**CHAPTER 1**

**INTRODUCTION**

Graphics provides one of the most natural means of communicating with a computer, since our highly developed 2D and 3D pattern recognition abilities allow us to perceive and process pictorial data rapidly and efficiently. Interactive computer graphics is the most important means of producing pictures since the invention of photography and television. It has the added advantage that, with the computer, we can make pictures not only of concrete real world objects but also of abstract, synthetic objects, such as mathematical surfaces and of data that have no inherent geometry, such as survey results.

Computer graphics started with the display of data on hardcopy plotters and cathode ray tube (CRT) screens soon after the introduction of computers.

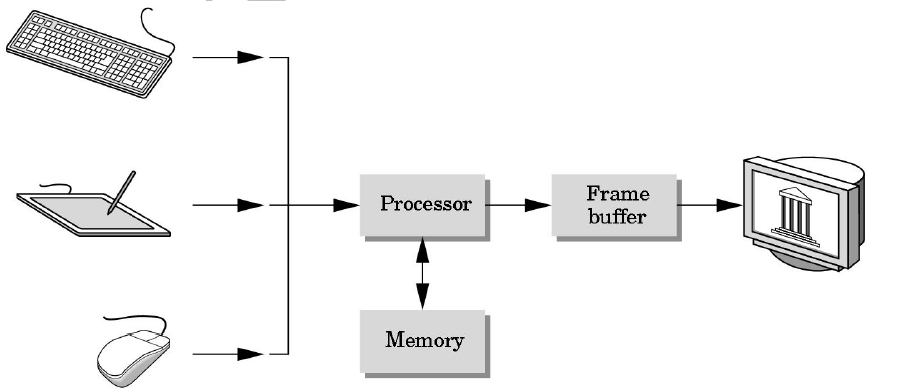
Computer graphics today largely interactive, the user controls the contents, structure, and appearance of objects and of displayed images by using input devices, such as keyboard, mouse, or touch-sensitive panel on the screen. Graphics based user interfaces allow millions of new users to control simple, low-cost application programs, such as spreadsheets, word processors, and drawing programs.

**CHAPTER 2**

**Background**

**Computer Graphics**

* Graphics provides one of the most natural means of communicating with a computer, since our highly developed 2D Or 3D pattern-recognition abilities allow us to perceive and process pictorial data rapidly.
* Computers have become a powerful medium for the rapid and economical production of pictures.
* Graphics provide a so natural means of communicating with the computer that they have become widespread.
* Interactive graphics is the most important means of producing pictures since the invention of photography and television .
* We can make pictures of not only the real world objects but also of abstract objects such as mathematical surfaces on 4D and of data that have no inherent geometry.
* A computer graphics system is a computer system with all the components of the general purpose computer system. There are five major elements in system: input devices, processor, memory, frame buffer, output devices.



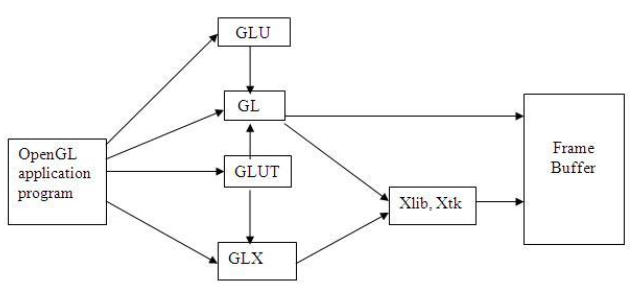
**Fig 1.1**

**OpenGL Technology**

* OpenGL is the premier environment for developing portable, interactive 2D and 3D graphics applications. Since its introduction in 1992, OpenGL has become the industry's most widely used and supported 2D and 3D graphics application programming interface (API), bringing thousands of applications to a wide variety of computer platforms.
* OpenGL fosters innovation and speeds application development by incorporating a broad set of rendering, texture mapping, special effects, and other powerful visualization functions. Developers can leverage the power of OpenGL across all popular desktop and workstation platforms, ensuring wide application deployment.
* OpenGL Available Everywhere: Supported on all UNIX® workstations, and shipped tandard with every Windows 95/98/2000/NT and MacOS PC, no other graphics API operates on a wider range of hardware platforms and software environments.
* OpenGL runs on every major operating system including Mac OS, OS/2, UNIX, Windows 95/98, Windows 2000, Windows NT, Linux, Open Step, and BeOS; it also works with every major windowing system, including Win32, MacOS, Presentation Manager, and X-Window System. OpenGL is callable from Ada, C, C++, Fortran, Python, Perl and Java and offers complete independence from network protocols and topologies.

**The OpenGL interface**

Our application will be designed to access OpenGL directly through functions in three libraries namely: gl,glu,glut.



**Fig 1.2**

OpenGL is an application program interface (API) offering various functions to implement primitives, models and images. This offers functions to create and manipulate render lighting, coloring, viewing the models. OpenGL offers different coordinate system and frames. OpenGL offers translation, rotation and scaling of objects. Most of our applications will be designed to access OpenGL directly through functions in three libraries. They are:

1. Main GL: Library has names that begin with the letter gl and are stored in a library usually referred to as GL.

2. OpenGL Utility Library (GLU): This library uses only GL functions but contains code for creating common objects and simplifying viewing.

3. OpenGL Utility Toolkit (GLUT): This provides the minimum functionality that should be accepted in any modern windowing system.

**Header Files**

Most of our applications will be designed to access OpenGL directly through functions in three libraries. Functions in the main GL library have names that begin with the letters gl and are stored in a library usually referred as GL. The second is the OpenGL Utility Library (GLU). This library uses only GL functions but contains code for creating common objects and simplifying viewing. To interface with the window system and to get input from external devices into our programs, we need at least one library. For the X window system, this library is called GLX, for windows, it is wall etc. GLUT will use GLX and X libraries. #include<GL/glut.h> OR #include<GLUT/glut.h> GLUT is the OpenGL Utility Toolkit, a window system independent toolkit for writing OpenGL programs. It implements a simple windowing application programming interface (API) for OpenGL. GLUT makes it considerably easier to learn about and explore OpenGL programming. GLUT provides a portable API so you can write a single OpenGL program that works across all PC and workstation OS platforms. GLUT is designed for constructing small to medium sized OpenGL programs. While GLUT is well-suited to learning OpenGL and developing simple OpenGL applications, GLUT is not a full-featured toolkit so large applications requiring sophisticated user interfaces are better off using native window system toolkits. GLUT is simple, easy, and small. The GLUT library has C, C++ (same as C), FORTRAN, and ADA programming bindings. The GLUT source code distribution is portable to nearly all OpenGL implementations and platforms. The current version is 3.7. Additional releases of the library are not anticipated. GLUT is not open source.

**CHAPTER 3**

**ABOUT THE PROJECT**

**Day Night Color - OpenGL Program**

We have name the program a Day Night Color because we can change the mode from Day to Night and vice versa. It is also a scenery program, where we have a house, mountain, tree, grass, sun and moon with stars.

When we execute the program, we will find simple house and tree sketched with line in black background color. There is mouse interaction which will let swap between the Day and Night view.

There are three mouse interaction added with use of menu. The three menu and their working is given below -

* quit - exit the program.
* night color change - choose the night mode.
* day color change -  select the day mode.

These menu will come when you will press the right mouse click. In day mode you will see sun and cloud, while in night mode stars and moon. House, Mountain and Tree will be same in both the mode.

**Purpose:**

The aim of this project is to develop a graphics package which supports basic operations which include building a 2D and 3d combination using Open GL. The package must also has a user-friendly interface. The objective of developing this model was to design and apply the skills we learnt in class.

**Scope:**

It provides most of the features that a graphics model should have. It is developed in C++ language. It has been implemented on windows platform. The graphics package designed here provides an interface for the users for controlling day night effect.

**CHAPTER 4**

**SYSTEM SPECIFICATION**

**HARDWARE REQUIREMENTS:**

* Dual Core Processor
* 2GB RAM
* 500GB Hard disk
* DVD drive
* Mouse and other pointing devices
* Keyboard

**SOFTWARE REQUIREMENTS:**

* Programming language – C/C++ using OpenGL
* Operating system – windows operating system
* Compiler – C Compiler
* Graphics library – GL/glut.h
* OpenGL 2.0

**FUNCTIONAL REQUIREMENTS**

**OpenGL APIs:**

If we want to have a control on the flow of program and if we want to interact with the window system then we use OpenGL API’S. Vertices are represented in the same manner internally, whether they are specified as two-dimensional or three-dimensional entities, everything that we do are here will be equally valid in three dimensions. Although OpenGL is easy to learn, compared with other APIs, it is nevertheless powerful. It supports the simple three dimensional programs and also supports the advanced rendering techniques.

**GL/glut.h:**

We use a readily available library called the OpenGL Utility Toolkit (GLUT), which provides the minimum functionality that should be expected in any modern windowing system.

The application program uses only GLUT functions and can be recompiled with the GLUT library for other window system. OpenGL makes a heavy use of macros to increase code readability and avoid the use of magic numbers. In most implementation, one of the include lines

**CHAPTER 5**

**DESIGN**

**EXISTING SYSTEM**

Existing system for a graphics is the TC++. This system will support only the 2D graphics. 2D graphics package being designed should be easy to use and understand. It should provide various options such as free hand drawing, line drawing, polygon drawing, filled polygons, flood fill, translation, rotation, scaling, clipping etc. Even though these properties were supported, it was difficult to render 2D graphics cannot be very difficult to get a 3 Dimensional object. Even the effects like lighting, shading cannot be provided.

**PROPOSED SYSTEM**

To achieve three dimensional effects, open GL software is proposed. It is software which provides a graphical interface. It is a interface between application program and graphics hardware. The advantages are:

1. Open GL is designed as a streamlined.

2. It’s a hardware independent interface i.e it can be implemented on many different hardware platforms.

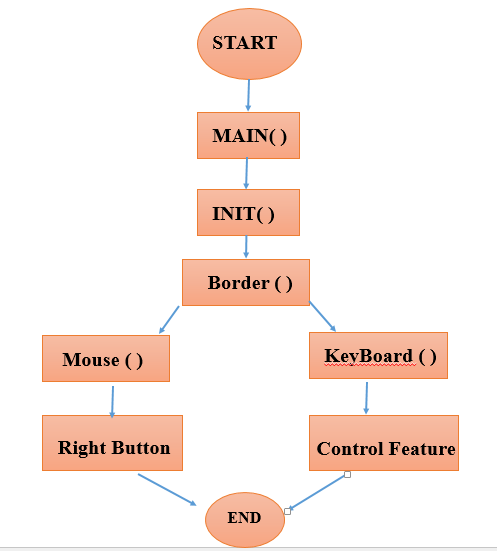
3. With openGL we can draw a small set of geometric primitives such as points, lines and polygons etc.

4. It provides double buffering which is vital in providing transformations.

5. It is event driven software.

6. It provides call back function.

**SOFTWARE DESIGN**



The controls are:-

1. Button b or B - > to initially view border .

2. Button d or D - > to view Sunlight.

3. Button n or N - > to view Moon light.

4. Button m - > to view animated moon light night.

5. Button M - > to view animated moon light night with night color changes.

6. Button s - > to view in daylight moving sun.

7. Button S - > to view in daylight moving sun with day color changes.

8. Button A - > to view move sky.

**Chapter 6**

**IMPLEMENTATION**

**FUNCTIONS GL\_LINES -** Treats each pair of vertices as an independent line segment. Vertices 2n - 1 and 2n define line n. N/2 lines are drawn.

**GL\_LINE\_LOOP -** Draws a connected group of line segments from the first vertex to the last, then back to the first. Vertices n and n + 1 define line n. The last line, however, is defined by vertices N and N lines are drawn.

Basic Functions

**glBegin, glEnd Function**

The glBegin and glEnd functions delimit the vertices of a primitive or a group of like primitives.

SYNTAX: void glBegin, glEnd(GLenum mode);

PARAMETERS: mode - The primitive or primitives that will be created from vertices presented between glBegin and the subsequent glEnd. The following are accepted symbolic constants and their meanings:

**CALL BACK FUNCTIONS**

**glutDisplayFunc Function**

glutDisplayFunc sets the display callback for the current window.

SYNTAX: void glutDisplayFunc(void (\*func)(void));

**glutReshapeFunc Function**

glutReshapeFunc sets the reshape callback for the current window.

SYNTAX: void glutReshapeFunc(void (\*func)(int width, int height));

**MAIN FUNCTION**

**glutInit Function :**glutInit is used to initialize the GLUT library.

SYNTAX:

glutInit(int \*argcp, char \*\*argv);

PARAMETERS:

A pointer to the program's unmodified argc variable from main. Upon return, the value pointed to by argcp will be updated, because glutInit extracts any command line options intended for the GLUT library. Argv - The program's unmodified argv variable from main. Like argcp, the data for argv will be updated because glutInit extracts any command line options understood by the GLUT library. glutInit(&argc,argv);

**glutInitDisplayMode Function :** glutInitDisplayMode sets the initial display mode.

SYNTAX: void glutInitDisplayMode(unsigned int mode);

PARAMETERS:

mode - Display mode, normally the bitwise OR-ing of GLUT display mode bit masks. See values below:

GLUT\_RGB: An alias for GLUT\_RGBA. GLUT\_DOUBLE:Bit mask to select a double buffered window. This overrides GLUT\_SINGLE. GLUT\_DEPTH: Bit mask to select a window with a depth buffer.

**glutMainLoop Function**

glutMainLoop enters the GLUT event processing loop.

SYNTAX:

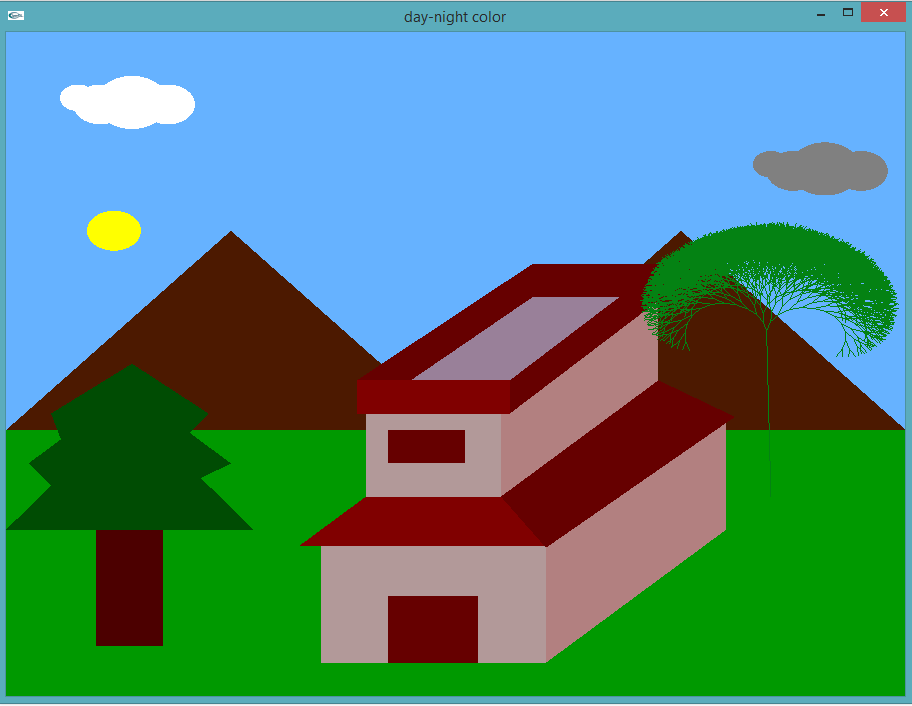
void glutMainLoop(void);

**Chapter 7**

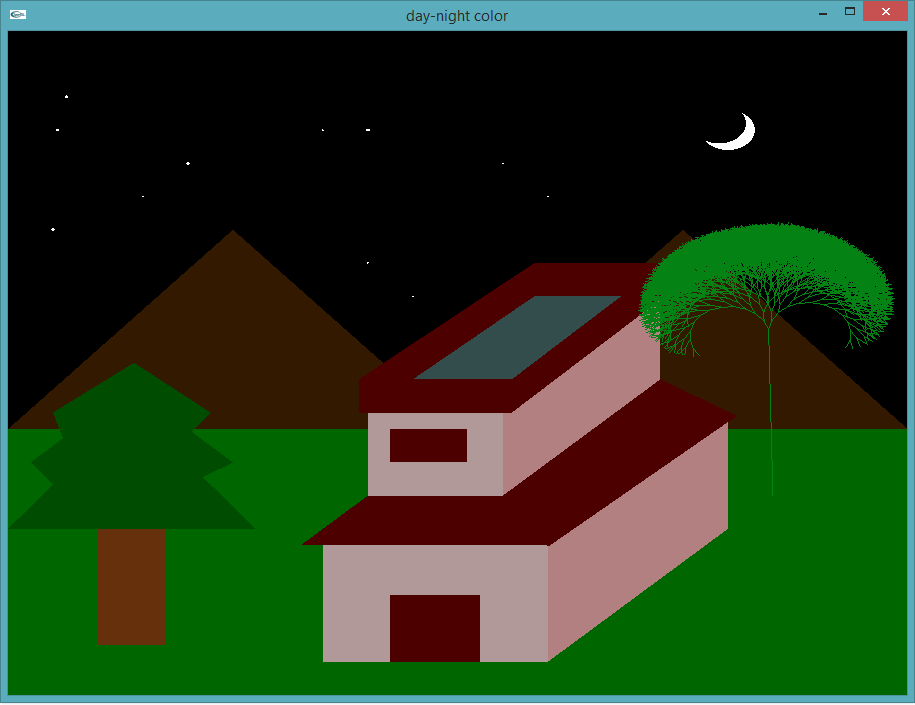
**Testing**

****

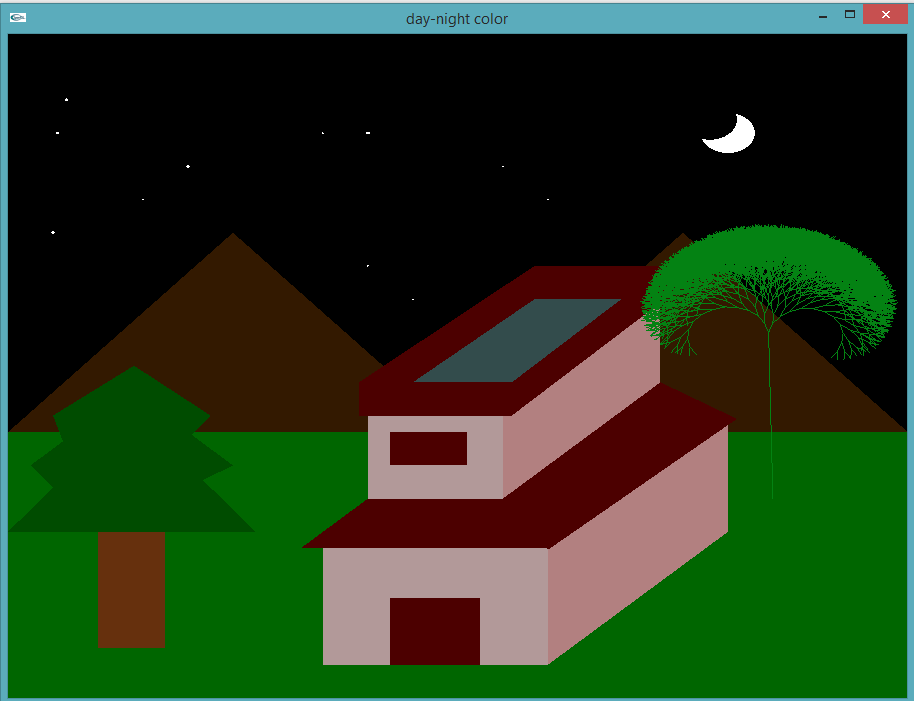
**Fig 7.1: Border**

****

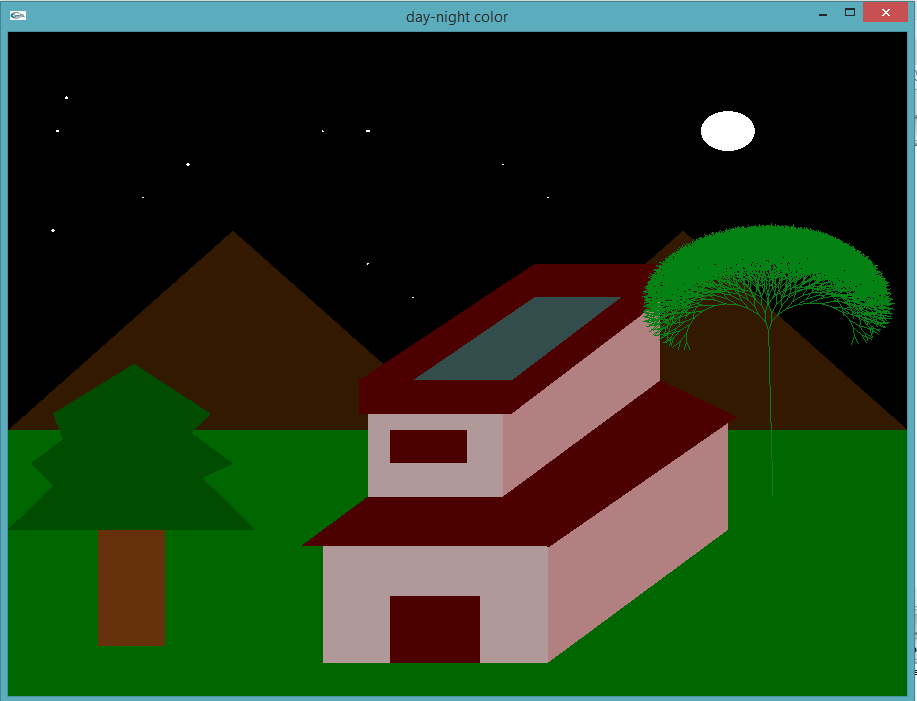
**Fig 7.2: Sunlight**

****

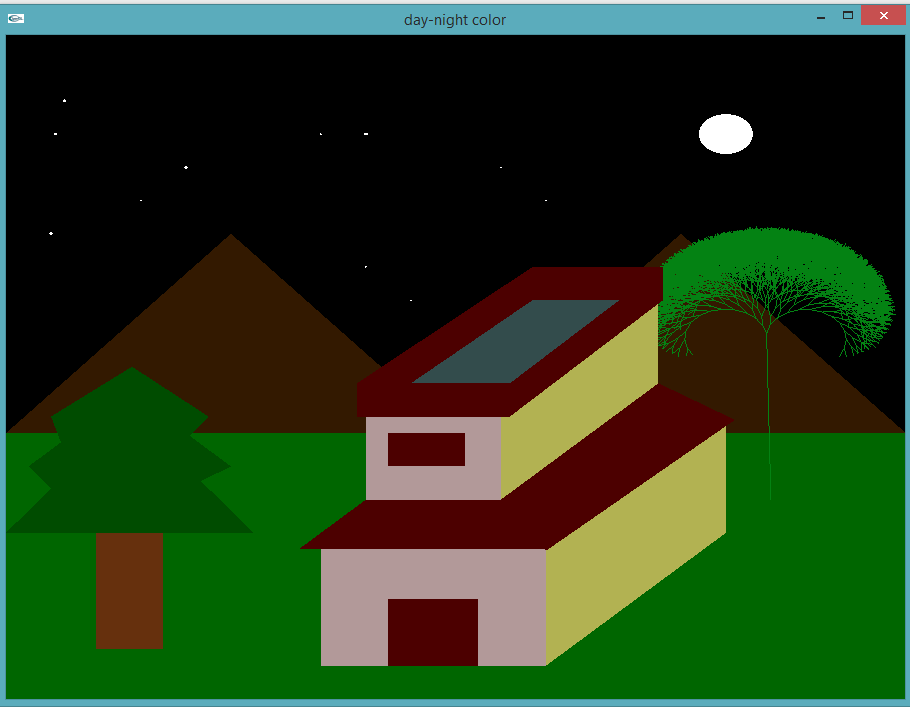
**Fig 7.3: Moon night**

****

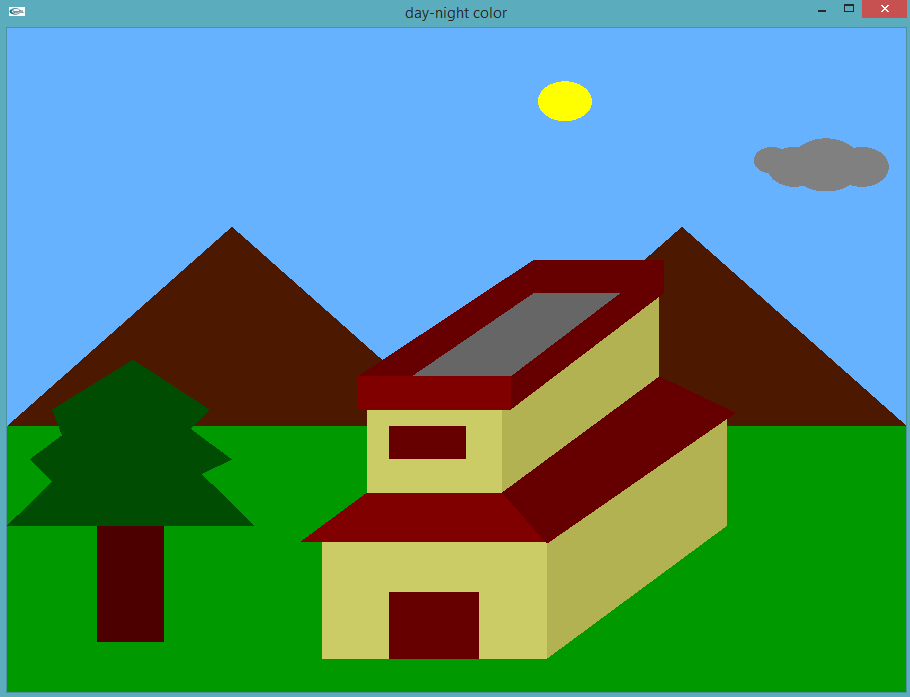
**Fig 7.4.1: Modifynight**

****

**Fig 7.4.2: Modifynight**

****

**Fig 7.5: Modifynight Color change**

****

**Fig 7.6: Sunlight Sun move**

****

**Fig 7.7: Sunlight move Sky**

**CONCLUSION AND FUTURE SCOPE**

**CONCLUSION**

The Project has been tested under Windows 10 and has been found to provide ease of use and manipulation to the user. The Project can be used to draw lines, boxes, circles, ellipses, and polygons. It has a very simple and aesthetic user interface.

We found designing and developing this project as a very interesting and learning experience. It helped us to learn about computer graphics, design of Graphical User Interfaces, interface to the user, user interaction handling and screen management. The graphics editor provides all and more than the features that have been detailed in the university syllabus.

**FUTURE ENHANCEMENTS**

These are the features that are planned to be supported in the future

\* Support for multiple canvases

\* Support for pattern filling

\* Support for 3d transformations

\* Support for transparency of layers