw() {
          msm_hidecursor(),

          x_min_extn=x_center-1,
          y_min_extn=y_center-1,
          x_max_extn=x_center
                    +((length+1+(a*mul))*horiz)+((diameter+1)*vert),
          y_max_extn=y_center
                    +((diameter+1)*horiz)+((length+1+(a*mul))*vert),

          extent_box [FG_X1]=x_min_extn,
          extent_box [FG_Y1]=y_min_extn,
          extent_box [FG_X2]=x_max_extn,
          extent_box [FG_Y2]=y_max_extn,
          if (extent_overlap==0) {extent_color=color,}
          else {extent_color=color-2,}
          fg_drawbox(extent_color,FG_MODE_XOR,~0,FG_LINE_MEDIUM_DASHED
                                                  ,extent_box,fg_displaybox),
          msm_showcursor(),
}
void animpipe  pextent_perase() {
          msm_hidecursor(),
          fg_drawbox (extent_color,FG_MODE_XOR,~0,FG_LINE_MEDIUM_DASHED
                                                 ,extent_box,fg_display box),
          msm_showcursor(),
}


void animpipe  pdraw() {
          msm_hidecursor(),
          flow_box [FG_X1]=x_center,
          flow_box [FG_Y1]=y_center,
          flow_box [FG_X2]=x_center+((length+(a*mul))*horiz)+(diameter*vert),
          flow_box [FG_Y2]=y_center+(diameter*horiz)+((length+(a*mul))*vert),
```

```
                        _bios_timeofday (0,&previous_time),
                        msm_showcursor(),
                    }
                    else{}
            }
}
void animpipe  resetvar(){
                        x_length=x_center+length+(a*mul),
                        y_length=y_center+length+(a*mul),

                        if (previous_value>=0)
                              xx=((x_center+shift)*horiz)+((x_center+1)*vert),
                        else xx=((x_length-shift)*horiz)+((x_center+1)*vert),

                        if (previous_value>=0)
                               yy=((y_center+1)*horiz)+((y_center+shift)*vert),
                        else yy=((y_center+1)*horiz)+((y_length-shift)*vert),

                        x_diameter=x_center+diameter,
                        y_diameter=y_center+diameter,
}
void animpipe  fillpipe(){
                        while ((xx<x_length)&&(horiz==1)&&
                                                (previous_value>=0)){
                            while (yy<y_diameter){
                                    fg_drawdot(scolor,FG_MODE_XOR,~0,xx,yy),
                                    yy=yy+alpha,
                            }
                        yy=y_center+1,
                        xx=xx+beta,
                        }

                        while ((xx>x_center)&&(horiz==1)&&(previous_value<0)){
                            while (yy<y_diameter){
                                    fg_drawdot(scolor,FG_MODE_XOR,~0,xx,yy),
                                    yy=yy+alpha,
                            }
                            yy=y_center+1,
                            xx=xx-beta,
                        }

                        while ((yy<y_length)&&(vert==1)&&(previous_value>=0)){
                            while (xx<x_diameter){
                                    fg_drawdot(scolor,FG_MODE_XOR,~0,xx,yy),
                                    xx=xx+alpha,
                                    }
                            xx=x_center+1,
                            yy=yy+beta,
                        }

                        while ((yy>y_center)&&(vert==1)&&(previous_value<0)){
                            while (xx<x_diameter){
                                    fg_drawdot(scolor,FG_MODE_XOR,~0,xx,yy),
                                    xx=xx+alpha,
                                      }
                            xx=x_center+1,
```

```
// file  animpipe npp
// Vr 5 0 12/06/90 11 45 PK
#ifndef ANIMPIPE_HPP
#define ANIMPIPE_HPP

#include "cadshape hpp"
# include <fg h>

class animpipe   public cadshape {

   protected
   char * bitmap,
   int a,prevcolor,shift,alpha,beta,eta,
   fg_box_t flow_box,
   fg_coord_t yy,xx,xx_saved,yy_saved,
   int length,diameter,y_length,x_diameter,x_length,y_diameter,,
   long current_time,
   long previous_time,
   long inter_sample_time,
   int vert,horiz,

   // private methods
   void resetvar(),
   void fillpipe(),

   public

      void pdraw(),
      void panimate(),
      void pextent_pgen(),
      void pextent_pdraw(),
      void pextent_perase(),

      animpipe(unsigned x,unsigned y,int  id,unsigned  xref,unsigned  yref,float
            m,float raw_value,unsigned state,int extent_onoff,int verthorz) (){
           x_center = x, y_center = y,
           identifier=id, // cadshape id
           cad_type=4,     // DIPAIP type
           vert=verthorz,
           horiz=0,
           if (verthorz==0) horiz=1
           mul=m,
           max_mul=200,
           min_mul=0,
           conv_m=1,                    // Conversion multiplier
           conv_c=0,                    // Conversion constant
           value=(raw_value*conv_m)+conv_c,
           previous_value=value,
           status=state,
           status_extn=extent_onoff,
           a=1,
           fgcolor=0,
           color=15,
           shift=1,                     // deafult pixel shift
           alpha=4,                     // y inter dot spacing
           beta=4,                      // x inter dot spacing
```

```
// file  animplot.cpp
// Vrs 2  20/07/90 15:00 PK
#include "animplot.hpp"
#include <msmouse.h>
#include <fg.h>
#include <time.h>
#include <stdio.h>

void animplot::extent_pgen() {
          // expand area if required
          x_min=x_center-x_offset-(a*mul),
          y_min=y_center-y_offset-(a*(mul/10)),
          x_max=x_center+x_offset+(a*mul),
          y_max=y_center+y_offset+(a*(mul/10)),

          // extents
          extents_sim[0]=x_min-6,
          extents_sim[1]=y_min-8-fg_box_height(fg_charbox),
          extents_sim[2]=x_max+6+4*fg_box_width(fg_charbox),
          extents_sim[3]=y_max+6,
          }

void animplot::extent_pdraw() {
          msm_hidecursor(),

          x_min=x_center-x_offset-(a*mul),
          y_min=y_center-y_offset-(a*(mul/10)),
          x_max=x_center+x_offset+(a*mul),
          y_max=y_center+y_offset+(a*(mul/10)),

          // extents
          x_min_extn=x_min-6,
          y_min_extn=y_min-8-fg_box_height(fg_charbox),
          x_max_extn=x_max+6+4*fg_box_width(fg_charbox),
          y_max_extn=y_max+6,

          extent_box [FG_X1]=x_min_extn,
          extent_box [FG_Y1]=y_min_extn,
          extent_box [FG_X2]=x_max_extn,
          extent_box [FG_Y2]=y_max_extn,
          if (extent_overlap==0) {extent_color=color,}
          else {extent_color=color-2,}
          fg_drawbox (extent_color,FG_MODE_XOR,~0,FG_LINE_MEDIUM_DASHED
                                                ,extent_box,fg_display_box),
          msm_showcursor(),

}
void animplot::extent_perase() {
          msm_hidecursor(),
          fg_drawbox (extent_color,FG_MODE_XOR,~0,FG_LINE_MEDIUM_DASHED
                                                ,extent_box,fg_display_box),
          msm_showcursor(),
}

void animplot::pdraw() {
          msm_hidecursor(),
```

```
ur_line[FG_X1]=anim_plot_obox[FG_X2],
ur_line[FG_Y1]=anim_plot_obox[FG_Y1],
ur_line[FG_X2]=anim_plot_obox[FG_X1],
ur_line[FG_Y2]=anim_plot_obox[FG_Y1],

ll_line_2[FG_X1]=anim_plot_obox[FG_X1]-1,
ll_line_2[FG_Y1]=anim_plot_obox[FG_Y1]-1,
ll_line_2[FG_X2]=anim_plot_obox[FG_X1]-1,
ll_line_2[FG_Y2]=anim_plot_obox[FG_Y2]+1,

ul_line_2[FG_X1]=anim_plot_obox[FG_X1]-1,
ul_line_2[FG_Y1]=anim_plot_obox[FG_Y2]+1,
ul_line_2[FG_X2]=anim_plot_obox[FG_X2]+1,
ul_line_2[FG_Y2]=anim_plot_obox[FG_Y2]+1,

lr_line_2[FG_X1]=anim_plot_obox[FG_X2]+1,
lr_line_2[FG_Y1]=anim_plot_obox[FG_Y2]+1,
lr_line_2[FG_X2]=anim_plot_obox[FG_X2]+1,
lr_line_2[FG_Y2]=anim_plot_obox[FG_Y1]-1,

ur_line_2[FG_X1]=anim_plot_obox[FG_X2]+1,
ur_line_2[FG_Y1]=anim_plot_obox[FG_Y1]-1,
ur_line_2[FG_X2]=anim_plot_obox[FG_X1]-1,
ur_line_2[FG_Y2]=anim_plot_obox[FG_Y1]-1,


fg_drawline(FG_BLACK,FG_MODE_SET,~0,FG_LINE_SOLID,ll_line),
fg_drawline(FG_BLACK,FG_MODE_SET,~0,FG_LINE_SOLID,ul_line),
fg_drawline(FG_WHITE,FG_MODE_SET,~0,FG_LINE_SOLID,lr_line),
fg_drawline(FG_WHITE,FG_MODE_SET,~0,FG_LINE_SOLID,ur_line),

fg_drawline(FG_BLACK,FG_MODE_SET,~0,FG_LINE_SOLID,ll_line_2),
fg_drawline(FG_BLACK,FG_MODE_SET,~0,FG_LINE_SOLID,ul_line_2),
fg_drawline(FG_WHITE,FG_MODE_SET,~0,FG_LINE_SOLID,lr_line_2),
fg_drawline(FG_WHITE,FG_MODE_SET,~0,FG_LINE_SOLID,ur_line_2),


x_text=anim_plot_obox [FG_X2]+4,
y_text=anim_plot_obox [FG_Y2]-0 75*fg_box_height(fg charbox),
valuout[0]='1',
valuout[1]='0',
valuout[2]='0',
valuout[3]='%',
valuout[4]='\0',
fg_puts (FG_MAGENTA,FG_MODE_XOR,~0,FG_ROT0,x_text,y_text,valuout
                                                    ,fg_displaybox),

y_text=anim_plot_obox [FG_Y1]-0 25*fg_box_height(fg charbox),
valuout[0]=' ',
valuout[1]='0',
valuout[2]=' ',
valuout[3]='%',
valuout[4]='\0',
fg_puts (FG_MAGENTA,FG_MODE_XOR,~0,FG_ROT0,x_text,y_text,valuout
                                                    ,fg_displaybox),
```

```
valuout[m]=' ',
valuout[m+1]='S',
valuout[m+2]='e',
valuout[m+3]='c',
valuout[m+4]='\0',
fg_puts (FG_BLACK,FG_MODE_SET,~0,FG_ROT0,x_text,y_text,valuout
                                                    ,fg_displaybox),
total_time=difftime(times[0],times[N-1]),
if (total_time ==0){
          total_time=1,}            // avoid div by 0 (delta_time=0),
delta_time=( total_time/(x_max-x_min) ),

// generate x co-ords
x[0]=x_max,
for (m=1, m<N, m++)
          {
          x[m]=x[m-1]-( difftime(times[m-1],times[m])/delta_time),

          // time record out of bounds
          if ((x[m] > x_max) || (x[m] <= x_min))
          {
          x[m] = x[m-1],}
           }

   // generate corresponding # y co-ords
   for ( p=0, p<N, p++)
          {
          y[p]=( y_min+((values[p]/100)*(y_max-y_min))),
          }

   // plot waveform
   for (q=0, q<(N-1), q++)
          {
          xc=x[q], yc=y[q],

          if (yc > y_max){
                    yc=y_max,}
          if (yc < y_min){
                    yc=y_min,}

          plotline [FG_X1]=xc,
          plotline [FG_Y1]=yc,
          xc=x[q+1], yc=y[q+1],

          if (attributes[q]==0) scolor=FG_LIGHT_GREEN,
          if (attributes[q]==2) scolor=FG_LIGHT_RED,

          if (yc > y_max)
                    {yc=y_max,
                     scolor=out_of_range_color,}

          if (yc < y_min)
                    {yc=y_min,
                     scolor=out_of_range_color,}
          if ((attributes[q]==1)||(attributes[q+1]==1))
              scolor=no_log_color,
```

```
// shift all raw data array entries
for (i=0, i<(N-1), i++)
          {
          values[(N-1)-i]=values[(N-2)-i],
          times[(N-1)-i]=times[(N-2)-i],
          attributes[(N-1)-i]=attributes[(N-2)-i],
          }

// insert current raw data
values[0]=value,
times[0]=current_time,
scolor=10+status*2,
attributes[0]=status*2,

n=0,
while (n==0){
          last_valid_log_time=times[1],
          for (m=0,m<(N-1),m++) {
                  if (times[m]>=last_valid_log_time+1 1*
                                    debounce_time){
                      //shift from t[m+1] backwards
                      for (i=0, i<(N-(m+2)), i++){
                          values[(N-1)-i]=values[(N-2)-i],
                          times[(N-1)-i]=times[(N-2)-i],
                          attributes[(N-1)-i]=
                                    attributes[(N-2)-i],
                          }
                      times[m+1]=times[m]-debounce_time,
                      attributes[m+1]=1, // no logging
                                values[m+1]=0,
                      }
                  else  {
                   //if (times[m]>=last_valid_log_time+
                                      1 1*debounce_time)
                                // discontinutity
                                // {attributes[m]=2,}
                               m=N-1,
                               n=1, // break while loop
                               }
                  }
          }

 for (i=1, i<N, i++){
          if (times[i]==0){
           times[i]=times[i-1]-debounce_time,
                  attributes[i]=1, // no logging
                  values[i]=0,}
        }

 for (i=0, i<(N-1), i++)
          {
          if (times[i+1] > times[i]){
            times[i+1]=times[i]-debounce_time,
                    attributes[i+1]=1, // no logging
                    values[i+1]=0,
                    }
```

```
                        // plot waveform
                        // printf ("commencing the plot"),
                        for (q=0, q<(N-1), q++)
                          {
                          xc=x[q], yc=y[q],
                          if (yc > y_max){
                                  yc=y_max,}
                          if (yc < y_min){
                                  yc=y_min,}

                          plotline [FG_X1]=xc,
                          plotline [FG_Y1]=yc,
                          xc=x[q+1], yc=y[q+1],

                          if (attributes[q]==0) scolor=FG_LIGHT_GREEN,
                          if (attributes[q]==2) scolor=FG_LIGHT_RED,

                          if (yc > y_max)
                                  {yc=y_max,
                                  scolor=out_of_range_color,}

                          if (yc < y_min)
                                  {yc=y_min,
                                  scolor=out_of_range_color,}
                          if ((attributes[q]==1)||(attributes[q-1]==1))
                                  scolor=no_log_color,
                          plotline[FG_X2]=xc,
                          plotline [FG_Y2]=yc,

                          fg_drawline (scolor,FG_MODE_SET,~0,FG_LINE_SOLID
                                                                  ,plotline),
                          fg_drawdot(FG_BLACK,FG_MODE_SET,~0,xc,(y_minbox-6)),
                        }
                        msm_showcursor(),
                }
        }
}


void animplot_perase() {
        msm_hidecursor(),
/*      y_text=y_minbox-y_text_offset-fg_box_height(fg charbox),
        x_text=x_minbox+fg_box_width(fg charbox),
        fg_puts (FG_RED,FG_MODE_SET,~0,FG_ROT0,x_text,y_text,valuout
                                                        ,fg_displaybox),
        y_text=y_minbox-(2*fg_box_height(fg charbox)),
        x_text=x_maxbox,
        fg_puts (FG_WHITE,FG_MODE_XOR,~0,FG_ROT270,x_text,y_text,timemsg0
                                                        ,fg_displaybox),
        x_text=x_minbox,
        fg_puts (FG_WHITE,FG_MODE_XOR,~0,FG_ROT270,x_text,y_text,timemsg1
                                                        ,fg_displaybox),

          // plot waveform
```

```
                    }
          msm_showcursor(),
}

// file  animplot.hpp
// V:5.20 20/07/90 14:45
// dasn on time intervals
#ifndef ANIMPLOT_HPP
#define ANIMPLOT_HPP

#include "cadshape.hpp"
#include <time.h>
#include <string.h>
#include <stdlib.h>

class animplot   public cadshape {
   char * bitamp,
   float x_min,y_min,x_max,y_max,xc,yc,
   fg_coord_t x_fill,y_fill,
   float x_text,y_text,
   float x_offset,y_offset,
   float y_text_offset,
   float x_minbox,y_minbox,x_maxbox,y_maxbox,
   int i,l,m,n,p,q,N,M,limit,a,
   fg_color_t out_of_range_color,precolor,
   fg_color_t no_log_color,
   fg_line_t plotline,
   fg_box_t anim_plot_box,
   fg_box_t anim_plot_obox,
   float values[100],
   time_t times[100],
   int attributes[100],
   time_t last_valid_log_time,
   time_t current_time,
   long run_debounce_time,
   double delta_time,
   double total_time,
   double debounce_time,
   double inter_sample_time,
   float x[100],
   float y[100],
   char datetime[26],
   char timemsg0[9],
   char timemsg1[9],
   char * timeptr,
   size_t timesize,
   int byte_length,
   fg_color_t far *pixel_in,
   char *valu,                                  // for converted debounce_time
   char valuout[11],                           // "    "            "
   int dec,sign,digits,                        // for float -> string conversion
   fg_line_t ll_line,ul_line,lr_line,ur_line,
   fg_line_t ll_line_2,ul_line_2,lr_line_2,ur_line_2,

   public
```

```
          size=sizeof(fg_color_t),
//        pixel_buffer=farmalloc (pixel_buffer_length),
//        pixel_buffer=farcalloc (pixel_buffer_length,size),
          file_rand=rand(),
          itoa (file_rand,sub_name,10),
          total_name[0]='a',
          total_name[1]='o',
          q=2,
          while (sub_name[q-2]!='\0'){
                    total_name[q]=sub_name[q-2],
                    q++,
                    }
          total_name[q]=' ',
          total_name[q+1]='s',
          total_name[q+2]='a',
          total_name[q+3]='v',
          total_name[q+4]='\0',

          bitmap=total_name,
     }


          void perase(),
          ~animplot() {
          perase(),
/*        source[0]='d',
          source[1]='e',
          source[2]='l',
          source[3]=' ',
          for (alpha=4,*bitmap!='\0',alpha++){
                    source[alpha]=*bitmap++,}
          source[alpha]='\0',
          system (source),
*/
//        farfree (pixel_buffer),
//        farfree (pixel_in),
          }
},
#endif ANIMPLOT_HPP
```

```
     if (value<0)   scolor=10,
     fg_fillbox (scolor,FG_MODE_XOR,~0,anim_sqr_fbox),
     msm_showcursor(),
     }

void animsqr  pextent_pgen() {
     extents_sim[0]=x_center-1,
     extents_sim[1]=y_center-1,
     extents_sim[2]=x_center+width+(a*mul)+1,
     extents_sim[3]=y_center+height+(a*mul)+1,
          }

void animsqr  pextent_pdraw() {
     msm_hidecursor(),
     x_min_extn=x_center-1,
     y_min_extn=y_center-1,
     x_max_extn=x_center+width+(a*mul)+1,
     y_max_extn=y_center+height+(a*mul)+1,

     extent_box[FG_X1]=x_min_extn,
     extent_box[FG_Y1]=y_min_extn,
     extent_box[FG_X2]=x_max_extn,
     extent_box[FG_Y2]=y_max_extn,

     if (extent_overlap==0) {extent_color=color,}
     else {extent_color=color-2,}
     fg_drawbox (extent_color,FG_MODE_XOR,~0,FG_LINE_MEDIUM_DASHED,extent_box
                                                          ,fg_display box),
     msm_showcursor(),
}

void animsqr  pextent_perase() {
     msm_hidecursor(),
     fg_drawbox (extent_color,FG_MODE_XOR,~0,FG_LINE_MEDIUM_DASHED,extent_box
                                                                ,fg_display box),
     msm_showcursor(),
}

void animsqr  panimate() {
          action--,
          if (action==0){
                    action=action_default,
                    if ((status==1)&&(value==previous_value)){   //blink
                              fg_fillbox (scolor,FG_MODE_XOR,~0,anim_sqr_fbox),
                              }
                    if (value!=previous_value){
                              msm_hidecursor(),
                              previous_value=value,
                              test_dot_color=fg_readdot((anim_sqr_fbox[FG_X1]+1)
                                                  ,(anim_sqr_fbox[FG_Y1] +1)),
                              if (test_dot_color!=0)
                                   fg_fillbox(scolor,FG_MODE_XOR,~0,anim_sqr_fbox),
                              scolor=10+value/20,
                              if (value>99) scolor=14,
                              if (value<0)   scolor=10,
                              fg_fillbox (scolor,FG_MODE_XOR,~0,anim_sqr_fbox),
```

```
public

   void pdraw(),
   void panimate(),
   void pextent_pgen(),
   void pextent_odraw(),
   void pextent_perase(),

   animsqr(unsigned x, unsigned y, int id, float m, float raw_value, unsigned
                                                    state, int extent_onoff)    () {
        x_center = x, y_center = y,
        identifier=id,
        cad_type=4,                              // DIPAIP type
        mul=m,
        max_mul=12,
        min_mul=0,
        conv_m=1,                                 // Conversion multiplier
        conv_c=0,                                 // Conversion constant
        value=(raw_value*conv_m)+conv_c,
        status=state,
        status_extn=extent_onoff,
        previous_value=value,
        previous_status=status,
        a=1,
        height=15,
        width=15,
        fgcolor=0,
        color=15,
        action_default=1,
        action=action_default,
        pextent_set(status_extn),
        max_extent_box[0]=x_center-1,
        max_extent_box[1]=y_center-1,
        max_extent_box[2]=x_center+width+(a*max_mul)+1,
        max_extent_box[3]=y_center+height+(a*max_mul)+1,
//      pixel_buffer_length=(sizeof(fg_color_t)* fg_box_area(max_extent_box)),
        size=sizeof(fg_color_t),
//      pixel_buffer=farcalloc (pixel_buffer_length,size),
//      pixel_buffer=malloc (pixel_buffer_length),
        file_rand=rand(),
        itoa (file_rand,sub_name,10),
        total_name[0]='a',
        total_name[1]='s',
        q=2,
        while (sub_name[q-2]!='\0'){
                  total_name[q]=sub_name[q-2],
                  q++,
                  }
        total_name[q]=' ',
        total_name[q+1]='s',
        total_name[q+2]='a',
        total_name[q+3]='v',
        total_name[q+4]='\0',

        bitmap=total_name,
    }
```

```
    fwrite (pixel_buffer,bytez,pixel_buffer_length,fptr),
    fclose (fptr),
    free (pixel_buffer),

    fg_fillbox (fgcolor,FG_MODE_SET,~0,background_box),

    fg_drawline(color,FG_MODE_SET,~0,FG_LINE_SOLID,anim_sqr_line_one),
    fg_drawline(color,FG_MODE_SET,~0,FG_LINE_SOLID,anim_sqr_line_two),

    x_radius=(width+a*mul-2)/2+1,
    y_radius=y_radius_const+a*mul/10,
    angl=1800,ang2=3600,
    x_ell_center=x_center+(width+(a*mul))/2,
    y_ell_center_bot=y_center,
    fg_drawellipse(color,FG_MODE_SET,~0,x_ell_center,y_ell_center_bot,x_radius
                                    ,y_radius,angl,ang 2,fg_displaybox),

    angl=0,ang2=3600,
    y_ell_center_top=y_center+height+(a*mul)+1,
    fg_drawellipse(color,FG_MODE_XOR,~0,x_ell_center,y_ell_center_top,x_radius
                                    ,y_radius,an gl,ang 2,fg_displaybox),
    scolor=10+(previous_status*2),
    level_height=height-2*y_radius-2,
    if (previous_value>100) y_ell_center_level=y_center+height+(a*mul)+1,
    else   y_ell_center_level=y_center+((level_height+(a*mul))*
                                                (previous_value/100) )+1,
    angl=0,ang2=3600,
    fg_drawellipse (color,FG_MODE_SET,~0,x_ell_center,y_ell_center_level
                                                    ,x_radius,y_radius,
    angl,a ng2,fg_displaybox),
    x_fill=x_center+1,
    y_fill=y_center+1,
    fg_fill (x_fill,y_fill,scolor,color),
    msm_showcursor(),
    }

void animvol  pextent_pgen() {
    extents_sim[0]=x_center,
    extents_sim[1]=y_center-(y_radius_const+a*mul/10)-4,
    extents_sim[2]=x_center+width+(a*mul)+2,
    extents_sim[3]=y_center+height+(a*mul)+2+(y_radius_const+a*mul/10),
        }

void animvol  pextent_pdraw() {
    msm_hidecursor(),

    x_min_extn=x_center,
    y_min_extn=y_center-(y_radius_const+a*mul/10)-4,
    x_max_extn=x_center+width+(a*mul)+2,
    y_max_extn=y_center+height+(a*mul)+2+(y_radius_const+a*mul/10),

    extent_box[FG_X1]=x_min_extn,
    extent_box[FG_Y1]=y_min_extn,
    extent_box[FG_X2]=x_max_extn,
    extent_box[FG_Y2]=y_max_extn,
    if (extent_overlap==0) {extent_color=color,}
```

```
  if (status'=previous_status){
     // Remove second half of old level ellipse
     fg_drawellipse(FG_BLACK,FG_MODE_SET,~0,x_ell_center
               ,prev_y_ell_center_level,x_radius,y _radius,angl,ang2
                                                    ,fg_displaybox),
     fg_drawline(color,FG_MODE_SET,~0,FG_LINE_SOLID
                                                    ,anim_sqr_line_one),

     fg_drawline(color,FG_MODE_SET,~0,FG_LINE_SOLID
                                                    ,anim_sqr_line_two),
     fg_fill(x_fill,y_fill,FG_BLACK,color),

     fg_fill (x_fill,y_fill,scolor,color),

     }

  else {
     fg_drawline(color,FG_MODE_SET,~0,FG_LINE_SOLID
                                                  ,anim_sqr_line_one),
     fg_drawline(color,FG_MODE_SET,~0,FG_LINE_SOLID
                                                  ,anim_sqr_line_two),
     fg_fill (x_fill,y_fill,scolor,color),

     // Remove second half  of  old level ellipse
     fg_drawellipse(scolor,FG_MODE_SET,~0,x_ell_center
               ,prev_y_ell_center_level,x_radius,y_radius, angl,ang2
                                                   ,fg_displaybox),

     fg_drawline(color,FG_MODE_SET,~0,FG_LINE_SOLID
                                                   ,anim_sqr_line_one),
     fg_drawline(color,FG_MODE_SET,~0,FG_LINE_SOLID
                                                   ,anim_sqr_line_two),
  }
}

else {// value < previous_value
  // draw half new ellipse
  angl=1800,ang2=3600,
  fg_drawellipse (color,FG_MODE_SET,~0,x_ell_center
            ,y_ell_center_level,x_radius,y_radius, angl,ang2
                                                      ,fg_displaybox),
  // fill with black between higher previous level and
  // lower new level

  x_fill=x_ell_center,
  y_fill=prev_y_ell_center_level-y_radius-(prev_y_ell_center_level-
                                          y_ell_center_ level)/2,
  fg_drawline(color,FG_MODE_SET,~0,FG_LINE_SOLID
                                          ,anim_sqr_line_one),
  fg_drawline(color,FG_MODE_SET,~0,FG_LINE_SOLID
                                          ,anim_sqr_line_two),
  fg_fill(x_fill,y_fill,FG_BLACK,color),

  // Remove old level ellipse
  // angl=1,ang2=1799,
  angl=0,ang2=1800,
```

```
                fclose (fptr),
                fg_writebox (background_box,pixel_buffer),
                }

        free (pixel_buffer),

        if (fptr'=NULL){
                source[0]='d',
                source[1]='e',
                source[2]='l',

                source[3]=' ',
                for (alpha=4,*bitmap'='\0',alpha++){
                        source[alpha]=*bitmap++,}
                source[alpha]='\0',
                system (source),
                }


/*   fg_fill (x_fill,y_fill,FG_BLACK,color),
     angl=1800,ang2=3600,
     fg_drawell_pse (color,FG_MODE_XOR,~0,x_ell_center,y_ell_center_bot
                              ,x_radius,y_radius,angl,ang 2,fg_displaybox),
     angl=0,ang2=3600,
     fg_drawellipse (color,FG_MODE_XOR,~0,x_ell_center,y_ell_center_top
                              ,x_radius,y_radius,angl,ang 2,fg_displaybox),
     angl=0,ang2=3600,
     fg_drawellipse (FG_BLACK,FG_MODE_SET,~0,x_ell_center,y_ell_center_level
                              ,x_radius,y_radius,ang 1,ang2,fg_displaybox),
     fg_drawline(color,FG_MODE_XOR,~0,FG_LINE_SOLID,anim_sqr_line_one),
     fg_drawline(color,FG_MODE_XOR,~0,FG_LINE_SOLID,anim_sqr_line_two),
*/

     msm_showcursor(),
}

// file  animvol hpp
// Vr 4 10 28/04/90 12 30 PK
#ifndef ANIMVOL_HPP
#define ANIMVOL_HPP

#include "cadshape hpp"

class animvol   public cadshape {

protected
   float x1,y1,x2,y2,
   float height,width,level_height,
   fg_coord_t x_fill,y_fill,
   fg_coord_t x_radius,y_radius,x_ell_center,y_radius_const,
   fg_coord_t y_ell_center_bot,y_ell_center_top,y_ell_center_level
                                                    ,prev_y_ell_center_l evel,
   fg_coord_t deadband_level,
   fg_line_t anim_sqr_line_one,anim_sqr_line_two,
   int a,angl,ang2
   public
```

```
        total_name[q+4]='\0',

        bitmap=total_name,
    }

    void perase(),
    ~animvol() {
        perase(),
/*      source[0]='d',
        source[1]='e',
        source[2]='l',
        source[3]=' ',
        for (alpha=4 *bitmap!='\0',alpha++){
                source[alpha]=*bitmap++,}
        source[alpha]='\0',
        system (source),
*/      }
},

#endif ANIMVOL_HPP



// file  basictile cpp
// Vr 5 1 16 07 90 09 40 PK
#include "basictile hpp"
#include <stdlib h>
#include <msmouse h>
#include <fg h>


void basictile  basic_pdraw() {

        x_center=x_center,
        y_center=y_center,

        value_box[FG_X1]=x_center-0 5*fg_box_width(fg charbox),
        value_box[FG_Y1]=y_center,
        value_box[FG_X2]=x_center+6 5*(fg_box_width(fg charbox)),
        value_box[FG_Y2]=y_center+items*(fg_box_height(fg charbox)),

        frame_box[FG_X1]=value_box[FG_X1]-5,
        frame_box[FG_Y1]=value_box[FG_Y1]-5,
        frame_box[FG_X2]=value_box[FG_X2]+5,
        frame_box[FG_Y2]=value_box[FG_Y2]+5,

        frame_box_2[FG_X1]=value_box[FG_X1]-15,
        frame_box_2[FG_Y1]=value_box[FG_Y1]-15,
        frame_box_2[FG_X2]=value_box[FG_X2]+15,
        frame_box_2[FG_Y2]=value_box[FG_Y2]+15,

        ll_line[FG_X1]=frame_box[FG_X1],
        ll_line[FG_Y1]=frame_box[FG_Y1],
        ll_line[FG_X2]=frame_box[FG_X1],
        ll_line[FG_Y2]=frame_box[FG_Y2];
```

```
        fg_fillbox(FG_MAGENTA,FG_MODE_SET,~0,frame_box_2),
        fg_fillbox(FG_CYAN,FG_MODE_SET,~0,frame_box),

        fg_drawline(FG_BLACK,FG_MODE_SET,~0,FG_LINE_SOLID,ll_line),
        fg_drawline(FG_BLACK,FG_MODE_SET,~0,FG_LINE_SOLID,ul_line),
        fg_drawline(FG_WHITE,FG_MODE_SET,~0,FG_LINE_SOLID,lr_line),
        fg_drawline(FG_WHITE,FG_MODE_SET,~0,FG_LINE_SOLID,ur_line),

        fg_drawline(FG_BLACK,FG_MODE_SET,~0,FG_LINE_SOLID,ll_line_2),
        fg_drawline(FG_BLACK,FG_MODE_SET,~0,FG_LINE_SOLID,ul_line_2),
        fg_drawline(FG_WHITE,FG_MODE_SET,~0,FG_LINE_SOLID,lr_line_2),
        fg_drawline(FG_WHITE,FG_MODE_SET,~0,FG_LINE_SOLID,ur_line_2),
}


// file  basictile hpp
// V5 2 17 07 90 09 35 PK
#ifndef BASICTILE_HPP
#define BASICTILE_HPP

#include "digvalue hpp"

class basictile   public digvalue {

int times,
char * bitmap,
  public

    void basic_pdraw(),

    basictile(unsigned x, unsigned y, int id, float m, float raw_value,
      unsigned state, int extent_onoff)    ( x, y, id, m, raw_value, state,
                                                          extent_onoff) {

        items=1,
        offset=15,

        max_extent_box[0]=x_center-(fg_box_width(fg charbox))-offset-deadband,
        max_extent_box[1]=y_center-offset-deadband,
        max_extent_box[2]=x_center+(7*(fg_box_width(fg charbox)))+offset
                                                              +deadband,
        max_extent_box[3]=y_center+(items*(fg_box_height(fg charbox)))+offset
                                                                +deadband,

//      pixel_buffer_length=(sizeof(fg_color_t)*fg_box_area(max_extent_box)),
        size=sizeof(fg_color_t),
//      pixel_buffer=farcalloc (pixel_buffer_length,size),
//      pixel_buffer=malloc (pixel_buffer_length),
        file_rand=rand(),
        itoa (file_rand,sub_name,10),
        total_name[0]='b',
        total_name[1]='t',
        q=2,
        while (sub_name[q-2]!='\0'){
                total_name[q]=sub_name[q-2],
```

```
        msm_showcursor(),
}

void bmal  pextent_perase() {
        msm_hidecursor(),
        fg_drawbox (extent_color,FG_MODE_XOR,~0,FG_LINE_MEDIUM_DASHED
                                                ,extent_box,fg_display box),
        msm_showcursor(),
}

void bmal  panimate() {}

void bmal  pdraw() {
        msm_nidecursor(),

        x_min_extn=x_center-1,
        y_min_extn=y_center-1,
        x_max_extn=x_center+1+abs(co_ord[0]-co_ord[2]),
        y_max_extn=y_center+1+abs(co_ord[1]-co_ord[3]),

        extent_box [FG_X1]=x_min_extn,
        extent_box [FG_Y1]=y_min_extn,
        extent_box [FG_X2]=x_max_extn,
        extent_box [FG_Y2]=y_max_extn,

        screen_cut [FG_X1]=x_center,
        screen_cut [FG_Y1]=y_center,
        screen_cut [FG_X2]=x_center+abs(co_ord[0]-co_ord[2]),
        screen_cut [FG_Y2]=y_center+abs(co_ord[1]-co_ord[3]),

        screen_ocut [FG_X1]=x_center-1,
        screen_ocut [FG_Y1]=y_center-1,
        screen_ocut [FG_X2]=x_center+abs(co_ord[0]-co_ord[2])+1,
        screen_ocut [FG_Y2]=y_center+abs(co_ord[1]-co_ord[3])+1,

        bitmap=total_name_in,
        fptr=fopen(bitmap,"rb"),
        byte_length=(filesize(bitmap))*size,   // bytes
        pixel_in=malloc (byte_length),
        fread (pixel_in,bytez,byte_length,fptr),
        fclose (fptr),

        bitmap=total_name_out,
        pixel_buffer_length=(sizeof(fg_color_t) * fg_box_area(screen_cut)),
        pixel_buffer=malloc (pixel_buffer_length),
        fg_readbox (screen_cut,pixel_buffer),
        bytez=(sizeof(int))/2,
        fptr=fopen(bitmap,"wb"),
        fwrite (pixel_buffer,bytez,pixel_buffer_length,fptr),
        fclose (fptr),
        free (pixel_buffer),

        fg_writebox (screen_cut,pixel_in),
        free (pixel_in),

        msm_showcursor(),
```

```
        int co_ord[4],                          // bitmap file co-ordinate info
        int byte_length,                        // length of dynamic pixel buffers
        fg_color_t far *pixel_in,
        fg_color_t far *pixel_out,
        fg_box_t screen_cut,screen_ocut,

        static char mess1[19],
        static char mess2[19],

        float ll_tx,ur_tx,ll_ty,ur_ty,   // text window coords
        long timer,                       // message timer

public
        void pdraw(),
        void panimate(),
        void pextent_pgen(),
        void pextent_perase(),
        void pextent_pdraw(),


        bmai (unsigned x,unsigned y, int id, int extent_onoff)    () {

        x_center=x,
        y_center=y,
        identifier=id,                          // cadshape id
        cad_type=12,                            // non-active cad type
        status_extn=extent_onoff,
        bytez=(sizeof(int))/2,
        extent_overlap=0,
        color=15,
        size=sizeof(fg_color_t),

        ll_tx=10, ur_tx=ll_tx+20*fg_box_width(fg charbox),
        ll_ty=02, ur_ty=ll_ty+fg_box_height(fg charbox),

        mess1[0]='z',
        mess1[1]='i',
        mess1[2]='p',
        mess1[3]=' ',
        mess1[4]='f',
        mess1[5]='i',
        mess1[6]='l',
        mess1[7]='e',
        mess1[8]=' ',
        mess1[9]='e',
        mess1[10]='r',
        mess1[11]='r',
        mess1[12]='o',
        mess1[13]='r',
        mess1[14]=' ',
        mess1[15]=' ',
        mess1[16]=' ',
        mess1[17]=' ',
        mess1[18]='\0',

        mess2[0]='b',
```

```
                source[alpha+2]='n',
                source[alpha+3]='u',
                source[alpha+4]='l',
                source[alpha+5]='\0',

                // Spawn a child process
//              if ((spawnlp(0,"pkunzip exe","-o","b zip","bitmap1 zmp",">nul"
//                                                              ,NULL))==-1){ }

                // Call command via Command com
                system (source),
        }

        bitmap="bitmap1 zmp",
        size=sizeof(fg_color_t),

        if ((fptr=fopen(bitmap,"rb"))==NULL)
                {
                // Open a text window
                window window1,                         // text window
                window1 open(ll_tx,ll_ty,ur_tx,ur_ty,FG_BLACK,FG_LIGHT_WHITE),
                window1 text(mess2,FG_LIGHT_RED),
                for (timer=0,timer<80000,timer++){}
                window1 erase(),

                fclose (fptr),
                window1 close(),

                // Generate a dummy bitmap
                co_ord[0]=100,
                co_ord[1]=100,
                co_ord[2]=199,
                co_ord[3]=199,

                byte_length=co_ord[2]-co_ord[0]+1,        // bytes
                byte_length*=co_ord[3]-co_ord[1]+1,
                byte_length*=sizeof(fg_coord_t),
                pixel_in=calloc (byte_length,size),
                memset (pixel_in,0x55,byte_length),

                }

        else    {
                // Read in bitmap info
                byte_length=(filesize(bitmap)-8)*size,   // bytes
                pixel_in=malloc (byte_length),

                fread (co_ord,sizeof(int),4,fptr),
                fread (pixel_in,bytez,byte_length,fptr),
                fclose (fptr),

                // Delete  zmp uncompressed bitmap data file
                source[0]='d',
                source[1]='e',
                source[2]='l',
                source[3]=' ',
```

```
                                source[2]='1',
                                source[3]=' ',
                                for (alpha=4,*bitmap!='\0',alpha++){
                                source[alpha]=*bitmap++,}
                                source[alpha]='\0',
                                system (source),
                        }
                }
},

#endif BMA1_HPP
// file  button cpp
// Vr5 1 29/06/90 08 45 PK
// Second level inheritance
// Zortech & Flash Graphics
#include "button hpp"
#include <msmouse h>
#include <fg h>

void button  pextent_pgen() {

          extents_sim[0]=x_center-(a*mul)-width-boundary,
          extents_sim[1]=y_center-(a*mul)-height-boundary,
          extents_sim[2]=x_center+(a*mul)+width+boundary,
          extents_sim[3]=y_center+(a*mul)+height+boundary,
}

void button  pextent_pdraw() {
          msm_hidecursor(),

          x_min_extn=x_center-(a*mul)-width-boundary,
          y_min_extn=y_center-(a*mul)-height-boundary,
          x_max_extn=x_center+(a*mul)+width+boundary,
          y_max_extn=y_center+(a*mul)+height+boundary,

          extent_box[FG_X1]=x_min_extn,
          extent_box[FG_Y1]=y_min_extn,
          extent_box[FG_X2]=x_max_extn,
          extent_box[FG_Y2]=y_max_extn,

          if (extent_overlap==0) {extent_color=color,}
          else {extent_color=color-2,}
          fg_drawbox(extent_color,FG_MODE_XOR,~0,FG_LINE_MEDIUM_DASHED
                                                    ,extent_box,fg_display box),
          msm_showcursor(),
          }

void button  pdraw() {
          msm_hidecursor(),

          small_box [FG_X1]=x_center-(a*mul)-width,
          small_box [FG_Y1]=y_center-(a*mul)-height,
          small_box [FG_X2]=x_center+(a*mul)+width,
          small_box [FG_Y2]=y_center+(a*mul)+height,

          small_fill_box [FG_X1]=small_box [FG_X1]+2,
```

```
tx_status=small_box[FG_X1]+a*mul,
ty_status=small_box[FG_Y1]+a*mul,
fg_drawbox (FG_WHITE,FG_MODE_SET,~0,FG_LINE_SOLID,outer_box_1
                                                        ,fg_displaybox),
if ( status==0)
      {
          for (i=4,i>=0,i--){
                    shadow_box_var[FG_X1]=outer_box_2[FG_X1]-i,
                    shadow_box_var[FG_Y1]=outer_box_2[FG_Y1]-i,
                    shadow_box_var[FG_X2]=outer_box_2[FG_X2]+i,
                    shadow_box_var[FG_Y2]=outer_box_2[FG_Y2]+i,
                    fg_drawbox(FG_BLACK,FG_MODE_SET,~0,FG_LINE_SOLID
                                        ,shadow_box_var,fg_displaybox) ,
                    i--,
                    }
          for (i=8,i>=0,i--){
                    shadow_box_var[FG_X1]=small_box[FG_X1]-i,
                    shadow_box_var[FG_Y1]=small_box[FG_Y1]-i,
                    shadow_box_var[FG_X2]=small_box[FG_X2]+i,
                    shadow_box_var[FG_Y2]=small_box[FG_Y2]+i,
                    j=8-i,
                    fg_drawbox(j,FG_MODE_SET,~0,FG_LINE_SOLID
                                        ,shadow_box_var,fg_displaybox),
                    }
          fg_fillbox (scolor,FG_MODE_SET,~0,small_box),
          fg_puts(scolor,FG_MODE_XOR,~0,FG_ROT0,tx_status,ty_status
                                                  ,text0,fg_displaybox),
 //       fg_drawbox (FG_WHITE,FG_MODE_SET,~0,FG_LINE_SOLID
                                          ,outer_box_1,fg_displaybox),
      }
if (status==1)
      {
//        fg_drawbox (FG_BLACK,FG_MODE_SET,~0,FG_LINE_SOLID
                                          ,outer_box_1,fg_displaybox),
          for (i=4,i>=0,i--){
                    shadow_box_var[FG_X1]=outer_box_2[FG_X1]-i,
                    shadow_box_var[FG_Y1]=outer_box_2[FG_Y1]-i,
                    shadow_box_var[FG_X2]=outer_box_2[FG_X2]+i,
                    shadow_box_var[FG_Y2]=outer_box_2[FG_Y2]+i,
                    fg_drawbox(i,FG_MODE_SET,~0,FG_LINE_SOLID
                                        ,shadow_box_var,fg_displaybox),
                    i--,
                    }
                    for (i=8,i>=0,i--){
                       shadow_box_var[FG_X1]=small_fill_box[FG_X1]-i,
                       shadow_box_var[FG_Y1]=small_fill_box[FG_Y1]-i,
                       shadow_box_var[FG_X2]=small_fill_box[FG_X2]+i,
                       shadow_box_var[FG_Y2]=small_fill_box[FG_Y2]+i,
                              j=8-i,
                       fg_drawbox(j,FG_MODE_SET,~0,FG_LINE_SOLID
                                        ,shadow_box_var,fg_displaybox),
                              }
                       //fg_drawline(FG_BLACK,FG_MODE_SET,~0
                                              ,FG_LINE_SOLID,ll_line),
                       //fg_drawline(FG_BLACK,FG_MODE_SET,~0
```

```
                    }
                fg_fillbox (scolor,FG_MODE_SET,~0,small_box),
                fg_puts(scolor,FG_MODE_XOR,~0,FG_ROT0,tx_status
                                ,ty_status,text0,fg_displaybox),
//          fg_drawbox (FG_WHITE,FG_MODE_SET,~0,FG_LINE_SOLID
                                ,outer_box_1,fg_displaybox),
                }
if (status==1)
                {
//          fg_drawbox (FG_BLACK,FG_MODE_SET,~0,FG_LINE_SOLID
                                ,outer_box_1,fg_displaybox),
                for (i=4,i>=0,i--){
                   shadow_box_var[FG_X1]=outer_box_2[FG_X1]-i,
                   shadow_box_var[FG_Y1]=outer_box_2[FG_Y1]-i,
                   shadow_box_var[FG_X2]=outer_box_2[FG_X2]+i,
                   shadow_box_var[FG_Y2]=outer_box_2[FG_Y2]+i,
                   k=i+3,
                   fg_drawbox(k,FG_MODE_SET,~0,FG_LINE_SOLID
                                ,shadow_box_var,fg_displaybox),
                        i--,
                        for (j=0,j<7000,j++){}
                        }
                for (i=8,i>=0,i--){
                   shadow_box_var[FG_X1]=small_fill_box[FG_X1]-i,
                   shadow_box_var[FG_Y1]=small_fill_box[FG_Y1]-i,
                   shadow_box_var[FG_X2]=small_fill_box[FG_X2]+i,
                   shadow_box_var[FG_Y2]=small_fill_box[FG_Y2]+i,
                   j=8-i,
                   fg_drawbox(j,FG_MODE_SET,~0,FG_LINE_SOLID
                                ,shadow_box_var,fg_displaybox),
                        }
                //fg_drawline(FG_BLACK,FG_MODE_SET,~0,FG_LINE_SOLID
                                                        ,ll_line),
                //fg_drawline(FG_BLACK,FG_MODE_SET,~0,FG_LINE_SOLID
                                                        ,ul_line),
                //fg_drawline(FG_BLACK,FG_MODE_SET,~0,FG_LINE_SOLID
                                                        ,lr_line),
                //fg_drawline(FG_BLACK,FG_MODE_SET,~0,FG_LINE_SOLID
                                                        ,ur_line),
                fg_fillbox (scolor,FG_MODE_SET,~0,small_fill_box),
                fg_puts (scolor,FG_MODE_XOR,~0,FG_ROT0,tx_status
                                ,ty_status,text1,fg_displaybox),
            }
          // Click
          for(sound_count=0,sound_count<10,sound_count++) sound_click(),
          msm_showcursor(),
          }
          else {}
}

void button  perase() {
     msm_hidecursor(),
     fg_drawbox (FG_BLACK,FG_MODE_SET,~0,FG_LINE_SOLID,outer_box_1
                                                ,fg_displaybox),
     fg_drawbox (FG_BLACK,FG_MODE_SET,~0,FG_LINE_SOLID,outer_box_2
                                                ,fg_displaybox),
```

```
    button(unsigned x, unsigned y, int id, float m, float raw_value, unsigned
          state, int extent_onoff)    (x,y,id,m,raw_value,state,extent_onoff) {

          cad_type=2,                              // DOP type
          x_input=x, y_input=y,                   // initialise input
          width=2*fg_box_width(fg charbox),// minimum width
          height=0 5*width,                                  // minimum height
          max_mul=10,                              // maximum expansion factor
          boundary=13,

          // text0 & text1
          text0[0]=' ',
          text0[1]='o',
          text0[2]='f',
          text0[3]='f',
          text0[4]=' ',
          text0[5]='\0',

          text1[0]=' ',
          text1[1]='o',
          text1[2]='n',
          text1[3]=' ',
          text1[4]=' ',
          text1[5]='\0',

          max_extent_box[0]=x_center-(a*max_mul)-width-boundary,
          max_extent_box[1]=y_center-(a*max_mul)-height-boundary,
          max_extent_box[2]=x_center+(a*max_mul)+width+boundary,
          max_extent_box[3]=y_center+(a*max_mul)+height+boundary,

//        pixel_buffer_length=(sizeof(fg_color_t)*fg_box_area(max_extent_box)),
          size=sizeof(fg_color_t),
//        pixel_buffer=farcalloc (pixel_buffer_length,size),
//        pixel_buffer=malloc (pixel_buffer_length),
          file_rand=rand(),
          itoa (file_rand,sub_name,10),
          total_name[0]='b',
          total_name[1]='u',
          q=2,
          while (sub_name[q-2]!='\0'){
                    total_name[q]=sub_name[q-2],
                    q++,
                    }
          total_name[q]=' ',
          total_name[q+1]='s',
          total_name[q+2]='a',
          total_name[q+3]='v',
          total_name[q+4]='\0',

          bitmap=total_name,
     }

     void perase()
     ~button() {
//        free (pixel_buffer),
          perase(),
```

```
// file  buttontile.cpp
// Vr5.1 16/07/90 11:45 PK
// Second level inheritance
// Zortech & Flash Graphics
#include "buttontile.hpp"
#include <msmouse.h>
#include <fg.h>

void buttontile::pdraw() {
          msm_hidecursor();

          small_box[FG_X1]=x_center-(a*mul)-width,
          small_box[FG_Y1]=y_center-(a*mul)-height,
          small_box[FG_X2]=x_center+(a*mul)+width,
          small_box[FG_Y2]=y_center+(a*mul)+height,

          outer_box_1[FG_X1]=small_box[FG_X1]-boundary,
          outer_box_1[FG_Y1]=small_box[FG_Y1]-boundary,
          outer_box_1[FG_X2]=small_box[FG_X2]+boundary,
          outer_box_1[FG_Y2]=small_box[FG_Y2]+boundary,

          ll_line[FG_X1]=small_box[FG_X1],
          ll_line[FG_Y1]=small_box[FG_Y1],
          ll_line[FG_X2]=small_box[FG_X1],
          ll_line[FG_Y2]=small_box[FG_Y2],

          ll_line_2[FG_X1]=small_box[FG_X1]+1,
          ll_line_2[FG_Y1]=small_box[FG_Y1]+1,
          ll_line_2[FG_X2]=small_box[FG_X1]+1,
          ll_line_2[FG_Y2]=small_box[FG_Y2]-1,

          ul_line[FG_X1]=small_box[FG_X1],
          ul_line[FG_Y1]=small_box[FG_Y2],
          ul_line[FG_X2]=small_box[FG_X2],
          ul_line[FG_Y2]=small_box[FG_Y2],

          ul_line_2[FG_X1]=small_box[FG_X1]+1,
          ul_line_2[FG_Y1]=small_box[FG_Y2]-1,
          ul_line_2[FG_X2]=small_box[FG_X2]-1,
          ul_line_2[FG_Y2]=small_box[FG_Y2]-1,

          lr_line[FG_X1]=small_box[FG_X2],
          lr_line[FG_Y1]=small_box[FG_Y2],
          lr_line[FG_X2]=small_box[FG_X2],
          lr_line[FG_Y2]=small_box[FG_Y1],

          lr_line_2[FG_X1]=small_box[FG_X2]-1,
          lr_line_2[FG_Y1]=small_box[FG_Y2]-1,
          lr_line_2[FG_X2]=small_box[FG_X2]-1,
          lr_line_2[FG_Y2]=small_box[FG_Y1]+1,

          ur_line[FG_X1]=small_box[FG_X2],
          ur_line[FG_Y1]=small_box[FG_Y1],
          ur_line[FG_X2]=small_box[FG_X1],
          ur_line[FG_Y2]=small_box[FG_Y1],
```

```
void buttontile  digital_input() {

          // test if x_input,y_input is within input range
          small_box [FG_X1]=x_center-(a*mul)-width,
          small_box [FG_Y1]=y_center-(a*mul)-height,
          small_box [FG_X2]=x_center+(a*mul)+width,
          small_box [FG_Y2]=y_center+(a*mul)+height,

          input_box[FG_X1]=small_box[FG_X1],
          input_box[FG_Y1]=small_box[FG_Y1],
          input_box[FG_X2]=small_box[FG_X2],
          input_box[FG_Y2]=small_box[FG_Y2],

          // test for mouse click
          //wiat for click
          inside=fg_pt_inbox(input_box,x_input,y_input),
          if (inside'=0){
                    msm_hidecursor(),

                    // Invert status and scolor
                    status='status,
                    scolor=10+(status*2),

                    if ( status==0)
                      {
                      fg_fillbox (FG_MAGENTA,FG_MODE_SET,~0,small_box),
                      fg_drawline(FG_WHITE,FG_MODE_SET,~0,FG_LINE_SOLID,ll_line),
                      fg_drawline(FG_WHITE,FG_MODE_SET,~0,FG_LINE_SOLID,ul_line),
                      fg_drawline(FG_BLACK,FG_MODE_SET,~0,FG_LINE_SOLID,lr_line),
                      fg_drawline(FG_BLACK,FG_MODE_SET,~0,FG_LINE_SOLID,ur_line),
                      fg_puts (scolor,FG_MODE_SET,~0,FG_ROT0,tx_status,ty_status
                                                      ,text0,fg_displaybox),
                          }
                    if (status==1)
                      {
                      fg_fillbox (FG_CYAN,FG_MODE_SET,~0,small_box),
                      fg_drawline(FG_BLACK,FG_MODE_SET,~0,FG_LINE_SOLID,ll_line),
                      fg_drawline(FG_BLACK,FG_MODE_SET,~0,FG_LINE_SOLID,ul_line),
                      fg_drawline(FG_WHITE,FG_MODE_SET,~0,FG_LINE_SOLID,lr_line),
                      fg_drawline(FG_WHITE,FG_MODE_SET,~0,FG_LINE_SOLID,ur_line),
                      fg_puts (scolor,FG_MODE_SET,~0,FG_ROT0,tx_status,ty_status
                                                      ,text1,fg_displaybox),
                          }

                    // Click
                    for(sound_count=0,sound_count<10;sound_count++) sound_click(),
                    msm_showcursor(),
                    }
                    else {}
}

void buttontile  perase() {
     msm_hidecursor(),
     fg_fillbox (FG_BLACK,FG_MODE_SET,~0,outer_box_1),
```

```
        text0[3]='f',
        text0[4]=' ',
        text0[5]='\0',

        text1[0]=' ',
        text1[1]='o',
        text1[2]='n',
        text1[3]=' ',
        text1[4]=' ',
        text1[5]='\0',

        max_extent_box[0]=x_center-(a*max_mul)-width-boundary,
        max_extent_box[1]=y_center-(a*max_mul)-height-boundary,
        max_extent_box[2]=x_center+(a*max_mul)+width+boundary,
        max_extent_box[3]=y_center+(a*max_mul)+height+boundary,

        size=sizeof(fg_color_t),
        file_rand=rand(),
        itoa (file_rand,sub_name,10),
        total_name[0]='b',
        total_name[1]='t',
        q=2,
        while (sub_name[q-2]!='\0'){
                total_name[q]=sub_name[q-2],
                q++,
                }
        total_name[q]=' ',
        total_name[q+1]='s',
        total_name[q+2]='a',
        total_name[q+3]='v',
        total_name[q+4]='\0',
        bitmap=total_name,
    }
    void perase(),
    ~buttontile() {
        perase(),
 }
},
#endif BUTTONTILE_HPP


// file  cadshape hpp Vr 5 4 23/06/90 13 45
#ifndef CADSHAPE_HPP
#define CADSHAPE_HPP

#include <time h>
#include <stdio h>
#include <fg h>
#include <dos h>
#include <stdlib h>

class cadshape {
        friend class template,
  protected    // so derived classes have access
    static char source[21],
```

```
// virtual functions must have SOME
// definition in the base class, even
// if it's just empty
virtual ~cadshape() {}
virtual void odraw() {}
virtual void animate() {}
virtual void erase() {}
virtual void extent_gen() {}
virtual void extent_draw() {}
virtual void extent_erase() {}
virtual void analog_input() {}
virtual void digital_input() {}

cadshape() {}

void idmod(int new_id){
     identifier=new_id,
     }

unsigned int idinq(){
     return identifier,
     }

unsigned * getcoords(){
     coords[0]=x_center,
     coords[1]=y_center,
     return coords,
     }

 void extent_set(int extenth_onoff) {
       status_extn=extenth_onoff,
 }

 int extent_view() {
      return status_extn,
 }

 int extent_test(float* extn_sim_ptr) {
               // test extents
               xmintx=*extn_sim_ptr,
               ymintx=*(extn_sim_ptr+1),
               xmaxtx=*(extn_sim_ptr+2),
               ymaxtx=*(extn_sim_ptr+3),

               overlap=0,
               ytest=0,
               xmintx_outside=0,
               xmaxtx_outside=0,
               ymintx_outside=0,
               ymaxtx_outside=0,


               if ((xmintx<(x_min_extn))||(xmintx>(x_max_extn)))
                        { if ((x_max_extn<(xmintx))||(x_max_extn>(xmaxtx)))
                                 { //printf ("-a"),
```

```
                xmaxtx=*(extn_sim_ptr+2),
                ymaxtx=*(extn_sim_ptr+3),
                overlap=0,

                if((xmintx<fg_displaybox[FG_X1])||
                                        (xmintx>fg_displaybox[FG_X2])||
                                        (xmaxtx<fg_disp laybox[FG_X1])||
                                        (xmaxtx>fg_displaybox[FG_X2]))
                  overlap=1,

                if (overlap==0){
                if ((ymintx<(fg_displaybox[FG_Y1]+25))||
                                        (ymintx>fg_displaybox[FG_Y2])||
                                        (ymaxtx<(fg_displaybox[FG_Y1]+25))||
                                        (ymaxtx>fg_displaybox[FG_Y2]))
                  overlap=1,}
        return overlap,
}

float*  pextent_gen(int  mul_switch,float   m_sim,   unsigned   new_x_sim,
                                                    unsigned new_y_sim) {
        extents_sim_ptr=&extents_sim[0],
        if (mul_switch'=0){
                mul_saved=mul,
                mul=m_sim+mul,
                }    // for pre-expand test
        else{
                x_saved=x_center,
                y_saved=y_center,
                x_center = new_x_sim,
                y_center = new_y_sim,
                }
        pextent_pgen(),

        if (mul_switch'=0){
                mul=mul_saved,
                }    // for pre-expand test
        else{
                x_center=x_saved,
                y_center=y_saved,
                }
        return extents_sim_ptr,
 }

int expandp(float m) {
        mul=m+mul,
        mul_break=0,
        if (mul>=max_mul){mul=max_mul,mul_break=1,}
        if (mul<=min_mul){mul=min_mul,mul_break=1,}
        pdraw(),
        return mul_break,
}

void modulatep(float raw_value, int state, time_t time_stamp, long
                                                                run_time) {
        status=state,
```

```
    }
},
#endif CADSHAPE_HPP

// file  chemical cpp
// from animvol
// Vr 1 00   06 05 90 21 00 PK
// Stores background behind object in dynamic buffer
#include "chemical hpp"
#include <msmouse h>
#include <stdlib h>
#include <fg h>

void chemical  pdraw() {
     msm_hidecursor(),

     anim_sqr_line_one[FG_X1]=x_center,
     anim_sqr_line_one[FG_Y1]=y_center,
     anim_sqr_line_one[FG_X2]=x_center,
     anim_sqr_line_one[FG_Y2]=y_center+height+(a*mul),

     anim_sqr_line_two[FG_X1]=x_center+width+(a*mul),
     anim_sqr_line_two[FG_Y1]=y_center,
     anim_sqr_line_two[FG_X2]=x_center+width+(a*mul),
     anim_sqr_line_two[FG_Y2]=y_center+height+(a*mul),

     x_min_extn=x_center,
     y_min_extn=y_center-(y_radius_const+a*mul/10)-4,
     x_max_extn=x_center+width+(a*mul)+2,
     y_max_extn=y_center+height+(a*mul)+2+(y_radius_const+a*mul/10),

     background_box[FG_X1]=x_min_extn,
     background_box[FG_Y1]=y_min_extn,
     background_box[FG_X2]=x_max_extn,
     background_box[FG_Y2]=y_max_extn,

          pixel_buffer_length=(sizeof(fg_color_t)*fg_box_area(background_box)),
          pixel_buffer=malloc (pixel_buffer_length),
          fg_readbox (background_box,pixel_buffer),
          bytez=(sizeof(int))/2,
          bitmap=total_name,
          fptr=fopen (bitmap,"wb"),
          fwrite (pixel_buffer,bytez,pixel_buffer_length,fptr),
          fclose (fptr),
          free (pixel_buffer),

     fg_fillbox (fgcolor,FG_MODE_SET,~0,background_box),

     fg_drawline(color,FG_MODE_SET,~0,FG_LINE_SOLID,anim_sqr_line_one),
     fg_drawline(color,FG_MODE_SET,~0,FG_LINE_SOLID,anim_sqr_line_two),

     x_radius=(width+a*mul-2)/2+1,
     y_radius=y_radius_const+a*mul/10,
     ang1=1800,ang2=3600,
     x_ell_center=x_center+(width+(a*mul))/2,
```

```
    // fill  to  new  level  with  status colour

    x_fill=x_ell_center,
    y_fill=y_ell_center_level-y_radius-
    (y_ell_center_level-prev_y_ell_center_level)/2,
    scolor=8+status,
    angl=1800,ang2=3600,

    if (status'=previous_status){
        // Remove second half  of  old
        //level ellipse
        fg_drawellipse (FG_BLACK,FG_MODE_SET,~0,x_ell_center
                  ,prev_y_ell_center_level,x_radius,y_radius,angl,ang2
                                                    ,fg_displaybox),
        fg_drawline(color,FG_MODE_SET,~0,FG_LINE_SOLID,
                                                  anim_sqr_line_one),
        fg_drawline(color,FG_MODE_SET,~0,FG_LINE_SOLID,
                                                  anim_sqr_line_two)
        fg_fill (x_fill,y_fill,FG_BLACK,color),

        fg_fill (x_fill,y_fill,scolor,color),
        }

    else {
        fg_drawline(color,FG_MODE_SET,~0,FG_LINE_SOLID,
                                                    anim_sqr_line_one),
        fg_drawline(color,FG_MODE_SET,~0,FG_LINE_SOLID,
                                                    anim_sqr_line_two),
        fg_fill (x_fill,y_fill,scolor,color),

        // Remove second half  of  old level ellipse
        fg_drawellipse (scolor,FG_MODE_SET,~0,x_ell_center
                ,prev_y_ell_center_level,x_radius,y_radius,angl,ang2,
                                                      fg_displaybox),
        fg_drawline(color,FG_MODE_SET,,FG_LINE_SOLID,
                                                  anim_sqr_line_one),
        fg_drawline(color,FG_MODE_SET,~0,FG_LINE_SOLID,
                                                  anim_sqr_line_two),
        }
    }

else {// value < previous_value

    // draw half new ellipse
    angl=1800,ang2=3600,
    fg_drawellipse (color,FG_MODE_SET,~0,x_ell_center,
              y_ell_center_level,x_radius,y_radius,angl,ang2
                                                    ,fg_displaybox),


    // fill  with  black  between  higher previous level and
    //  lower new level
```

```
// file  chemical hpp
// Vr 1 00 06/05/90 20 00 PK
#ifndef CHEMICAL_HPP
#define CHEMICAL_HPP

#include "animvol hpp"

class chemical   public animvol {
  char * bitmap,
  public

     void panimate(),
     void pdraw(),

     chemical(unsigned x, unsigned y, int id, float m, float raw_value,
                                        unsigned state, int extent_onoff)
                                   (x, y, id, m, raw_value, state, extent_onoff){
          cad_type=11,                 // Multi-DIP AIP type
          file_rand=rand(),
          itoa (file_rand,sub_name,10),
          total_name[0]='c',
          total_name[1]='h',
          q=2,
          while (sub_name[q-2]!='\0'){
                    total_name[q]=sub_name[q-2],
                    q++,
                    }
          total_name[q]=' ',
          total_name[q+1]='s',
          total_name[q+2]='a',
          total_name[q+3]='v',
          total_name[q+4]='\0',

          bitmap=total_name,
          }

     ~chemical() {
          perase(),
          }
},

#endif CHEMICAL_HPP


// Design cpp
// by PK for MSc project
// V9 00 26/11/90 18 15  Include Simulation, Analog and Digital input

#include          <bios h>
#include          <stdlib h>
#include          <fg h>
#include          <msmouse h>
#include          <time h>
```

```
  " Pump",        ANIMPIE,
  " Pie",         PIE,
  " Square",      SQUARE,
  " Led",         ANIMSQR,
  " Text",        DUALTEXT,
  " Status",      STATUSTEXT,
  " Status3D",    STATUS3D,
  " Func Menu",  EXIT_6,
  "",-1           /* end */
},
msmenu mouse6(sub_menu_6),

enum {ANIMPLOT,BITMAP1,GAS,EXIT_7},
struct menu_s sub_menu_7[] = {
   " Plot",        ANIMPLOT,
   " Bitmap",      BITMAP1,
   " Gas",         GAS,
   " Func Menu",  EXIT_7,
   "",-1           /* end */
},
msmenu mouse7(sub_menu_7),

// associate unique integers with the following
enum   {NEW_ICON_1,NEW_ICON_2,NEW_ICON_3,NEW_ICON_4,MOVE,E_MOVE,EXPAND,WIPE,
       SHOW,DELETE,CLEAR,INFORMATION,SIMULATE,A_INPUT,D_INPUT,EXIT_DOS},

struct menu_s my_menu[] = {
   " New Icon A",    NEW_ICON_1,
   " New Icon B",    NEW_ICON_2,
   " New Icon C",    NEW_ICON_3,
   " New Icon D",    NEW_ICON_4,
   " Move",        MOVE,
   " E_Move",      E_MOVE,
   " Size",        EXPAND,
   " Delete",      DELETE,
   " Wipe",        WIPE,
   " Clear",       CLEAR,
   " Show",        SHOW,
   " Info",        INFORMATION,
   " Animate",     SIMULATE,
   " A_Input",     A_INPUT,
   " D_Input",     D_INPUT,
   " Dos",         EXIT_DOS,
   "", -1                /* end marker */
},
fgnd foreground(0,0),

main (argc,argv)
int argc,
char* argv[],{
static char mess1[]="OOPS Foreground Designer Vr 9 0 by P  Kiernan",
static char mess4[]="Extent overlap     ",
static char mess5[]="No extent overlap ",
static char mess6[]="Ordered overlap",
static char mess2[]="System too slow",
static char mess3[]="System wait     ",
```

```
unsigned minx,              // mouse cursor coords
unsigned miny,
unsigned maxx,
unsigned maxy,

// linked-in external routines
void menugns1(),                    // fill foreground (foreground) with cadshapes
void menugns2(),                    // fill foreground (foreground) with cadshapes
void menugns3(),                    // fill foreground (foreground) with cadshapes
void menugns4(),                    // fill foreground (foreground) with cadsnapes
void fg_init_palette(),            // flash graphics and palette initialisation
void wipesrc(char *),               // destroys calling program
void bdisplay (char *,char*),     // displays background file
void selectbg(),                    // allows selection of zip and zpi mimic file

void info_display(template *, cadshape *, fg_coord_t, fg_coord_t),
int info_edit(template *, cadshape *, fg_coord_t, fg_coord_t),

          srce=argv[0],
          wipesrc(srce),

          msmenu mouse(my_menu),
          fg_init_palette(),                  // init graphics

          minx=fg_displaybox[FG_X1]+3,
          miny=fg_displaybox[FG_Y1],
          maxx=fg_displaybox[FG_X2]-3,      // < max to maintain cursor visibility
          maxy=fg_displaybox[FG_Y2]-3,

          msm_setareax(minx,maxx),
          msm_setareay(miny,maxy),

          // Establish message and coordinate counter windows coords ,
          ll_ty = 02, ur_ty=ll_ty+fg_box_height(fg charbox),

          ll_tx1= 10, ur_tx1= 629,
          ll_tx2=fg_displaybox[FG_X2]-125,
          ur_tx2= ll_tx2+5*fg_box_width(fg charbox),
          ll_tx3=fg_displaybox[FG_X2]-80,
          ur_tx3= ll_tx3+5*fg_box_width(fg charbox),

          // Introductory message
          window window1,
          window1 open(ll_tx1,ll_ty,ur_tx1,ur_ty,FG_LIGHT_BLUE,FG_LIGHT_WHITE),
          window1 text(mess1,FG_LIGHT_WHITE),
          for (long wait=0,wait<150000,wait++){}
          window1 erase(),
          window1 close(),

          // Message window coords ,
          ur_tx1= ll_tx1+19*fg_box_width(fg charbox),

          // display background mimic
          if (argc<=2)
                    selectbg(),
          else
```

```
                                        extent_result=foreground
                                          extent_test(extents_sim_ptr,mv),
                                        extent_result+=(mv->boundary_test
                                                    (extents_sim_ptr)),
                                        if (extent_result<1){
                                                x_saved=x,
                                                y_saved=y,
                                                mv->movep(x_saved,y_saved),
                                                mv->perase(),
                                                }
                                    }           // drag symbol
                            msm_showcursor(),
                            mv->movep(x_saved,y_saved),
                            }
        }
        mouse default_cursor(),
        break,

case INFORMATION
        cadshape * infoets =foreground test(),
        if (infoets !=(cadshape *)0){
                mouse_button=0,
                while (mouse_button<2){
                        mouse cross_cursor(),
                        mouse wait_left_pressed(&x,&y),
                        mouse translate_coords(&x,&y),
                        infoets = foreground nearest(x,y),
                        infoets->perase(), // pick it up
                        for (int i=0,i<10000,i++){}
                        infoets->pdraw(),
                        mouse_button=msm_getstatus(&x,&y),
                        mouse translate_coords(&x,&y),
                }
                template template1(infoets,x,y),
                void * template_ptr,
                template_ptr=&template1,
                info_display(template_ptr,infoets,x,y),

                mouse default_cursor(),
                exit_edit=0,
                mouse_button=0,
                while (exit_edit!=1){
                        mouse wait_right_pressed(&x,&y),
                        mouse translate_coords(&x,&y),
                        exit_edit=info_edit(template_ptr,infoets,x,y),
                }
                msm_showcursor(),
        }
        break,

case E_MOVE
        cadshape * ets =foreground test(),
        if (ets !=(cadshape *)0){
                mouse cross_cursor(),
                mouse wait_left_pressed(&x,&y),
                mouse translate_coords(&x,&y),
```

```
                                        window1 erase(),
                                        extent_overlap_status=0,
                                        window1 text(mess5
                                                ,FG_LIGHT_WHITE),
                                        msm_showcursor(),
                                        }
                                    x_saved=x, // no object overlap
                                    y_saved=y,
                                }
                                else   {// extent_result > 0
                                    window1 erase(),
                                    extent_overlap_status=1,
                                    valu=itoa(extent_result,valuout
                                                        ,10),
                                    mess4[15]=valuout[0],
                                    mess4[16]=valuout[1],
                                    window1 text(mess4
                                              ,FG_LIGHT_WHITE),
                                    }

                                    // Coords message
                                    valu_x=itoa  (x_saved,valuout_x,10),
                                    valu_y=itoa  (y_saved,valuout_y,10),
                                    window2 text(valuout_x,FG_LIGHT_WHITE),
                                    window3 text(valuout_y,FG_LIGHT_WHITE),

                                    emv->pextent_movep(x,y,extent_result),
                                    emv->pextent_pdraw(),
                                    emv->pextent_perase(), //   drag symbol
                            }

                            if (extent_overlap_status==1){
                                      x_saved=*coords,
                                      y_saved=*(coords+1),
                                      }
                            emv->perase(),
                            emv->movep(x_saved,y_saved),   // place object
                                                  // at new destination
                    }

                    mouse default_cursor(),
                    // restore background
                    msm_hidecursor(),
                    window1 erase(),
                    window2 erase(),
                    window3.erase(),

                    window1 close(),
                    window2 close(),
                    window3 close(),
                    msm_showcursor(),
                    }
        break,

case SHOW
        if ('clear){
```

```
                                        msm_showcursor(),
                                        }
                                outside_mul_limits=mv->expandp(mul),
                                if (outside_mul_limits>0){mul=0,}
                                        for (int i=1, i<100, i++) {
                                            for (int j=1, j<100,j++){}}
                                        mv->perase(),
                                } // expand shape
                              else
                                { // extent_result > 0
                                window1 erase(),
                                extent_overlap_status=1,
                                valu=itoa(extent_result,valuout,10),
                                mess4[15]=valuout[0],
                                mess4[16]=valuout[1],
                                window1 text(mess4,FG_LIGHT_WHITE),
                                }
                              }
          if (msm_getstatus(&x,&y) & 2){
             msm_hidecursor(),
             mul=mul-1,
             mul_input=1,   // use test mul factor
             extents_sim_ptr=mv->pextent_gen(mul_input,mul,x,y),
             extent_result=foreground extent_test
                                        (extents_sim_ptr,mv),
             extent_result+=(mv->boundary_test(extents_sim_ptr)),

                    if (extent_result<1){
                              if (extent_overlap_status==1){
                                        msm_hidecursor(),
                                        window1 erase(),
                                        extent_overlap_status=0,
                                 window1 text(mess5,FG_LIGHT_WHITE),
                                        msm_showcursor(),
                                        }
                       outside_mul_limits=mv->expandp(mul),
                              if (outside_mul_limits>0){mul=0,}
                              for (int i=1, i<100, i++) {
                                        for (int j=1, j<100, j++){} }
                              mv->perase(),
                    }
                    else     {                  // extent_result > 0
                              window1 erase(),
                              extent_overlap_status=1,
                              valu=itoa (extent_result,valuout,10),
                              mess4[15]=valuout[0],
                              mess4[16]=valuout[1],
                              window1 text(mess4,FG_LIGHT_WHITE),
                              }
                    }
                    msm_showcursor(),
          }
msm_showcursor();
mv->pdraw(),

// restore background
```

```
        valuout[6]= ' ',
        valuout[7]= 'e',
        valuout[8]='r',
        valuout[9]='r',
        valuout[10]='o',
        valuout[11]='r',
        valuout[12]='\0',
        if (percent_error > 0){
                window1 text(valuout,FG_LIGHT_WHITE),
                        for (long_n=120000,long_n>0,long_n--){}
                window1 erase(),
        }
        window1 close(),
        msm_showcursor(),
                break,

case A_INPUT
        cadshape * a_inp =foreground test(),
        if (a_inp !=(cadshape *)0){
                mouse cross_cursor(),
                mouse wait_left_pressed(&x,&y),
                mouse translate_coords(&x,&y),
                a_inp = foreground nearest(x,y),
                mouse default_cursor(),

                while (msm_getstatus(&x,&y) & 1)
                        {mouse translate_coords(&x,&y),
                        mul=1,
                        input_value=a_inp->analog_inputp(x,y),
                        }
        }
        break,

case D_INPUT
        cadshape * d_inp =foreground test(),
        if (d_inp !=(cadshape *)0){
                mouse default_cursor(),
                mouse wait_left_pressed(&x,&y),
                mouse translate_coords(&x,&y),
                d_inp = foreground nearest(x,y),

                while (msm_getstatus(&x,&y) & 1)
                        {mouse translate_coords(&x,&y),
                        mul=1,
                        input_onoff=d_inp->digital_inputp(x,y),
                        for (long_n=40000,long_n>0,long_n--){}
                }
        } break,

case EXIT_DOS
        msm_showcursor(),
//      store foreground to disc
        quit++,
        break,

default
```

```
                    break,
            }
        }
}
            // Erase any  bm* sav constructor generated files left by 'free' calls
            source[0]='d',
            source[1]='e',
            source[2]='l',
            source[3]=' ',
            source[4]='b',
            source[5]='m',
            source[6]='*',
            source[7]=' ',
            source[8]='s',
            source[9]='a',
            source[10]='v',
            source[11]='\0',
            system (source),

            fg_term(),
            msm_term(),}
```

```
x_max_extn=x_center+outer_radius+5*(fg_box_width(fg charbox)),
y_max_extn=y_center+outer_radius+fg_box_height(fg charbox),

extent_box[FG_X1]=x_min_extn,
extent_box[FG_Y1]=y_min_extn,
extent_box[FG_X2]=x_max_extn,
extent_box[FG_Y2]=y_max_extn,

background_box[FG_X1]=x_min_extn,
background_box[FG_Y1]=y_min_extn,
background_box[FG_X2]=x_max_extn,
background_box[FG_Y2]=y_max_extn,

pixel_buffer_length=(sizeof(fg_color_t)*fg_box_area(background_box)),
pixel_buffer=malloc (pixel_buffer_length),
fg_readbox (background_box,pixel_buffer),
bytez=(sizeof(int))/2,
bitmap=total_name,
fptr=fopen (bitmap,"wb"),
fwrite (pixel_buffer,bytez,pixel_buffer_length,fptr),
fclose (fptr),
free (pixel_buffer),

fg_fillbox (fgcolor,FG_MODE_SET,~0,background_box),
fg_drawarc(color,FG_MODE_XOR,~0,x1,y1,inner_radius,angl,ang3
                                                    ,fg_displaybox),
fg_drawarc(color,FG_MODE_XOR,~0,x1,y1,outer_radius,angl,ang3
                                                    ,fg_displaybox),

line [FG_X1]=x1-outer_radius,
line [FG_X2]=x1+outer_radius,
line [FG_Y1]=y1,
line [FG_Y2]=y1,
fg_drawline(color,FG_MODE_XOR,~0,FG_LINE_SOLID,line),

// status colour
scolor=10+(status*2),

// angle proportional to 1->100% value
ang2=((value/100)*1800),
angle=(value/100)*3 1415926,
if (angle>3 1415926) angle=3 1415926,
previous_ang2=ang2,
previous_angle=angle,

line [FG_X1]=x1,
line [FG_X2]=x1+(x_radius*(cos(angle))),
line [FG_Y2]=y1+(x_radius*(sin(angle))),,

// avoid drawing sides of pie with such a small angle
// that xor of adjacent sides occurs

if ((x_radius*(sin(angle))<1) && (x_radius*(sin(angle))>-1)){
          if ((x_radius*(cos(angle)))>0){}
          else {fg_drawline(scolor,FG_MODE_XOR,~0,FG_LINE_SOLID,line),}
          }
```

```
                        fg_puts(color,FG_MODE_XOR,~0,FG_ROT0,tx,ty,number_out
                                                        ,fg_displaybox),
                }
                text_angle=text_angle+delta_text_angle,
        }
}

void dial  panimate(){
        action--,
        if (action==0){
                action=action_default,
                if ((value'=previous_value) || (status'=previous_status)){
                //        angle proportional to 1->100% value
                        angle=(value/100)*3 1415926,
                        if ( fabs(previous_angle-angle) > ((3 1415926)/180)){
                          if (angle>3 1415926) angle=3 1415926,
                          msm_hidecursor(),
                          previous_value=value,
                          previous_status=status,
                          // locate centre
                          x1=x_center,
                          y1=y_center,

                          // set radius
                          x_radius=x_raddef+(a*mul),

                          line [FG_X1]=x1,
                          line [FG_Y1]=y1,
                          line [FG_X2]=x1+(x_radius*(cos(previous_angle))),
                          line [FG_Y2]=y1+(x_radius*(sin(previous_angle))),
                          // avoid drawing sides  of  pie  with  such  a
                          // small angle
                          // that xor of adjacent sides occurs
                          if ((x_radius*(sin(previous_angle))<1) &&
                                        (x_radius*(sin(previous_angle))>-1)){
                              if ((x_radius*(cos(previous_angle)))>0){}
                              else {fg_drawline(scolor,FG_MODE_XOR,~0
                                                ,FG_LINE_SOLID,line),}
                              }
                          else fg_drawline(scolor,FG_MODE_XOR,~0
                                                ,FG_LINE_SOLID,line),
                          tx=x1+((outer_radius+(fg_box_width(fg charbox))*2 5)
                                                *cos(previous_angle) ),
                          ty=y1+((outer_radius+(fg_box_height(fg charbox))/2)
                                                *sin(previous_angle)),
                          tx_value=tx-1 5*(fg_box_width(fg charbox)),
                          ty_value=ty-(fg_box_height(fg charbox)/2),
                          fg_puts (scolor,FG_MODE_XOR,~0,FG_ROT0
                                ,tx_value,ty_value,valuout,fg_displaybox),

                          // status colour
                          scolor=10+(status*2),

                          line [FG_X1]=x1-outer_radius,
                          line [FG_X2]=x1+outer_radius,
                          line [FG_Y2]=y1,
```

```
        fg_drawarc(color,FG_MODE_XOR,~0,x1,y1,outer_radius,ang1,ang3
                                                              ,fg_displaybox),

        line [FG_X1]=x1-outer_radius,
        line [FG_X2]=x1+outer_radius,
        line [FG_Y1]=y1,
        line [FG_Y2]=y1,
        fg_drawline(color,FG_MODE_XOR,~0,FG_LINE_SOLID,line),

        // status colour
        scolor=10+(status*2),

        line [FG_X1]=x1,
        line [FG_X2]=x1+(x_radius*(cos(previous_angle))),
        line [FG_Y2]=y1+(x_radius*(sin(previous_angle))),,

        // avoid drawing sides of pie with such a small angle
        // that xor of adjacent sides occurs

        if ((x_radius*(sin(previous_angle))<1)&&(x_radius*
                                              (sin(previous_angle))>-1)){
                if ((x_radius*(cos(previous_angle)))>0){}
                else {fg_drawline(scolor,FG_MODE_XOR,~0,FG_LINE_SOLID,line),}
                }
        else fg_drawline(scolor,FG_MODE_XOR,~0,FG_LINE_SOLID,line),

        scale(0),
        pixel_buffer=malloc (pixel_buffer_length),
        bitmap=total_name,
        fptr=fopen (bitmap,"rb"),

        if (fptr'=NULL){
                fread (pixel_buffer,bytez,pixel_buffer_length,fptr),
                fclose (fptr),
                fg_writebox (background_box,pixel_buffer),
                }
        free (pixel_buffer),

        if (fptr'=NULL){
                source[0]='d',
                source[1]='e',
                source[2]='l',
                source[3]=' ',
                for (alpha=4,*bitmap'='\0',alpha++){
                        source[alpha]=*bitmap++,}
                source[alpha]='\0',
                system (source),
                }

        msm_showcursor(),
}

// file  dial hpp flash graphics
// Vr 5 0 19/04/90 09 00 PK
// second level derived class
#ifndef DIAL_HPP
```

```
        total_name[0]='d',
        total_name[1]='1',
        q=2,
        while (sub_name[q-2]!='\0'){
                  total_name[q]=sub_name[q-2],
                  q++,
                  }
        total_name[q]=' ',
        total_name[q+1]='s',
        total_name[q+2]='a',
        total_name[q+3]='v',
        total_name[q+4]='\0',

        bitmap=total_name,
        }

    void perase(),

    ~dial() {
         perase(),
},


// file  digvalue cpp
// Vr 5 2 16 07 90 09 10 PK
#include "digvalue hpp"
#include <stdlib h>
#include <msmouse h>
#include <fg h>

void digvalue  pextent_pgen() {

  extents_sim[0]=x_center-0 5*fg_box_width(fg charbox)-offset-deadband,
  extents_sim[1]=y_center-offset-deadband,
  extents_sim[2]=x_center+6 5*(fg_box_width(fg charbox))+offset+deadband,
  extents_sim[3]=y_center+items*(fg_box_height(fg charbox))+offset+deadband,
}

void digvalue  pextent_pdraw() {
         msm_hidecursor(),

         x_min_extn=x_center-0 5*fg_box_width(fg charbox)-offset-deadband,
         y_min_extn=y_center-offset-deadband,
         x_max_extn=x_center+6 5*(fg_box_width(fg charbox))+offset+deadband,
         y_max_extn=y_center+items*(fg_box_height(fg charbox))+offset+deadband,

         extent_box[FG_X1]=x_min_extn,
         extent_box[FG_Y1]=y_min_extn,
         extent_box[FG_X2]=x_max_extn,
         extent_box[FG_Y2]=y_max_extn,

         if (extent_overlap==0) {extent_color=color,}
         else {extent_color=color-2,}
         fg_drawbox(extent_color,FG_MODE_XOR,~0,FG_LINE_MEDIUM_DASHED
                                                ,extent_box,fg_display box),
         msm_showcursor(),
```

```
y_center=y_center,

value_box[FG_X1]=x_center-0 5*fg_box_width(fg charbox),
value_box[FG_Y1]=y_center,
value_box[FG_X2]=x_center+6 5*(fg_box_width(fg charbox)),
value_box[FG_Y2]=y_center+items*(fg_box_height(fg charbox)),

frame_box[FG_X1]=value_box[FG_X1]-10,
frame_box[FG_Y1]=value_box[FG_Y1]-10,
frame_box[FG_X2]=value_box[FG_X2]+10,
frame_box[FG_Y2]=value_box[FG_Y2]+10,

frame_box_2[FG_X1]=value_box[FG_X1]-7,
frame_box_2[FG_Y1]=value_box[FG_Y1]-7,
frame_box_2[FG_X2]=value_box[FG_X2]+7,
frame_box_2[FG_Y2]=value_box[FG_Y2]+7,

ll_line[FG_X1]=frame_box[FG_X1],
ll_line[FG_Y1]=frame_box[FG_Y1],
ll_line[FG_X2]=value_box[FG_X1],
ll_line[FG_Y2]=value_box[FG_Y1],

ul_line[FG_X1]=frame_box[FG_X1],
ul_line[FG_Y1]=frame_box[FG_Y2],
ul_line[FG_X2]=value_box[FG_X1],
ul_line[FG_Y2]=value_box[FG_Y2],

lr_line[FG_X1]=value_box[FG_X2],
lr_line[FG_Y1]=value_box[FG_Y1],
lr_line[FG_X2]=frame_box[FG_X2],
lr_line[FG_Y2]=frame_box[FG_Y1],

ur_line[FG_X1]=value_box[FG_X2],
ur_line[FG_Y1]=value_box[FG_Y2],
ur_line[FG_X2]=frame_box[FG_X2],
ur_line[FG_Y2]=frame_box[FG_Y2],

ll_line_2[FG_X1]=frame_box_2[FG_X1],
ll_line_2[FG_Y1]=frame_box_2[FG_Y1],
ll_line_2[FG_X2]=frame_box[FG_X1],
ll_line_2[FG_Y2]=frame_box[FG_Y1],

ul_line_2[FG_X1]=frame_box_2[FG_X1],
ul_line_2[FG_Y1]=frame_box_2[FG_Y2],
ul_line_2[FG_X2]=frame_box[FG_X1],
ul_line_2[FG_Y2]=frame_box[FG_Y2],

lr_line_2[FG_X1]=frame_box[FG_X2],
lr_line_2[FG_Y1]=frame_box[FG_Y1],
lr_line_2[FG_X2]=frame_box_2[FG_X2],
lr_line_2[FG_Y2]=frame_box_2[FG_Y1],

ur_line_2[FG_X1]=frame_box[FG_X2],
ur_line_2[FG_Y1]=frame_box[FG_Y2],
ur_line_2[FG_X2]=frame_box_2[FG_X2],
ur_line_2[FG_Y2]=frame_box_2[FG_Y2],
```

```
        fg_drawline(FG_BLACK,FG_MODE_XOR,~0,FG_LINE_SOLID,ur_line),

        fg_drawline(FG_LIGHT_WHITE,FG_MODE_XOR,~0,FG_LINE_SOLID,ll_line_2),
        fg_drawline(FG_LIGHT_WHITE,FG_MODE_XOR,~0,FG_LINE_SOLID,ul_line_2),
        fg_drawline(FG_LIGHT_WHITE,FG_MODE_XOR,~0,FG_LINE_SOLID,lr_line_2),
        fg_drawline(FG_LIGHT_WHITE,FG_MODE_XOR,~0,FG_LINE_SOLID,ur_line_2),
}

void digvalue  panimate() {
        msm_hidecursor(),
        action--,
        if (action==0){
                action=action_default,
                y3=y_center+fg_box_height(fg charbox),

                if ((value'=previous_value)||(status'=previous_status)){
                        fg_puts (scolor,FG_MODE_XOR,~0,FG_ROT0,x_center,y3
                                                ,valuout,fg_displaybox),
                        fg_puts (scolor,FG_MODE_XOR,~0,FG_ROT0,x_center
                                        ,y_center,blank,fg_displaybox),
                        previous_status=status,
                        scolor=10+(status*2),
                        if ( status==0)
                                {
                                blank=text0,
                                fg_puts(scolor,FG_MODE_XOR,~0,FG_ROT0,x_center
                                        ,y_center,text0,fg_displaybox),
                                }
                        if (status==1)
                                {
                                blank=text1,
                                fg_puts(scolor,FG_MODE_XOR,~0,FG_ROT0,x_center
                                        ,y_center,text1,fg_displaybox),
                                }
                        previous_value=value,
                        // convert float value to char string valu[],
                        valu=ecvt (value,digits,&dec,&sign),
                        if (sign'=0)
                                valuout[0]='-',
                        else     valuout[0]='+',
                        m=1,
                        for (n=0,n<(digits+1),n++){
                                if (dec'=0)
                                        valuout[m]=valu[n],
                                else{
                                        valuout[m]=' ',
                                        m=m+1,
                                        valuout[m]=valu[n],}
                                m++,
                                dec--,}
                        // move to 2nd text position
                        fg_puts (scolor,FG_MODE_XOR,~0,FG_ROT0,x_center,y3
                                                ,valuout,fg_displaybox),
                        msm_showcursor(),
                        }
                }  }
```

```
public

   virtual void basic_pdraw(),
   virtual void pdraw(),
   virtual void panimate(),
   virtual void pextent_pgen(),
   virtual void pextent_pdraw(),
   virtual void pextent_perase(),

   digvalue(unsigned x,  unsigned  y,  int  id,  float  m,  float  raw_value,
                                  unsigned state, int extent_onoff)    () {
        x_center = x, y_center = y,
        identifier=id,                                  // cadshape id
        cad_type=13,                                     // AIPDIP type 2
        mul=m,
        max_mul=30,
        min_mul=0,
        status_extn=extent_onoff,
        status=state,
        conv_m=1,                                         // conversion multiplier
        conv_c=0,                                         //    ''  ''    constant
        value=(raw_value*conv_m)+conv_c,
        previous_status=status,                        // start-up conditions
        previous_value=value,
        fgcolor=0,
        color=15,
        digits=4,
        fillcolor=8,
        deadband=1,                                       // extents to object deadband
        items=2,
        action_default=12,
        action=action_default,
        offset=10,
        pextent_set(status_extn),

        // text0 & text1
        text0[0]='n',
        text0[1]='o',
        text0[2]='r',
        text0[3]='m',
        text0[4]='a',
        text0[5]='l',
        text0[6]='\0',

        text1[0]=' ',
        text1[1]='f',
        text1[2]='a',
        text1[3]='u',
        text1[4]='l',
        text1[5]='t',
        text1[6]='\0',

        max_extent_box[0]=x_center-(fg_box_width(fg charbox))-offset-deadband,
        max_extent_box[1]=y_center-offset-deadband,
        max_extent_box[2]=x_center+(7*(fg_box_width(fg charbox)))+offset+
                                                                    deadband,
```

```
     msm_hidecursor(),
     x_center=new_x,y_center=new_y,
     msm_setcurpos (x_center,y_center),

     reset(),
     while ((cadsh = next()) != 0){
          if (cadsh->pextent_view()) cadsh->perase(),
          }

     reset(),
     while ((cadsh = next()) != 0){
          if (!cadsh->pextent_view()) cadsh->perase(),
          }

     msm_setcurpos (x_center,y_center),
     msm_showcursor(),
}


void fgnd  move(unsigned new_x,unsigned new_y) {

     x_center=new_x,
     y_center=new_y,
     draw(x_center,y_center),
}

unsigned long fgnd  range(unsigned xr,unsigned yr) {
     // a measure of the distance between a selected point
     // and this fgnds root point
     unsigned long xx=
          xr > x_center ?
              xr-x_center    x_center-xr,
     unsigned long yy=
          yr > y_center ?
              yr-y_center    y_center-yr,
     xx *= xx,
     yy *= yy,
     return xx+yy,
     }

void fgnd  expand(float m){
     mul=m+mul,
     reset(),
     while ((cadsh = next()) !=0){
          cadsh->perase(),
          cadsh->expandp(mul),}

}

void fgnd  modulate(int  id,float  percent,unsigned  state,time_t  time_stamp,
                                                                  long run_time){
     identifier=id,
     value=percent,
     status=state,
     timestamp=time_stamp,
     runtime=run_time,
```

```
    void expand(float m),
    void modulate(int id,float percent,unsigned  state,time_t  time_stamp,long
                                                                    run_time),
    int  extent_test(float* extents_sim_ptr, cadshape* moving_shape),
    unsigned long range(unsigned x,unsigned y),

    fgnd(unsigned x, unsigned y)    () {
        x_center = x, y_center = y,

    ~fgnd() {
//        erase(),
          reset(),
          while ((cadsh=next()) != 0) {
                  remove (cadsh),
                  delete cadsh,
                  }
     }
},

// file  information hpp
// by PK for MSc project 20/07/90
#ifndef INFORMATION_HPP
#define INFORMATION_HPP

#include          <fg h>
#include          <msmouse h>
#include          "msmenu hpp"
#include          "fgnd hpp"
#include          "template hpp"

int change,                                          // objects data change flag
void info_display(template *, cadshape *, fg_coord_t, fg_coord_t),
int info_edit(template, cadshape *, fg_coord_t, fg_coord_t),
#endif INFORMATION_HPP
```

```
void msmenu  default_cursor() {
     msm_setgraphcur(-1,-1,default_cur),
}

void msmenu  menu_cursor() {
     msm_setgraphcur(-16,-8,menu_cur),
}

void msmenu  cross_cursor() {
     msm_setgraphcur(-8,-8,cross_cur),
}

int
msmenu  get_selection(unsigned x, unsigned yy) {
     unsigned int u,v, startv,
     unsigned y = yy,
     fg_box_t     read_box,
     translate_coords(&x,&y),

     // test for boundary conditions, set menu box (x,y)
     if (x<=fg_displaybox[FG_X1]+BUFFER_OUTER)
          x=fg_displaybox[FG_X1]+BUFFER_OUTER+35,
     if ((x+xsize)>=fg_displaybox[FG_X2]) x=fg_displaybox[FG_X2]-xsize,
     if ((y+LINESIZE)>=fg_displaybox[FG_Y2]-BUFFER_OUTER)
          y=fg_displaybox[FG_Y2]-LINESIZE-BUFFER_OUTER,
     if (y<=ysize-LINESIZE+fg_displaybox[FG_Y1]+BUFFER_OUTER)
          y=fg_displaybox[FG_Y1]+BUFFER_OUTER+ysize-LINESIZE,
     yy=fg_displaybox[FG_Y2]-y,   // update cursor position (MOUSE coords)

     read_box[FG_X1] = x,
     read_box[FG_Y1] = y - ysize + LINESIZE,
     read_box[FG_X2] = x + xsize,
     read_box[FG_Y2] = y + LINESIZE,

     msm_hidecursor(),
     // use special cursor
     menu_cursor(),
     // color_p = malloc(msize),
     fg_readbox(read_box, color_p),
     // blank section out
     fg_writebox(read_box, blank_p),
     fg_fillbox(FG_LIGHT_BLUE,FG_MODE_SET,~0,read_box),
     fg_drawbox(FG_WHITE,FG_MODE_SET,~0,FG_LINE_SOLID,read_box,fg displaybox) ,
     // call a private function
     drawmenu(x,y),
     // trial-and-error
     msm_setcurpos(x - 35 , yy -22),
     msm_showcursor(),
     msm_setareax(x -35, x - 35),
     msm_setareay(yy - 22,yy - 22 + ysize - LINESIZE ),
     startv = yy - 26,
     wait_left_pressed(&u,&v),
     wait_left_released(&u,&v),
```

```
    int count,   // for messages

    // private function -- can only be called
    // by member functions
    void drawmenu(int x, int y),

  public
    // constructor called for a new object

msmenu(struct menu_s * mp) {
    char buf[15],
    count = 0,
    // attach the menu pointer to this object
    m = mp,
    // Store the size of the menu
    height = 0,
    struct menu_s * mm = m,
    // look for end marker
    while ( mm->item_number != -1) {
        height++,
        mm++,
    }
    int ss, i,
    for(i = 0, width = 0, i < height, i++)
        if ((ss = strlen(mp[i] name)) > width)
            // find the largest element
            width = ss,

    // graphic width
    xsize = width * CHARWIDTH,
    // graphic height
    ysize = height * LINESIZE,

    sizing_box[FG_X1] = 0,
    sizing_box[FG_Y1] = 30,
    sizing_box[FG_X2] = xsize + LINESIZE,
    sizing_box[FG_Y2] = 30+ysize + LINESIZE,
    msize = sizing_box[FG_X2]
        - sizing_box[FG_X1] + 1,
    msize *= sizing_box[FG_Y2]
        - sizing_box[FG_Y1] + 1,
    msize *= sizeof(fg_color_t),
    unsigned long size=sizeof(fg_color_t),
    // allocate memory for graphics storage
    color_p=calloc (msize,size),
    blank_p=calloc (msize,size),

    // initialize mouse
    msm_init(),}

    // destructor called when object goes out
    // of scope or delete is used
~msmenu() {
    free (color_p),
    free (blank_p),
    // turn mouse cursor off
```

```
// file  pie cpp
// V4 8 05 04 90 17 00 PK
// Pushes background 'behind' object to dynamic buffer on draw
// Pulls background from buffer on erase
// Zortech and Flash Graphics
// Re-test of 'or' to screen 20/04/90 10 00 PK
#include "pie hpp"
#include <msmouse h>
#include <math h>

void pie  pextent_pgen() {
          // set radius
          x_radius=x_raddef+(a*mul),
          outer_radius=x_radius+3,
          extents_sim[0]=x_center-outer_radius,
          extents_sim[1]=y_center-outer_radius,
          extents_sim[2]=x_center+outer_radius,
          extents_sim[3]=y_center+outer_radius,
}     }

void pie  pextent_perase() {
          msm_hidecursor(),
          fg_drawbox(extent_color,FG_MODE_XOR,~0,FG_LINE_MEDIUM_DASHED
                                                          ,extent_box,fg_display box),
          msm_showcursor(),
}

void pie  pextent_pdraw() {
          msm_hidecursor(),

          // set radius
          x_radius=x_raddef+(a*mul),
          outer_radius=x_radius+3,

          x_min_extn=x_center-outer_radius,
          y_min_extn=y_center-outer_radius,
          x_max_extn=x_center+outer_radius,
          y_max_extn=y_center+outer_radius

          extent_box[FG_X1]=x_min_extn,
          extent_box[FG_Y1]=y_min_extn,
          extent_box[FG_X2]=x_max_extn,
          extent_box[FG_Y2]=y_max_extn,
          if (extent_overlap==0) {extent_color=color,}
          else {extent_color=color-2,}
          fg_drawbox (extent_color,FG_MODE_XOR,~0,FG_LINE_MEDIUM_DASHED
                                                  ,extent_box,fg_display box),
          msm_showcursor(),
}

void pie  pdraw() {
          msm_hidecursor(),

          // locate centre
```

```
}
void pie  panimate(){
          action--,
          if (action==0){
                    action=action_default,
                    if ((value'=previous_value) || (status'=previous_status))
                              // angle proportional to 1->100% value
                              ang2=((value/100)*3600),
                              angle=(value/100)*2*3 1415926,
                              if (fabs(previous_angle-angle) > ((2*3 1415926)/360)){
                                        {
                                        msm_hidecursor(),
                                        previous_value=value,
                                        previous_status=status,
                                        // locate centre
                                        x1=x_center,
                                        y1=y_center,

                                        // set radius
                                        x_radius=x_raddef+(a*mul),

                                        line [FG_X1]=x1+1,
                                        line [FG_X2]=x1+x_radius,
                                        line [FG_Y1]=y1,
                                        line [FG_Y2]=y1,
                                        fg_drawline(color,FG_MODE_XOR,~0,FG_LINE_SOLID
                                                                        ,line),

                                        line [FG_X1]=x1,
                                        line [FG_Y1]=y1,
                                        line [FG_X2]=x1+(x_radius
                                                  *(cos(previous_angle))),
                                        line [FG_Y2]=y1+(x_radius
                                                  *(sin(previous_angle))),
                                        // avoid drawing sides  of  pie  with  such  a
                                        // small angle
                                        // that xor of adjacent sides occurs
                                        if ((x_radius*(sin(previous_angle))<1) &&
                                              (x_radius*(sin(previous_angle))>-1)){
                                        if ((x_radius*(cos(previous_angle)))>0){}
                                        else {fg_drawline(color,FG_MODE_XOR,~0
                                                  ,FG_LINE_SOLID,line),}
                                        }
                                        else fg_drawline(color,FG_MODE_XOR,~0
                                                  ,FG_LINE_SOLID,line),
                                        fg_drawarc(scolor,FG_MODE_XOR,~0,x1,y1
                                          ,x_radius,ang1,previous_ang2,fg_displaybox),

                                        // status colour
                                        scolor=10+(status*2),

                                        line [FG_X1]=x1+1,
                                        line [FG_X2]=x1+x_radius,
                                        line [FG_Y2]=y1,
                                        fg_drawline(color,FG_MODE_XOR,~0,FG_LINE_SOLID
                                                                        ,line),
```

```
        else fg_drawline(color,FG_MODE_XOR,~0,FG_LINE_SOLID,line),

        fg_drawarc(scolor,FG_MODE_XOR,~0,x1,y1,x_radius,ang1,previous_ang2
                                                        ,fg_displaybox),
        if (fptr'=NULL){
                source[0]='d',
                source[1]='e',
                source[2]='l',
                source[3]=' ',
                for (alpha=4,*bitmap'='\0',alpha++){
                        source[alpha]=*bitmap++,}
                source[alpha]='\0',
                system (source),
                }

        msm_showcursor(),
}


// file  pie hpp flash graphics
#ifndef PIE_HPP
#define PIE_HPP
#include "cadshape hpp"

class pie    public cadshape {

protected
   char * bitmap,
   int a,
   int ang1,ang2,previous_ang2,
   double angle,previous_angle,deadband_angle,fullcir,
   fg_coord_t x1,y1,x_radius,
   fg_line_t line,
   float x_raddef,
   float outer_radius,
   int degrad,

   public

   void pdraw(),
   void panimate(),
   void pextent_pgen(),
   void pextent_pdraw(),
   void pextent_perase(),

   pie(unsigned x, unsigned y, int  id,  float  m,  float  raw_value,  unsigned
                                        state, int entent_onoff)    () {
        identifier=id,  // cadshape id
        cad_type=4,      // DIPAIP type
        color=15,         // default colour
        fgcolor=0,
        degrad=1,         // degrees - radians falg
        fullcir=3600,
        x_raddef=15,     // default pie radius
        a=2,               // multiplication factor
        ang2=0,           // default value for pie stop angle
```

```
// file  pot cpp
// V4 8 06/04/90  10 15 Zortech and Flash Graphics
#include "pot hpp"
#include <msmouse h>
#include <math h>
#include <stdlib h>

void pot  pextent_pgen() {
          // set radius
          radius2=raddef2+(a*mul),

          extents_sim[0]=x_center-radius2-2*(fg_box_width(fg charbox)),
          extents_sim[1]=y_center-radius2-3*(fg_box_height(fg charbox)),
          extents_sim[2]=x_center+radius2+2*(fg_box_width(fg charbox)),
          extents_sim[3]=y_center+radius2+(fg_box_height(fg charbox)),
}

void pot  pextent_pdraw() {
          msm_hidecursor(),

          // set radius
          radius2=raddef2+(a*mul),
          x_min_extn=x_center-radius2-2*(fg_box_width(fg charbox)),
          y_min_extn=y_center-radius2-3*(fg_box_height(fg charbox)),
          x_max_extn=x_center+radius2+2*(fg_box_width(fg charbox)),
          y_max_extn=y_center+radius2+(fg_box_height(fg charbox)),

          extent_box[FG_X1]=x_min_extn,
          extent_box[FG_Y1]=y_min_extn,
          extent_box[FG_X2]=x_max_extn,
          extent_box[FG_Y2]=y_max_extn,
          if (extent_overlap==0) {extent_color=color,}
          else {extent_color=color-2,}
          fg_drawbox(extent_color,FG_MODE_XOR,~0,FG_LINE_MEDIUM_DASHED
                                                ,extent_box,fg_display box),
          msm_showcursor(),
}
void pot  pextent_perase() {
          msm_hidecursor(),
          fg_drawbox (extent_color,FG_MODE_XOR,~0,FG_LINE_MEDIUM_DASHED
                                                ,extent_box,fg_display box),
          msm_showcursor(),
}

void pot  pdraw() {
          msm_hidecursor(),
          // locate centre
          x1=x_center,y1=y_center,

          // set radius
          radius1=raddef1+(a*mul),
          radius2=raddef2+(a*mul),

          x_min_extn=x_center-radius2-2*(fg_box_width(fg charbox)),
          y_min_extn=y_center-radius2-3*(fg_box_height(fg charbox)),
          x_max_extn=x_center+radius2+2*(fg_box_width(fg charbox)),
```

```
        value_box[FG_Y1]=ty_value,
        value_box[FG_X2]=tx_value+6*(fg_box_width(fg charbox)),
        value_box[FG_Y2]=ty_value+(fg_box_height(fg charbox)),
        fg_drawbox (FG_GRAY,FG_MODE_XOR,~0,FG_LINE_SOLID,value_box
                                                    ,fg_displaybox),
        msm_showcursor(),
}

void pot  panimate() {}

void pot  analog_pinput() {
                // test if x_input,y_input is within input range
                input_box[FG_X1]=extent_box[FG_X1]-5,
                input_box[FG_Y1]=extent_box[FG_Y1]-5,
                input_box[FG_X2]=extent_box[FG_X2]+5,
                input_box[FG_Y2]=extent_box[FG_Y2]+5,

                inside=fg_pt_inbox(input_box,x_input,y_input),
                if (inside'=0){
                        x1=x_center,y1=y_center,
                        delta_x=x_input-x1,delta_y=y_input-y1,
                        if (delta_y==0 && delta_x==0) {angle=0,}
                        else {angle=atan2 (delta_y,delta_x),}
                        if ((angle<start_angle)&&(angle>stoo_angle)){}
                        else     {
                                msm_hidecursor(),
                                fg_drawline(FG_LIGHT_BLUE,FG_MODE_XOR,~0
                                                        ,FG_LINE_SOLID,line),
                                fg_puts (FG_LIGHT_BLUE,FG_MODE_XOR,~0,FG_ROT0
                                    ,tx_value,ty_value,valuout,fg_displaybox),
                                line [FG_X1]=x1,
                                line [FG_Y1]=y1,
                                line [FG_X2]=x1+(radius1*(cos(angle))),
                                line [FG_Y2]=y1+(radius1*(sin(angle))),
                                fg_drawline(FG_LIGHT_BLUE,FG_MODE_XOR,~0
                                                    ,FG_LINE_SOLID,line),
                                if (angle>=0){value=((angle-start_angle)/
                                        (2*pi+stop_angle-start_angle))*100,}
                                if (angle<0){value=((pi-start_angle)/
                                        (2*pi+stop_angle-start_angle))*100 0,
                                value=value+((-pi-angle)/(-pi-stop_angle))*
                                                        (100 0-value),}

                                // generate output value for transducer
                                output_value=(value*conv_m)+conv_c,

                                // convert float value to char string valu[],
                                valu=ecvt (value,digits,&dec,&sign),

                                if (sign<0)
                                        valuout[0]='-',
                                else    valuout[0]='+',

                                m=1,
                                for (n=0 n<(digits+1),n++){
                                if (dec'=0)
```

```
                    source[0]='d',
                    source[1]='e',
                    source[2]='l',
                    source[3]=' ',
                    for (alpha=4,*bitmap!='\0',alpha++){
                              source[alpha]=*bitmap++,}
                    source[alpha]='\0',
                    system (source),
                    }
          msm_showcursor(),
}

// file  pot hpp
// V4 9 10/04/90 12 00 PK
#ifndef POT_HPP
#define POT_HPP
#include "cadshape npp"

class pot    public cadshape {
   char * bitmap,
   int dec,sign,n,m,digits,
   int a,angl,ang2,inside,
   double angle,start_angle,stop_angle,fullcir,text_angle,delta_text_angle,
   fg_coord_t x1,y1,tx,ty,tx_value,ty_value,radius1,radius2,
   fg_line_t line,
   float raddef1,raddef2,
   double delta_x,delta_y,
   int step,max_step,                          // pot graduations
   char *valu,                                   // for value
   char valuout[9],
   char *num,                                    // for step draw
   char number_out[3],                         // for step draw

   public

   void pdraw(),
   void panimate(),
   void analog_pinput(),
   void pextent_pgen(),
   void pextent_pdraw(),
   void pextent_perase(),

      pot(unsigned x,unsigned y,int id,float m,int extent_onoff)    () {
           identifier=id,   // cadshape id
           cad_type=3,      // AOP type
           digits=4,         // for value float -> ascii conversion
           valuout[0]='\0',// terminate string  for  pdraw()  cals  before  first
                             // pinput(),
           color=15,         // default colour
           fgcolor=0,         // default colour
           raddef1=7,       // default inner radius
           raddef2=14,       // default outer radius
           a=1,               // multiplication factor
           ang2=3600,        // default value for pie stop angle
           ang1=0,            // default value for pie start angle
           x_center = x, y_center = y,
```

```
                source[alpha]=*bitmap++,}
        source[alpha]='\0',
        system (source),
*/      }
},
#endif POT_HPP


// file  rotshaft cpp
// V4 3  Flash graphics 19 02 90 18 50
// V4 8  05 04 90 17 00 PK
// Pushes background 'behind' object to dynamic buffer on draw
// Pulls  background back to screen on erase
// V5 0  14/06/90 12 00
// Horizontal and verticle
#include "rotshaft hpp"
#include <msmouse h>
#include <bios h>
#include <fg h>

void rotshaft  pdraw() {
        msm_hidecursor(),

        flow_box [FG_X1]=x_center,
        flow_box [FG_Y1]=y_center,
        flow_box [FG_X2]=x_center+((length+(a*mul))*horiz)+(diameter*vert),
        flow_box [FG_Y2]=y_center+(diameter*horiz)+((length+(a*mul))*vert),

        x_min_extn=x_center-1,
        y_min_extn=y_center-1,
        x_max_extn=x_center+((length+1+(a*mul))*horiz)+((diameter+1)*vert),
        y_max_extn=y_center+((diameter+1)*horiz)+((length+1+(a*mul))*vert),

        background_box[FG_X1]=x_min_extn,
        background_box[FG_Y1]=y_min_extn,
        background_box[FG_X2]=x_max_extn,
        background_box[FG_Y2]=y_max_extn,

        pixel_buffer_length=(sizeof(fg_color_t)*fg_box_area(background_box)),
        pixel_buffer=malloc (pixel_buffer_length),
        fg_readbox (background_box,pixel_buffer),
        bytez=(sizeof(int))/2,
        bitmap=total_name,
        fptr=fopen (bitmap,"wb"),
        fwrite (pixel_buffer,bytez,pixel_buffer_length,fptr),
        fclose (fptr),
        free (pixel_buffer),

        fg_fillbox (FG_BLACK,FG_MODE_SET,~0,background_box),

        scolor=10+(previous_status*2),
        fg_drawbox (scolor,FG_MODE_XOR,~0,FG_LINE_SOLID,flow_box
                                                        ,fg_displaybox),
        resetvar(),
        fillpipe(),
        resetvar(),
```

```
                    xx=xx+beta,   // increment x
                }
                yy=yy+alpha,       // increment y
                xx=x_center+1,     // reset x
            }

            while ((yy>y_center)&&(horiz==1)&&(previous_value<0)){
                while (xx<x_length){
                    fg_drawdot (scolor,FG_MODE_XOR,~0,xx,yy),
                    xx=xx+beta,   // increment x
                }
                yy=yy-alpha,       // increment y
                xx=x_center+1,     // reset x
            }

            while((xx<x_diameter)&&(vert==1)&&(previous_value>=0))
                {
                while (yy<y_length){
                    fg_drawdot (scolor,FG_MODE_XOR,~0,xx,yy),
                    yy=yy+alpha,
                }
                yy=y_center+1,
                xx=xx+beta,
            }


            while ((xx>x_center)&&(vert==1)&&(previous_value<0)){
                while (yy<y_length){
                   fg_drawdot (scolor,FG_MODE_XOR,~0,xx,yy),
                   yy=yy+alpha,
                }
                yy=y_center+1,
                xx=xx-beta,
            }
}

void rotshaft  perase() {

        msm_hidecursor(),
        yy_saved=(yy*horiz),
        xx_saved=(xx*vert),

        while ((xx<x_length)&&(horiz==1)){
            while (yy<y_diameter){
                fg_drawdot (scolor,FG_MODE_XOR,~0,xx,yy),
                yy=yy+alpha,
                }
            yy=yy_saved,
            xx=xx+beta,
            }
        while ((yy<y_length)&&(vert==1)){
            while (xx<x_diameter){
                fg_drawdot (scolor FG_MODE_XOR,~0,xx,yy);
                xx=xx+alpha,
                }
            xx=xx_saved,
```

```
        cad_type=4,                // DIPAIP type
        shift=2,                    // deafult pixel shift
        alpha=6,                    // y inter dot spacing
        beta=6,           // x inter dot spacing
        length=40,                  // default box length
        diameter=8,                 // default box diameter
        status=state,
        previous_status=status, // default status

        max_extent_box[0]=x_center-1,
        max_extent_box[1]=y_center-1,
        max_extent_box[2]=x_center+((length+(a*max_mul)+1)*horiz)+
                                                        ((diameter+1)*vert),
        max_extent_box[3]=y_center+((diameter+1)*horiz)+((length+
                                                      (a*max_mul)+1)*vert),
        size=sizeof(fg_color_t),
        file_rand=rand(),
        itoa (file_rand,sub_name,10),
        total_name[0]='r',
        total_name[1]='o',
        q=2,
        while (sub_name[q-2]!='\0'){
                total_name[q]=sub_name[q-2],
                q++,
                }
        total_name[q]=' ',
        total_name[q+1]='s',
        total_name[q+2]='a',
        total_name[q+3]='v',
        total_name[q+4]='\0',

        bitmap=total_name,
    }

    void perase(),
    ~rotshaft() {
        perase(),
},
#endif ROTSHAFT_HPP


// file  shapelst cpp Zortech C++
#include "shapelst hpp"

cadshape * shapelist  next() {
    cadshape * r = current->shp,
    if ( r !=(cadshape *)0 )
        current = current->next,
    return r,
}
cadshape * shapelist  prev() {
    cadshape * r = current->shp,
    if ( r !=(cadshape *)0 )
        current = current->next,
    return r,
}
```

```
shapelist    findid (int a)
{
  int idiq,
  cadshape * cshape=0,
  cadshape * kk,
  reset(),                          // start at top of shapelst
  while ((kk = next()) != 0) {
         if ((idiq = kk->idinq ()) == a){cshape=kk,}
  }
   return cshape,
}


// file  shapelst hpp

/*
    A shapelist contains a list of shapelist elements (shapelist_els), which
    in turn hold a cadshape and a link to the next shapelist element
*/


#ifndef SHAPELST_HPP
#define SHAPELST_HPP
#include "cadshape hpp"
class cadshape,
// inform compiler that the class exists
class shapelist,

class shapelist_el {
     cadshape * shp,
     shapelist_el * next,
   public
     // allow shapelist access to the
     // private els of this class
     friend shapelist,
     shapelist_el(cadshape * s,
       shapelist_el * hd) {
          shp = s,
          next = hd,
     }
},

class shapelist  {
   protected
     shapelist_el * head, * current,

   public
//   virtual functions must have a definition in the base class
     virtual void draw() {}
     virtual void erase() {}
     virtual void move(unsigned x, unsigned y) {}
     virtual void expand(float m) {}

     shapelist() { current = head =
         new shapelist_el(
              (cadshape *)0, (shapelist_el *)0),
```

```
        x_max_extn=x_center+bar_width/2+2*(fg_box_width(fg charbox)),
        y_max_extn=y_center+slot_height/2+fg_box_height(fg charbox),

        extent_box[FG_X1]=x_min_extn,
        extent_box[FG_Y1]=y_min_extn,
        extent_box[FG_X2]=x_max_extn,
        extent_box[FG_Y2]=y_max_extn,
        if (extent_overlap==0) {extent_color=color,}
        else {extent_color=color-2,}
        fg_drawbox (extent_color,FG_MODE_XOR,~0,FG_LINE_MEDIUM_DASHED
                                              ,extent_box,fg_display box),
        msm_showcursor(),
}
void slider  pextent_perase() {
        msm_hidecursor(),
        fg_drawbox (extent_color,FG_MODE_XOR,~0,FG_LINE_MEDIUM_DASHED
                                              ,extent_box,fg_display box),
        msm_showcursor(),
}

void slider  pdraw() {
        msm_hidecursor(),
        // locate centre
        x1=x_center,y1=y_center,

        bar_width=bar_width_ref+(a*mul),
        bar_height=bar_height_ref,
        dash_width=dash_width_ref+(a*mul),
        slot_height=slot_height_ref+(a*mul),
        slot_width=(bar_width_ref/3)+(a*mul),

        x_min_extn=x_center-bar_width/2-dash_width/2-4*(fg_box_width
                                                        (fg charbox)),
        y_min_extn=y_center-slot_height/2-2*(fg_box_height(fg charbox)),
        x_max_extn=x_center+bar_width/2+2*(fg_box_width(fg charbox)),
        y_max_extn=y_center+slot_height/2+fg_box_height(fg charbox),

        extent_box[FG_X1]=x_min_extn,
        extent_box[FG_Y1]=y_min_extn,
        extent_box[FG_X2]=x_max_extn,
        extent_box[FG_Y2]=y_max_extn,

        background_box[FG_X1]=x_min_extn,
        background_box[FG_Y1]=y_min_extn,
        background_box[FG_X2]=x_max_extn,
        background_box[FG_Y2]=y_max_extn,

        pixel_buffer_length=(sizeof(fg_color_t)*fg_box_area(background_box)),
        pixel_buffer=malloc (pixel_buffer_length),
        fg_readbox (background_box,pixel_buffer),
        bytez=(sizeof(int))/2,
        bitmap=total_name,
        fptr=fopen (bitmap,"wb"),
        fwrite (pixel_buffer,bytez,pixel_buffer_length,fptr),
        fclose (fptr),
        free (pixel_buffer),
```

```
void slider  analog_pinput() {
                    // test if x_input,y_input is within input range
                    input_box[FG_X1]=extent_box[FG_X1],
                    input_box[FG_Y1]=extent_box[FG_Y1],
                    input_box[FG_X2]=extent_box[FG_X2]+5,
                    input_box[FG_Y2]=extent_box[FG_Y2]+5,

                    inside=fg_pt_inbox(input_box,x_input,y_input),
                    if (inside'=0){
                      if ((y_input<=y_center+slot_height/2-bar_height)&&(y_input>=
                                          y_center-slot_height/ 2+bar_height)){
                          msm_hidecursor(),
                          fg_drawbox(FG_LIGHT_BLUE,FG_MODE_XOR,~0,FG_LINE_SOLID
                                                    ,bar_box,fg_displaybox),
                          fg_puts (FG_LIGHT_BLUE,FG_MODE_XOR,~0,FG_ROT0,tx_value
                                                  ,ty_value,valuout,fg_displaybox),
                          bar_box[FG_Y1]=y_input-bar_height/2,
                          bar_box[FG_Y2]=y_input+bar_height/2,
                          fg_drawbox (FG_LIGHT_BLUE,FG_MODE_XOR,~0,FG_LINE_SOLID
                                                    ,bar_box,fg_displaybox),

                          value_offset=y_input-(y_center-slot_height/2+bar_height),
                          value=(value_offset/(slot_height-(2*bar_height)))*100,

                          // generate output transducer value
                          output_value=(value*conv_m)+conv_c,

                          // convert float value to char string valu[],
                          valu=ecvt (value,digits,&dec,&sign),

                          if (sign<0)
                            valuout[0]='-',
                          else     valuout[0]='+',

                          m=1,
                          for (n=0,n<(digits+1),n++){
                               if (dec'=0)
                                     valuout[m]=valu[n],
                               else{
                                     valuout[m]=' ',
                                     m=m+1,
                                     valuout[m]=valu[n],}
                               m++,
                               dec--,}
                          fg_puts (FG_LIGHT_BLUE,FG_MODE_XOR,~0,FG_ROT0,tx_value
                                                  ,ty_value,valuout,fg_displaybox),
                          msm_showcursor(),
                        }
                      msm_showcursor(),
                      }
                    else {}
          }

void slider  perase() {
          msm_hidecursor(),
```

```
// file  slider.hpp
// V4.9 10/04/90 11:30 PK
#ifndef SLIDER_HPP
#define SLIDER_HPP
#include "cadshape.hpp"

class slider   public cadshape {
  char * bitmap,
  int dec,sign,n,m,digits,                  // for float -> string conversion
  int a,inside,                               // mul constant and input box flag

  fg_coord_t x1,y1,tx,ty,tx_value,ty_value,

  float bar_width,bar_height,              // slider bar
  float dash_width,                          // dashes for value
  float slot_width,slot_height,            // slider slot
  float bar_width_ref,bar_height_ref,    // slider bar default
  float dash_width_ref,                      // dashes for value default
  float slot_width_ref,slot_height_ref, // slider slot default

  fg_box_t slot_box,                         // box for slot
  fg_box_t bar_box,                          // box for slider bar
  fg_line_t line,                            // dash to center of step

  int step,start_step,max_step,mul_step,// step counter, first, max steps  and
                                                 // mul step factor
  float delta_text_step,                   // inter step spacing
  double step_double,                        // log allocation of digit  space  for
                                                 // steps
  int digit_spaces,

  float value_offset,                        // y_input - start_step point
  char *valu,                                  // for value
  char valuout[9],
  char *num,                                   // for step draw
  char number_out[3],                        // for step draw


  public

  void pdraw(),
  void panimate(),
  void analog_pinput(),
  void pextent_pgen(),
  void pextent_pdraw(),
  void pextent_perase(),

    slider(unsigned x,unsigned y,int id,float m,int extent_onoff)    () {
        identifier=id,                                  // cadshape id
        cad_type=3,                                      // analog output
        bar_width_ref=15,                              // slider bar - default
        bar_height_ref=2,
        dash_width_ref=5,                              // dashes for value -  default
        slot_width_ref=5,                              // slider slot - default
        slot_height_ref=60,
```

```
// file  valuetile.cpp
// Vr 5.2.17 07:90 09:40 PK
#include "valuetile.hpp"
#include <stdlib.h>
#include <msmouse.h>
#include <fg.h>

void valuetile  pdraw() {
          msm_hidecursor();
          basic_pdraw();
          scolor=10+(previous_status*2);

          // convert float value to char string valu[];
          valu=ecvt (previous_value,digits,&dec,&sign);

          if (sign!=0)
                    valuout[0]='-';
          else     valuout[0]='+';
          m=1;
          for (n=0;n<(digits+1);n++){
                    if (dec!=0)
                              valuout[m]=valu[n];
                    else{
                              valuout[m]=' ';
                              m=m+1;
                              valuout[m]=valu[n];}
                    m++;
                    dec--;}
          fg_puts (scolor,FG_MODE_SET,~0,FG_ROT0,x_center,y_center,valuout
                                                              ,fg_displaybox);
          msm_showcursor();
}

void valuetile  panimate() {
          msm_hidecursor();
          action--;
          if (action==0){
                    action=action_default;

                    if ((value!=previous_value)||(status!=previous_status)){
                              fg_puts (FG_CYAN,FG_MODE_SET,~0,FG_ROT0,x_center
                                        ,y_center,valuout,fg_displaybox);
                              previous_status=status;
                              scolor=10+(status*2);
                              previous_value=value;
                              // convert float value to char string valu[];
                              valu=ecvt (value,digits,&dec,&sign);
                              if (sign!=0)
                                        valuout[0]='-';
                              else     valuout[0]='+';
                              m=1;
                              for (n=0;n<(digits+1);n++){
                                        if (dec!=0)
                                                  valuout[m]=valu[n];
                                        else{
                                                  valuout[m]=' ';
```

```
        total_name[q+2]='a',
        total_name[q+3]='v',
        total_name[q+4]='\0',
        bitmap=total_name,
    }
    ~valuetile() {
        perase(),
},
#endif VALUETILE_HPP

// file  valuetx cpp
// Vr 5 1 20 06 90 08 40 PK
#include "valuetx hpp"
#include <stdlib h>
#include <msmouse h>
#include <fg h>

void valuetx  pdraw() {
        msm_hidecursor(),
        basic_pdraw(),
        scolor=10+(previous_status*2),

        // convert float value to char string valu[],
        valu=ecvt (previous_value,digits,&dec,&sign),

        if (sign'=0)
                valuout[0]='-',
        else     valuout[0]='+',
        m=1,
        for (n=0,n<(digits+1),n++){
                if (dec'=0)
                        valuout[m]=valu[n],
                else{
                        valuout[m]=' ',
                        m=m+1,
                        valuout[m]=valu[n],}
                m++,
                dec--,}
        fg_puts (scolor,FG_MODE_XOR,~0,FG_ROT0,x_center,y_center,valuout
                                                        ,fg_displaybox),
        msm_showcursor(),
}
void valuetx  panimate() {
        msm_hidecursor(),
        action--,
        if (action==0){
                action=action_default,

                if ((value'=previous_value)||(status'=previous_status)){
                        fg_puts (scolor,FG_MODE_XOR,~0,FG_ROT0,x_center
                                ,y_center,valuout,fg_displaybox),
                        previous_status=status,
                        scolor=10+(status*2),
                        previous_value=value,
                        // convert float value to char string valu[],
                        valu=ecvt (value,digits,&dec,&sign),
```

```
          while (sub_name[q-2]!='\0'){
                    total_name[q]=sub_name[q-2],
                    q++,
                    }
          total_name[q]=' ',
          total_name[q+1]='s',
          total_name[q+2]='a',
          total_name[q+3]='v',
          total_name[q+4]='\0',
          bitmap=total_name,
     }
     ~valuetx() {
          perase(),
},
#endif VALUETX_HPP

// file  valve cpp
// Vr5 0 05/05/90 09 50 PK
// Second level inheritance
// Zortech & Flash Graphics
#include "valve hpp"
#include <msmouse h>
#include <fg h>

void valve  pextent_pgen() {

          extents_sim[0]=x_center-1 25*width,
          extents_sim[1]=y_center-1 25*height-2*fg_box_height(fg charbox),
          extents_sim[2]=x_center+1 25*width,
          extents_sim[3]=y_center+3 25*height,
}

void valve  pextent_pdraw() {
          msm_hidecursor(),

          x_min_extn=x_center-1 25*width,
          y_min_extn=y_center-1 25*height-2*fg_box_height(fg charbox),
          x_max_extn=x_center+1 25*width,
          y_max_extn=y_center+3 25*height,

          extent_box[FG_X1]=x_min_extn,
          extent_box[FG_Y1]=y_min_extn,
          extent_box[FG_X2]=x_max_extn,
          extent_box[FG_Y2]=y_max_extn,

          if (extent_overlap==0) {extent_color=color,}
          else {extent_color=color-2,}
          fg_drawbox (extent_color,FG_MODE_XOR,~0,FG_LINE_MEDIUM_DASHED
                                                  ,extent_box,fg_displaybox),
          msm_showcursor(),
          }

void valve .pdraw() {
          msm_hidecursor(),

          height=valve_height+(a*mul),
```

```
poly_valve[16]=small_box[FG_X2],
poly_valve[17]=small_box[FG_Y2]-3*height/4,
poly_valve[18]=small_box[FG_X2]+0 25*width,
poly_valve[19]=small_box[FG_Y2]-0 75*height,
poly_valve[20]=small_box[FG_X2]+0 25*width,
poly_valve[21]=small_box[FG_Y1]+0 75*height,
poly_valve[22]=small_box[FG_X2],
poly_valve[23]=small_box[FG_Y1]+0 75*height,
poly_valve[24]=large_box[FG_X2],
poly_valve[25]=large_box[FG_Y1],
poly_valve[26]=small_box[FG_X2]-width/6,
poly_valve[27]=small_box[FG_Y1],
poly_valve[28]=bot_cross_box[FG_X2]+1,
poly_valve[29]=small_box[FG_Y1],
poly_valve[30]=bot_cross_box[FG_X2]+1,
poly_valve[31]=bot_cross_box[FG_Y1]-1,
poly_valve[32]=bot_cross_box[FG_X1]-1,
poly_valve[33]=bot_cross_box[FG_Y1]-1,
poly_valve[34]=bot_cross_box[FG_X1]-1,
poly_valve[35]=small_box[FG_Y1],
poly_valve[36]=small_box[FG_X1]+width/6,
poly_valve[37]=small_box[FG_Y1],
poly_valve[38]=large_box[FG_X1],
poly_valve[39]=large_box[FG_Y1],
poly_valve[40]=small_box[FG_X1],
poly_valve[41]=small_box[FG_Y1]+0 75*height,
poly_valve[42]=small_box[FG_X1]-0 25*width,
poly_valve[43]=small_box[FG_Y1]+3*height/4,
poly_valve[44]=small_box[FG_X1]-0 25*width,
poly_valve[45]=small_box[FG_Y2]-0 75*height,
poly_valve[46]=small_box[FG_X1],
poly_valve[47]=small_box[FG_Y2]-0 75*height,
poly_valve[48]=large_box[FG_X1],
poly_valve[49]=large_box[FG_Y2],
x_min_extn=x_center-1 25*width,
y_min_extn=y_center-1 25*height-2*fg_box_height(fg charbox),
x_max_extn=x_center+1 25*width,
y_max_extn=y_center+3 25*height,


background_box[FG_X1]=x_min_extn,
background_box[FG_Y1]=y_min_extn,
background_box[FG_X2]=x_max_extn,
background_box[FG_Y2]=y_max_extn,

pixel_buffer_length=(sizeof(fg_color_t)*fg_box_area(background_box)),
pixel_buffer=malloc (pixel_buffer_length),
fg_readbox (background_box,pixel_buffer),
bytez=(sizeof(int))/2,
bitmap=total_name,
fptr=fopen (bitmap,"wb"),
fwrite (pixel_buffer,bytez,pixel_buffer_length,fptr),
fclose (fptr),
free (pixel_buffer),

fg_fillbox (fgcolor,FG_MODE_SET,~0,background_box),
```

```
void valve  panimate() {
          action--,
          if (action==0){
                    action=action_default,
                    if ((value'=previous_value)||(status'=previous_status)){
                              if (value>100) value=100,
                              if (value<0) value=0,
                              previous_status=status,
                              previous_value=value,
                              msm_hidecursor(),
                              fg_drawbox (color,FG_MODE_XOR,~0,FG_LINE_SOLID
                                                  ,top_cross_box,fg_displaybox),
                              fg_drawbox (color,FG_MODE_XOR,~0,FG_LINE_SOLID
                                                  ,bot_cross_box,fg_displaybox),
                              fg_puts (scolor,FG_MODE_XOR,~0,FG_ROT0,tx_status
                                                  ,ty_status,blank,fg_displaybox),
                              fg_fillbox (scolor,FG_MODE_XOR,~0,top_cross_box),
                              fg_fillbox (scolor,FG_MODE_XOR,~0,bot_cross_box),

                              top_cross_box [FG_Y1]=top_cross_box [FG_Y2]-0 25*
                                        height-height*(previous_value/100),
                              bot_cross_box [FG_Y2]=bot_cross_box[FG_Y1]+0 25*
                                        height+height*(previous_value/100),
                              fg_drawbox (color,FG_MODE_XOR,~0,FG_LINE_SOLID
                                                  ,top_cross_box,fg_displaybox),
                              fg_drawbox (color,FG_MODE_XOR,~0,FG_LINE_SOLID
                                                  ,bot_cross_box,fg_displaybox),

                              scolor=10+(status*2),
                              fg_fillbox (scolor,FG_MODE_XOR,~0,top_cross_box),
                              fg_fillbox (scolor,FG_MODE_XOR,~0,bot_cross_box),
                              if (status==0)              // open,shut or run
                                        {
                                        if (value<=10) blank=text0,   //open
                                        if (value>=90) {blank=text1,}        // shut
                                                  else blank=text2,                 // run
                                        }
                              else {blank=text3,}     // fail
                              fg_puts (scolor,FG_MODE_XOR,~0,FG_ROT0,tx_status
                                                  ,ty_status,blank,fg_displaybox)
                              msm_showcursor(),
                    }
          }
}

void valve  perase() {
     msm_hidecursor(),
fg_drawpolygon(color,FG_MODE_XOR,~0,FG_LINE_SOLID,24,poly_valve
                                                  ,fg displaybox) ,
     fg_drawbox  (color,FG_MODE_XOR,~0,FG_LINE_SOLID,top_cross_box
                                                  ,fg_displaybox),
     fg_drawbox  (color,FG_MODE_XOR,~0,FG_LINE_SOLID,bot_cross_box
                                                  ,fg_displaybox),
     fg_fillbox  (scolor,FG_MODE_XOR,~0,top_cross_box),
     fg_fillbox  (scolor,FG_MODE_XOR,~0,bot_cross_box),
     fg_drawbox  (color,FG_MODE_XOR,~0,FG_LINE_SOLID,actuator_box
```

```
  int ang1,ang2,                                    // ellipse start and stop angles
  fg_line_t handle_line,                           // ellipse closing line

  public
     void pdraw(),
     void panimate(),
     void pextent_pgen(),
     void pextent_pdraw(),

     valve(unsigned x,unsigned y,int id,float m,float raw_value,unsigned state,
                    int extent_onoff)    (x,y,id,m,raw_value,state,extent_onoff){
          cad_type=4,                              // DIPAIP type
          max_mul=5,                               // maximum expansion factor
          valve_height=5,valve_width=20,
          height=valve_height+a*max_mul,
          width=valve_width+a*max_mul,
          action_default=3,
          action=action_default,
          max_extent_box[0]=x_center-1 25*width,
          max_extent_box[1]=y_center-1 25*height-2*fg_box_height(fg charbox),
          max_extent_box[2]=x_center+1 25*width,
          max_extent_box[3]=y_center+3 25*height,
          size=sizeof(fg_color_t),
          file_rand=rand(),
          itoa (file_rand,sub_name,10),
          total_name[0]='v',
          total_name[1]='a',
          q=2,
          while (sub_name[q-2]!='\0'){
                    total_name[q]=sub_name[q-2],
                    q++,
                    }
          total_name[q]=' ',
          total_name[q+1]='s',
          total_name[q+2]='a',
          total_name[q+3]='v',
          total_name[q+4]='\0',
          bitmap=total_name,
     }
      void perase(),
      ~valve() {
          perase(),
},
#endif VALVE_HPP

// file  vertpipe cpp
// V4 3  Flash graphics 19 02 90 18 50
// V4 8  05 04 90 17 00 PK
// Pushes background 'behind' object to dynamic buffer on draw
// Pulls  background back to screen on erase
// Horizontal only
#include "vertpipe hpp"
#include <msmouse h>
#include <bios h>
#include <fg h>
```

```
        fclose (fptr),
        free (pixel_buffer),

        fg_fillbox (fgcolor,FG_MODE_SET,~0,background_box),
        scolor=10+(status*2),
        yy=y_center+shift,
        xx=x_center+1,
        y_length=y_center+length+(a*mul),
        x_diameter=x_center+diameter,
        while (yy<y_length){
                while (xx<x_diameter){
                        fg_drawdot (scolor,FG_MODE_XOR,~0,xx,yy),
                        xx=xx+alpha,
                        }
                xx=x_center+1,
                yy=yy+beta,
        }
        yy=y_center+shift,
        xx=x_center+1,
        y_length=y_center+length+(a*mul),
        x_diameter=x_center+diameter,

        fg_drawbox (color,FG_MODE_XOR,~0,FG_LINE_SOLID,flow_box
                                                        ,fg_displaybox),
        msm_showcursor(),
}

void vertpipe  panimate() {
        msm_hidecursor(),
        action--,
        if (action==0){
                action=action_default,
                _bios_timeofday(0,&current_time),              // mode 0 read,
                inter_sample_time=current_time-previous_time,
                if ((inter_sample_time) > (250/(value+1)))
                        {
                        yy=y_center+shift,
                        xx=x_center+1,
                        y_length=y_center+length+(a*mul),
                        x_diameter=x_center+diameter,
                        while (yy<y_length){
                                while (xx<x_diameter){
                                        fg_drawdot (scolor,FG_MODE_XOR,~0,xx,yy),
                                        xx=xx+alpha,
                                        }
                                xx=x_center+1,
                                yy=yy+beta,
                        }
                        shift++,
                        if (shift>4){shift=1,}
                        scolor=10+(status*2),
                        //prevcolor=scolor,

                        yy=y_center+shift,
                        xx=x_center+1,
                        y_length=y_center+length+(a*mul),
```

```
// file  vertpipe hpp
// Vr 4 9 12/06/90 10 45 PK
#ifndef VERTPIPE_HPP
#define VERTPIPE_HPP

#include "cadshape hpp"
#include <fg h>

class vertpipe   public cadshape {
  char * bitmap,
  int a,prevcolor,shift,alpha,beta,
  fg_box_t flow_box,
  fg_coord_t yy,xx,xx_saved,
  int length,diameter,y_length,x_diameter,,
  long current_time,
  long previous_time,
  long inter_sample_time,
  public

    void pdraw(),
    void panimate(),
    void pextent_pgen(),
    void pextent_pdraw(),
    void pextent_perase(),

    vertpipe(unsigned x,unsigned y,int  id,unsigned  xref,unsigned  yref,float
                m,float raw_value,unsigned state,int extent_onoff)    () {
        x_center = x, y_center = y,
        identifier=id,            // cadshape id
        mul=m,
        max_mul=120,
        min_mul=0,
        conv_m=1,                  // Conversion multiplier
        conv_c=0,                  // Conversion constant
        value=(raw_value*conv_m)+conv_c,
        status=state,
        status_extn=extent_onoff,
        a=1,
        fgcolor=0,
        color=15,
        shift=1,                   // deafult pixel shift
        alpha=4,                   // y inter dot spacing
        beta=4,                    // x inter dot spacing
        length=50,                 // default box length
        diameter=7,                // default box diameter
        action_default=1,
        action=action_default,
        pextent_set(status_extn),

        max_extent_box[0]=x_center-1,
        max_extent_box[1]=y_center-1,
        max_extent_box[2]=x_center+diameter+1,
        max_extent_box[3]=y_center+length+(a*max_mul)+1,

        size=sizeof(fg_color_t),
        file_rand=rand(),
```

```
        lower_left_x=fg_displaybox[FG_X1],
      if (lower_left_y<fg_displaybox[FG_Y1])
        lower_left_y=fg_displaybox[FG_Y1],
      if (upper_right_x>fg_displaybox[FG_X2])
        upper_right_x=fg_displaybox[FG_X2],
      if (upper_right_y>fg_displaybox[FG_Y2])
        upper_right_y=fg_displaybox[FG_Y2],

      // Ensure maximum area for Window < 64k points
      // by reduction of y till area < 64k
      while (((upper_right_x-lower_left_x)*(upper_right_y-lower_left_y))>=
                                                              max_window_area)
            upper_right_y--,

      window_box [FG_X1]=lower_left_x,
      window_box [FG_Y1]=lower_left_y,
      window_box [FG_X2]=upper_right_x,
      window_box [FG_Y2]=upper_right_y,

      background_box[FG_X1]=window_box [FG_X1],
      background_box[FG_Y1]=window_box [FG_Y1],
      background_box[FG_X2]=window_box [FG_X2],
      background_box[FG_Y2]=window_box [FG_Y2],

      pixel_buffer=malloc (pixel_buffer_length),
      fg_readbox (background_box,pixel_buffer),

      fg_fillbox (background_color,FG_MODE_SET,~0,background_box),
      fg_drawbox (boundary_color,FG_MODE_XOR,~0,FG_LINE_SOLID,window_box
                                                        ,fg_displaybox),
      msm_showcursor(),
      }

void window  text(char * msg, fg_color_t t_color) {
      msm_hidecursor(),

      text_box [FG_X1]=window_box[FG_X1]+fg_box_width(fg charbox),
      text_box [FG_Y1]=window_box[FG_Y1],
      text_box [FG_X2]=window_box[FG_X2]-fg_box_width(fg charbox),
      text_box [FG_Y2]=window_box[FG_Y2],

      message_string=msg,
      text_color=t_color,
      message_length=strlen(message_string)*fg_box_width(fg charbox),
      while (message_length>fg_box_width(text_box))
            message_length-=fg_box_width(fg charbox),
      text_x=text_box[FG_X1],
      text_y=text_box[FG_Y2],
      fg_puts (text_color,FG_MODE_XOR,~0,FG_ROT0,text_x,text_y
                                          ,message_string,text_box),
      //message_string=*msg+message_length
      text_y=text_box[FG_Y2]-fg_box_height(fg charbox),
      fg_puts (text_color,FG_MODE_XOR,~0,FG_ROT0,text_x,text_y
                                          ,message_string,text_box),
      msm_showcursor(),
      }
```

```
  unsigned long size,                          // element size for farcalloc
  fg_color_t * pixel_buffer,                  // background buffer pointer
  char * message_string,                       // window message
  size_t message_length,                       // window message length

  fg_color_t background_color,                // window box fill colour
  fg_color_t boundary_color,                   // window boundary colour
  fg_color_t text_color,                       // text colour

  unsigned max_window_area,                    // maximum window_box area (64k)

public

     void open (unsigned llx,unsigned lly,unsigned urx,unsigned  ury,fg_color_t
                                                    b_color,fg_color_t l_color),
     void text (char * msg,fg_color_t t_color),
     fg_coord_t text_position(),
     void erase(),
     void close(),

     window(){
          text_box[FG_X1]=0,
          message_length=0,
          max_window_area=15360,
          pixel_buffer_length=(max_window_area)*sizeof(fg_color_t),
          size=sizeof(fg_color_t),
          file_rand=rand(),
          itoa (file_rand,sub_name,10),
          total_name[0]='w',
          total_name[1]='i',
          q=2,
          while (sub_name[q-2]!='\0'){
                    total_name[q]=sub_name[q-2],
                    q++,
                    }
          total_name[q]=' ',
          total_name[q+1]='s',
          total_name[q+2]='a',
          total_name[q+3]='v',
          total_name[q+4]='\0',
          bitmap=total_name,
     }
     ~window() {
},
#endif WINDOW_HPP

// file  windowtile cpp
// V5 20 Zortech & Flash Graphics
// 17/07/90 10 20 PK

#include "windowtile hpp"
#include <msmouse h>

fg_coord_t windowtile  text_position(){
          current_text_position=text_box[FG_X1]+message_length+flange,
          return current_text_position,
```

```
        l_line_2[FG_X1]=window_box [FG_X1]-1,
        l_line_2[FG_Y1]=window_box [FG_Y1]-1,
        l_line_2[FG_X2]=window_box [FG_X1]-1,
        l_line_2[FG_Y2]=window_box [FG_Y2]+1,
        t_line_1[FG_X1]=window_box [FG_X1],
        t_line_1[FG_Y1]=window_box [FG_Y2],
        t_line_1[FG_X2]=window_box [FG_X2],
        t_line_1[FG_Y2]=window_box [FG_Y2],
        t_line_2[FG_X1]=window_box [FG_X1]-1,
        t_line_2[FG_Y1]=window_box [FG_Y2]+1,
        t_line_2[FG_X2]=window_box [FG_X2]+1,
        t_line_2[FG_Y2]=window_box [FG_Y2]+1,
        r_line_1[FG_X1]=window_box [FG_X2],
        r_line_1[FG_Y1]=window_box [FG_Y2],
        r_line_1[FG_X2]=window_box [FG_X2],
        r_line_1[FG_Y2]=window_box [FG_Y1],
        r_line_2[FG_X1]=window_box [FG_X2]+1,
        r_line_2[FG_Y1]=window_box [FG_Y2]+1,
        r_line_2[FG_X2]=window_box [FG_X2]+1,
        r_line_2[FG_Y2]=window_box [FG_Y1]-1,
        b_line_1[FG_X1]=window_box [FG_X2],
        b_line_1[FG_Y1]=window_box [FG_Y1],
        b_line_1[FG_X2]=window_box [FG_X1],
        b_line_1[FG_Y2]=window_box [FG_Y1],
        b_line_2[FG_X1]=window_box [FG_X2]+1,
        b_line_2[FG_Y1]=window_box [FG_Y1]-1,
        b_line_2[FG_X2]=window_box [FG_X1]-1,
        b_line_2[FG_Y2]=window_box [FG_Y1]-1,

        background_box[FG_X1]=outer_window_box [FG_X1],
        background_box[FG_Y1]=outer_window_box [FG_Y1],
        background_box[FG_X2]=outer_window_box [FG_X2],
        background_box[FG_Y2]=outer_window_box [FG_Y2],

        pixel_buffer=malloc(pixel_buffer_length),
        fg_readbox (background_box,pixel_buffer),

        fg_fillbox (FG_MAGENTA,FG_MODE_SET,~0,outer_window_box),
        fg_fillbox (background_color,FG_MODE_SET,~0,window_box),

        fg_drawline(FG_BLACK,FG_MODE_SET,~0,FG_LINE_SOLID,l_line_1),
        fg_drawline(FG_BLACK,FG_MODE_SET,~0,FG_LINE_SOLID,t_line_1),
        fg_drawline(FG_WHITE,FG_MODE_SET,~0,FG_LINE_SOLID,r_line_1),
        fg_drawline(FG_WHITE,FG_MODE_SET,~0,FG_LINE_SOLID,b_line_1),

        fg_drawline(FG_BLACK,FG_MODE_SET,~0,FG_LINE_SOLID,l_line_2),
        fg_drawline(FG_BLACK,FG_MODE_SET,~0,FG_LINE_SOLID,t_line_2),
        fg_drawline(FG_WHITE,FG_MODE_SET,~0,FG_LINE_SOLID,r_line_2),
        fg_drawline(FG_WHITE,FG_MODE_SET,~0,FG_LINE_SOLID,b_line_2),

        msm_showcursor(),
        }
void windowtile_close() {
    msm_hidecursor(),
    fg_fillbox (FG_BLACK,FG_MODE_SET,~0,background_box),
        fg_writebox (background_box,pixel_buffer),
```

```
// file  valuetile.cpp
// Vr 5.2.17 07/90 09:40 PK
#include "valuetile.hpp"
#include <stdlib.h>
#include <msmouse.h>
#include <fg.h>

void valuetile  pdraw() {
          msm_hidecursor(),
          basic_pdraw(),
          scolor=10+(previous_status*2),

          // convert float value to char string valu[],
          valu=ecvt (previous_value,digits,&dec,&sign),

          if (sign'=0)
                    valuout[0]='-',
          else     valuout[0]='+',
          m=1,
          for (n=0,n<(digits+1),n++){
                    if (dec'=0)
                              valuout[m]=valu[n],
                    else{
                              valuout[m]=' ',
                              m=m+1,
                              valuout[m]=valu[n],}
                    m++,
                    dec--,}
          fg_puts (scolor,FG_MODE_SET,~0,FG_ROT0,x_center,y_center,valuout
                                                              ,fg_displaybox),
          msm_showcursor(),
}

void valuetile  panimate() {
          msm_hidecursor(),
          action--,
          if (action==0){
                    action=action_default,

                    if ((value'=previous_value)||(status'=previous_status)){
                              fg_puts (FG_CYAN,FG_MODE_SET,~0,FG_ROT0,x_center
                                        ,y_center,valuout,fg_displaybox),
                              previous_status=status,
                              scolor=10+(status*2),
                              previous_value=value,
                              // convert float value to char string valu[],
                              valu=ecvt (value,digits,&dec,&sign),
                              if (sign'=0)
                                        valuout[0]='-',
                              else     valuout[0]='+',
                              m=1,
                              for (n=0,n<(digits+1),n++){
                                        if (dec'=0)
                                                  valuout[m]=valu[n],
                                        else{
                                                  valuout[m]=' ',
```

```
        total_name[q+2]='a',
        total_name[q+3]='v',
        total_name[q+4]='\0',
        bitmap=total_name,
    }
    ~valuetile() {
        perase(),
},
#endif VALUETILE_HPP

// file  valuetx cpp
// Vr 5 1 20 06 90 08 40 PK
#include "valuetx hpp"
#include <stdlib h>
#include <msmouse h>
#include <fg h>

void valuetx  pdraw() {
        msm_hidecursor(),
        basic_pdraw(),
        scolor=10+(previous_status*2),

        // convert float value to char string valu[],
        valu=ecvt (previous_value,digits,&dec,&sign),

        if (sign'=0)
                valuout[0]='-',
        else     valuout[0]='+',
        m=1,
        for (n=0,n<(digits+1),n++){
                if (dec'=0)
                        valuout[m]=valu[n],
                else{
                        valuout[m]=' ',
                        m=m+1,
                        valuout[m]=valu[n],}
                m++,
                dec--,}
        fg_puts (scolor,FG_MODE_XOR,~0,FG_ROT0,x_center,y_center,valuout
                                                        ,fg_displaybox),
        msm_showcursor(),
}
void valuetx  panimate() {
        msm_hidecursor(),
        action--,
        if (action==0){
                action=action_default,

                if ((value'=previous_value)||(status'=previous_status)){
                        fg_puts (scolor,FG_MODE_XOR,~0,FG_ROT0,x_center
                                ,y_center,valuout,fg_displaybox),
                        previous_status=status;
                        scolor=10+(status*2),
                        previous_value=value,
                        // convert float value to char string valu[],
                        valu=ecvt (value,digits,&dec,&sign),
```

```
          while (sub_name[q-2]!='\0'){
                    total_name[q]=sub_name[q-2],
                    q++,
                    }
          total_name[q]=' ',
          total_name[q+1]='s',
          total_name[q+2]='a',
          total_name[q+3]='v',
          total_name[q+4]='\0',
          bitmap=total_name,
     }
     ~valuetx() {
          perase(),
},
#endif VALUETX_HPP

// file  valve cpp
// Vr5 0 05/05/90 09 50 PK
// Second level inheritance
// Zortech & Flash Graphics
#include "valve hpp"
#include <msmouse h>
#include <fg h>

void valve  pextent_pgen() {

          extents_sim[0]=x_center-1 25*width,
          extents_sim[1]=y_center-1 25*height-2*fg_box_height(fg charbox),
          extents_sim[2]=x_center+1 25*width,
          extents_sim[3]=y_center+3 25*height,
}

void valve  pextent_pdraw() {
          msm_hidecursor(),

          x_min_extn=x_center-1 25*width,
          y_min_extn=y_center-1 25*height-2*fg_box_height(fg charbox),
          x_max_extn=x_center+1 25*width,
          y_max_extn=y_center+3 25*height,

          extent_box[FG_X1]=x_min_extn,
          extent_box[FG_Y1]=y_min_extn,
          extent_box[FG_X2]=x_max_extn,
          extent_box[FG_Y2]=y_max_extn,

          if (extent_overlap==0) {extent_color=color,}
          else {extent_color=color-2,}
          fg_drawbox (extent_color,FG_MODE_XOR,~0,FG_LINE_MEDIUM_DASHED
                                                  ,extent_box,fg_displaybox),
          msm_showcursor(),
          }

void valve  pdraw() {
          msm_hidecursor(),

          height=valve_height+(a*mul),
```

```
poly_valve[16]=small_box[FG_X2],
poly_valve[17]=small_box[FG_Y2]-3*height/4,
poly_valve[18]=small_box[FG_X2]+0 25*width,
poly_valve[19]=small_box[FG_Y2]-0 75*height,
poly_valve[20]=small_box[FG_X2]+0 25*width,
poly_valve[21]=small_box[FG_Y1]+0 75*height,
poly_valve[22]=small_box[FG_X2],
poly_valve[23]=small_box[FG_Y1]+0 75*height,
poly_valve[24]=large_box[FG_X2],
poly_valve[25]=large_box[FG_Y1],
poly_valve[26]=small_box[FG_X2]-width/6,
poly_valve[27]=small_box[FG_Y1],
poly_valve[28]=bot_cross_box[FG_X2]+1,
poly_valve[29]=small_box[FG_Y1],
poly_valve[30]=bot_cross_box[FG_X2]+1,
poly_valve[31]=bot_cross_box[FG_Y1]-1,
poly_valve[32]=bot_cross_box[FG_X1]-1,
poly_valve[33]=bot_cross_box[FG_Y1]-1,
poly_valve[34]=bot_cross_box[FG_X1]-1,
poly_valve[35]=small_box[FG_Y1],
poly_valve[36]=small_box[FG_X1]+width/6,
poly_valve[37]=small_box[FG_Y1],
poly_valve[38]=large_box[FG_X1],
poly_valve[39]=large_box[FG_Y1],
poly_valve[40]=small_box[FG_X1],
poly_valve[41]=small_box[FG_Y1]+0 75*height,
poly_valve[42]=small_box[FG_X1]-0 25*width,
poly_valve[43]=small_box[FG_Y1]+3*height/4,
poly_valve[44]=small_box[FG_X1]-0 25*width,
poly_valve[45]=small_box[FG_Y2]-0 75*height,
poly_valve[46]=small_box[FG_X1],
poly_valve[47]=small_box[FG_Y2]-0 75*height,
poly_valve[48]=large_box[FG_X1],
poly_valve[49]=large_box[FG_Y2],
x_min_extn=x_center-1 25*width,
y_min_extn=y_center-1 25*height-2*fg_box_height(fg charbox),
x_max_extn=x_center+1 25*width,
y_max_extn=y_center+3 25*height,


background_box[FG_X1]=x_min_extn,
background_box[FG_Y1]=y_min_extn,
background_box[FG_X2]=x_max_extn,
background_box[FG_Y2]=y_max_extn,

pixel_buffer_length=(sizeof(fg_color_t)*fg_box_area(background_box)),
pixel_buffer=malloc (pixel_buffer_length),
fg_readbox (background_box,pixel_buffer),
bytez=(sizeof(int))/2,
bitmap=total_name,
fptr=fopen (bitmap,"wb"),
fwrite (pixel_buffer,bytez,pixel_buffer_length,fptr),
fclose (fptr),
free (pixel_buffer),

fg_fillbox (fgcolor,FG_MODE_SET,~0,background_box),
```

```
void valve  panimate() {
          action--
          if (action==0){
                    action=action_default,
                    if ((value!=previous_value)||(status!=previous_status)){
                              if (value>100) value=100,
                              if (value<0) value=0,
                              previous_status=status,
                              previous_value=value,
                              msm_hidecursor(),
                              fg_drawbox (color,FG_MODE_XOR,~0,FG_LINE_SOLID
                                        ,top_cross_box,fg_displaybox),
                              fg_drawbox (color,FG_MODE_XOR,~0,FG_LINE_SOLID
                                        ,bot_cross_box,fg_displaybox),
                              fg_puts (scolor,FG_MODE_XOR,~0,FG_ROT0,tx_status
                                        ,ty_status,blank,fg_displaybox),
                              fg_fillbox (scolor,FG_MODE_XOR,~0,top_cross_box),
                              fg_fillbox (scolor,FG_MODE_XOR,~0,bot_cross_box),

                              top_cross_box [FG_Y1]=top_cross_box [FG_Y2]-0 25*
                                        height-height*(previous_value/100),
                              bot_cross_box [FG_Y2]=bot_cross_box[FG_Y1]+0 25*
                                        height+height*(previous_value/100),
                              fg_drawbox (color,FG_MODE_XOR,~0,FG_LINE_SOLID
                                        ,top_cross_box,fg_displaybox),
                              fg_drawbox (color,FG_MODE_XOR,~0,FG_LINE_SOLID
                                        ,bot_cross_box,fg_displaybox),

                              scolor=10+(status*2),
                              fg_fillbox (scolor,FG_MODE_XOR,~0,top_cross_box),
                              fg_fillbox (scolor,FG_MODE_XOR,~0,bot_cross_box),
                              if (status==0)             // open,shut or run
                                        {
                                        if (value<=10) blank=text0,   //open
                                        if (value>=90) {blank=text1,}        // shut
                                                  else blank=text2,                 // run
                                        }
                              else {blank=text3,}     // fail
                              fg_puts (scolor,FG_MODE_XOR,~0,FG_ROT0,tx_status
                                        ,ty_status,blank,fg_displaybox),
                              msm_showcursor(),
                    }
          }
}

void valve  perase() {
     msm_hidecursor(),
fg_drawpolygon(color,FG_MODE_XOR,~0,FG_LINE_SOLID,24,poly_valve
                                                  ,fg displaybox) ,
     fg_drawbox (color,FG_MODE_XOR,~0,FG_LINE_SOLID,top_cross_box
                                                  ,fg_displaybox),
     fg_drawbox (color,FG_MODE_XOR,~0,FG_LINE_SOLID,bot_cross_box
                                                  ,fg_displaybox),
     fg_fillbox (scolor,FG_MODE_XOR,~0,top_cross_box),
     fg_fillbox (scolor,FG_MODE_XOR,~0,bot_cross_box),
     fg_drawbox (color,FG_MODE_XOR,~0,FG_LINE_SOLID,actuator_box
```

```
  int angl,ang2,                                    // ellipse start and stop angles
  fg_line_t handle_line,                           // ellipse closing line

  public
     void pdraw(),
     void panimate(),
     void pextent_pgen(),
     void pextent_pdraw(),

     valve(unsigned x,unsigned y,int id,float m,float raw_value,unsigned state,
                    int extent_onoff)    (x,y,id,m,raw_value,state,extent_onoff){
          cad_type=4,                              // DIPAIP type
          max_mul=5,                               // maximum expansion factor
          valve_height=5,valve_width=20,
          height=valve_height+a*max_mul,
          width=valve_width+a*max_mul,
          action_default=3,
          action=action_default,
          max_extent_box[0]=x_center-1 25*width,
          max_extent_box[1]=y_center-1 25*height-2*fg_box_height(fg charbox),
          max_extent_box[2]=x_center+1 25*width,
          max_extent_box[3]=y_center+3 25*height,
          size=sizeof(fg_color_t),
          file_rand=rand(),
          itoa (file_rand,sub_name,10),
          total_name[0]='v',
          total_name[1]='a',
          q=2,
          while (sub_name[q-2]!='\0'){
                    total_name[q]=sub_name[q-2],
                    q++,
                    }
          total_name[q]=' ',
          total_name[q+1]='s',
          total_name[q+2]='a',
          total_name[q+3]='v',
          total_name[q+4]='\0',
          bitmap=total_name,
     }
      void perase(),
      ~valve() {
           perase(),
},
#endif VALVE_HPP

// file  vertpipe cpp
// V4 3  Flash graphics 19 02 90 18 50
// V4 8  05.04 90 17 00 PK
// Pushes background 'behind' object to dynamic buffer on draw
// Pulls  background back to screen on erase
// Horizontal only
#include "vertpipe hpp"
#include <msmouse h>
#include <bios h>
#include <fg h>
```

```
        fclose (fptr),
        free (pixel_buffer),

        fg_fillbox (fgcolor,FG_MODE_SET,~0,background_box),
        scolor=10+(status*2),
        yy=y_center+shift,
        xx=x_center+1,
        y_length=y_center+length+(a*mul),
        x_diameter=x_center+diameter,
        while (yy<y_length){
                while (xx<x_diameter){
                        fg_drawdot (scolor,FG_MODE_XOR,~0,xx,yy),
                        xx=xx+alpha,
                        }
                xx=x_center+1,
                yy=yy+beta,
        }
        yy=y_center+shift,
        xx=x_center+1,
        y_length=y_center+length+(a*mul),
        x_diameter=x_center+diameter,

        fg_drawbox (color,FG_MODE_XOR,~0,FG_LINE_SOLID,flow_box
                                                        ,fg_displaybox),
        msm_showcursor(),
}

void vertpipe  panimate() {
        msm_hidecursor(),
        action--,
        if (action==0){
                action=action_default,
                _bios_timeofday(0,&current_time),              // mode 0 read,
                inter_sample_time=current_time-previous_time,
                if ((inter_sample_time) > (250/(value+1)))
                        {
                        yy=y_center+shift,
                        xx=x_center+1,
                        y_length=y_center+length+(a*mul),
                        x_diameter=x_center+diameter,
                        while (yy<y_length){
                                while (xx<x_diameter){
                                     fg_drawdot (scolor,FG_MODE_XOR,~0,xx,yy),
                                     xx=xx+alpha;
                                     }
                                xx=x_center+1,
                                yy=yy+beta,
                        }
                        shift++,
                        if (shift>4){shift=1,}
                        scolor=10+(status*2),
                        //prevcolor=scolor,

                        yy=y_center+shift,
                        xx=x_center+1,
                        y_length=y_center+length+(a*mul),
```

```
// file  vertpipe hpp
// Vr 4 9 12/06/90 10.45 PK
#ifndef VERTPIPE_HPP
#define VERTPIPE_HPP

#include "cadshape hpp"
#include <fg h>

class vertpipe   public cadshape {
   char * bitmap,
   int a,prevcolor,shift,alpha,beta,
   fg_box_t flow_box,
   fg_coord_t yy,xx,xx_saved,
   int length,diameter,y_length,x_diameter,,
   long current_time,
   long previous_time,
   long inter_sample_time,
   public

      void pdraw(),
      void panimate(),
      void pextent_pgen(),
      void pextent_pdraw(),
      void pextent_perase(),

      vertpipe(unsigned x,unsigned y,int  id,unsigned  xref,unsigned  yref,float
                  m,float raw_value,unsigned state,int extent_onoff)    () {
           x_center = x, y_center = y,
           identifier=id,             // cadshape id
           mul=m,
           max_mul=120,
           min_mul=0,
           conv_m=1,                   // Conversion multiplier
           conv_c=0,                   // Conversion constant
           value=(raw_value*conv_m)+conv_c,
           status=state,
           status_extn=extent_onoff,
           a=1,
           fgcolor=0,
           color=15,
           shift=1,                    // deafult pixel shift
           alpha=4,                    // y inter dot spacing
           beta=4,                     // x inter dot spacing
           length=50,                 // default box length
           diameter=7,                // default box diameter
           action_default=1,
           action=action_default,
           pextent_set(status_extn),

           max_extent_box[0]=x_center-1,
           max_extent_box[1]=y_center-1,
           max_extent_box[2]=x_center+diameter+1,
           max_extent_box[3]=y_center+length+(a*max_mul)+1,

           size=sizeof(fg_color_t),
           file_rand=rand(),
```

```
        lower_left_x=fg_displaybox[FG_X1],
      if (lower_left_y<fg_displaybox[FG_Y1])
        lower_left_y=fg_displaybox[FG_Y1],
      if (upper_right_x>fg_displaybox[FG_X2])
        upper_right_x=fg_displaybox[FG_X2],
      if (upper_right_y>fg_displaybox[FG_Y2])
        upper_right_y=fg_displaybox[FG_Y2],

      // Ensure maximum area for Window < 64k points
      // by reduction of y till area < 64k
      while (((upper_right_x-lower_left_x)*(upper_right_y-lower_left_y))>=
                                                              max_window_area)
            upper_right_y--,

      window_box [FG_X1]=lower_left_x,
      window_box [FG_Y1]=lower_left_y,
      window_box [FG_X2]=upper_right_x,
      window_box [FG_Y2]=upper_right_y,

      background_box[FG_X1]=window_box [FG_X1],
      background_box[FG_Y1]=window_box [FG_Y1],
      background_box[FG_X2]=window_box [FG_X2],
      background_box[FG_Y2]=window_box [FG_Y2],

      pixel_buffer=malloc (pixel_buffer_length),
      fg_readbox (background_box,pixel_buffer),

      fg_fillbox (background_color,FG_MODE_SET,~0,background_box),
      fg_drawbox (boundary_color,FG_MODE_XOR,~0,FG_LINE_SOLID,window_box
                                                          ,fg_displaybox),
      msm_showcursor(),
      }

void window_text(char * msg, fg_color_t t_color) {
      msm_hidecursor(),

      text_box [FG_X1]=window_box[FG_X1]+fg_box_width(fg charbox),
      text_box [FG_Y1]=window_box[FG_Y1],
      text_box [FG_X2]=window_box[FG_X2]-fg_box_width(fg charbox),
      text_box [FG_Y2]=window_box[FG_Y2],

      message_string=msg,
      text_color=t_color,
      message_length=strlen(message_string)*fg_box_width(fg charbox),
      while (message_length>fg_box_width(text_box))
            message_length-=fg_box_width(fg charbox),
      text_x=text_box[FG_X1],
      text_y=text_box[FG_Y2],
      fg_puts (text_color,FG_MODE_XOR,~0,FG_ROT0,text_x,text_y
                                          ,message_string,text_box),
      //message_string=*msg+message_length
      text_y=text_box[FG_Y2]-fg_box_height(fg charbox)
      fg_puts (text_color,FG_MODE_XOR,~0,FG_ROT0,text_x,text_y
                                          ,message_string,text_box),
      msm_showcursor(),
      }
```

```
  unsigned long size,                          // element size for farcalloc
  fg_color_t * pixel_buffer,                  // background buffer pointer
  char * message_string,                       // window message
  size_t message_length,                       // window message length

  fg_color_t background_color,                // window box fill colour
  fg_color_t boundary_color,                  // window boundary colour
  fg_color_t text_color,                       // text colour

  unsigned max_window_area,                    // maximum window_box area (64k)

public

     void open (unsigned llx,unsigned lly,unsigned urx,unsigned  ury,fg_color_t
                                                          b_color,fg_color_t l_color),
     void text (char * msg,fg_color_t t_color),
     fg_coord_t text_position(),
     void erase()
     void close(),

     window(){
          text_box[FG_X1]=0,
          message_length=0,
          max_window_area=15360,
          pixel_buffer_length=(max_window_area)*sizeof(fg_color_t),
          size=sizeof(fg_color_t),
          file_rand=rand(),
          itoa (file_rand,sub_name,10)
          total_name[0]='w',
          total_name[1]='i',
          q=2,
          while (sub_name[q-2]!='\0'){
                    total_name[q]=sub_name[q-2],
                    q++,
                    }
          total_name[q]=' ',
          total_name[q+1]='s',
          total_name[q+2]='a',
          total_name[q+3]='v',
          total_name[q+4]='\0',
          bitmap=total_name,
     }
     ~window() {
},
#endif WINDOW_HPP

// file  windowtile cpp
// V5 20 Zortech & Flash Graphics
// 17/07/90 10 20 PK

#include "windowtile hpp"
#include <msmouse h>

fg_coord_t windowtile  text_position(){
          current_text_position=text_box[FG_X1]+message_length+flange,
          return current_text_position,
```

```
        l_line_2[FG_X1]=window_box [FG_X1]-1,
        l_line_2[FG_Y1]=window_box [FG_Y1]-1,
        l_line_2[FG_X2]=window_box [FG_X1]-1,
        l_line_2[FG_Y2]=window_box [FG_Y2]+1,
        t_line_1[FG_X1]=window_box [FG_X1],
        t_line_1[FG_Y1]=window_box [FG_Y2],
        t_line_1[FG_X2]=window_box [FG_X2],
        t_line_1[FG_Y2]=window_box [FG_Y2],
        t_line_2[FG_X1]=window_box [FG_X1]-1,
        t_line_2[FG_Y1]=window_box [FG_Y2]+1,
        t_line_2[FG_X2]=window_box [FG_X2]+1,
        t_line_2[FG_Y2]=window_box [FG_Y2]+1,
        r_line_1[FG_X1]=window_box [FG_X2],
        r_line_1[FG_Y1]=window_box [FG_Y2],
        r_line_1[FG_X2]=window_box [FG_X2],
        r_line_1[FG_Y2]=window_box [FG_Y1],
        r_line_2[FG_X1]=window_box [FG_X2]+1,
        r_line_2[FG_Y1]=window_box [FG_Y2]+1,
        r_line_2[FG_X2]=window_box [FG_X2]+1,
        r_line_2[FG_Y2]=window_box [FG_Y1]-1,
        b_line_1[FG_X1]=window_box [FG_X2],
        b_line_1[FG_Y1]=window_box [FG_Y1],
        b_line_1[FG_X2]=window_box [FG_X1],
        b_line_1[FG_Y2]=window_box [FG_Y1],
        b_line_2[FG_X1]=window_box [FG_X2]+1,
        b_line_2[FG_Y1]=window_box [FG_Y1]-1,
        b_line_2[FG_X2]=window_box [FG_X1]-1,
        b_line_2[FG_Y2]=window_box [FG_Y1]-1,

        background_box[FG_X1]=outer_window_box [FG_X1],
        background_box[FG_Y1]=outer_window_box [FG_Y1],
        background_box[FG_X2]=outer_window_box [FG_X2],
        background_box[FG_Y2]=outer_window_box [FG_Y2],

        pixel_buffer=malloc(pixel_buffer_length),
        fg_readbox (background_box,pixel_buffer),

        fg_fillbox (FG_MAGENTA,FG_MODE_SET,~0,outer_window_box),
        fg_fillbox (background_color,FG_MODE_SET,~0,window_box),

        fg_drawline(FG_BLACK,FG_MODE_SET,~0,FG_LINE_SOLID,l_line_1),
        fg_drawline(FG_BLACK,FG_MODE_SET,~0,FG_LINE_SOLID,t_line_1),
        fg_drawline(FG_WHITE,FG_MODE_SET,~0,FG_LINE_SOLID,r_line_1),
        fg_drawline(FG_WHITE,FG_MODE_SET,~0,FG_LINE_SOLID,b_line_1),

        fg_drawline(FG_BLACK,FG_MODE_SET,~0,FG_LINE_SOLID,l_line_2),
        fg_drawline(FG_BLACK,FG_MODE_SET,~0,FG_LINE_SOLID,t_line_2),
        fg_drawline(FG_WHITE,FG_MODE_SET,~0,FG_LINE_SOLID,r_line_2),
        fg_drawline(FG_WHITE,FG_MODE_SET,~0,FG_LINE_SOLID,b_line_2),

        msm_showcursor(),
        }
void windowtile_close() {
    msm_hidecursor(),
    fg_fillbox (FG_BLACK,FG_MODE_SET,~0,background_box),
        fg_writebox (background_box,pixel_buffer),
```