VISVESVARAYA TECHNOLOGICAL UNIVERSITY

“JNANA SANGAMA”, BELAGAVI-590018

A MINI PROJECT REPORT ON

A SMART CITY

Submitted in partial fulfillment of the requirement

For the award of degree of

Bachelor of Engineering

In

Computer Science and Engineering

By

YASHWANTH K

[1KS18CS121]

Under the guidance of

Dr. Dayananda R B	**Mrs. Sougandhika Narayan**
Prof, Dept. Of CSE	Asst. Prof, Dept. Of CSE

Department of Computer Science & Engineering

K.S.INSTITUTE OF TECHNOLOGY

#14, Raghuvanahalli, Kanakapura Main Road, Bengaluru-560109

2020-21

K.S.INSTITUTE OF TECHNOLOGY

#14, Raghuvanahalli, Kanakapura Main Road, Bengaluru-56010

Department of Computer Science & Engineering

CERTIFICATE

This is to certify that mini project work entitled " A SMART CITY " carried out by Mr. YASHWANTH.K bearing USN 1KS18CS121 bonafide student of **K.S. Institute of Technology** in the partial fulfilment for the award of the **Bachelor of Engineering** in **Computer Science & Engineering** of the **Visvesvaraya Technological University, Belagavi,** during the year 2021. It is certified that all corrections/suggestions indicated for Internal Assessment have been incorporated in the report deposited in the departmental library. The mini project report has been approved as it satisfies the academic requirements in respect of mini Project work prescribed for the said degree for the 6th semester

Dr . Rekha B Venkatapur
Prof & HOD, CS & E Department

Dr . Dilip Kumar K
Principal/Director, KSIT

Dr. Dayananda R B
Prof, Dept. Of CSE

Mrs. Sougandhika Narayan
Asst. Prof, Dept. Of CSE

Name of the Examiners **Signature with date**

1.

2.

ACKNOWLEDGEMENT

I take this opportunity to thank one and all involved in building this project. Firstly I would like to thank the college for providing us an opportunity to work on the project.

I would also like to thank the management of **K. S. Institute of Technology** for providing all the resources required for the project. I wish to acknowledge my sincere gratitude to our beloved Principal,
Dr. Dilip Kumar K for his encouragement and providing all facilities for the accomplishment of this project.

This project would not have been possible without the support of our beloved Prof & HOD, **Dr. Rekha.B.Venkatapur**, Dept. of CSE.

I am also highly grateful to my project guides, **Dr. Dayananda R B and Mrs.Sougandhika Narayan,** Dept. of CSE who have been very generous in assisting and supporting, to do this Project "A SMART CITY "**.**

I also would like to thank all other teaching and non-teaching staff members who have extended their support and co-operation while bringing up this project.

YASHWANTH K
(1KS18CS121)

ABSTRACT

A smart city is a project which gives good view of a city. Which contains animation of flight, Vehicles, Boat, Clouds.

It consist single view of a city . Vehicles moving one by one on road include street lights. One Flight which is moving horizontally on a screen. In the backside of the road consist River with Boats moving one by one. And animation of Clouds give beautiful look to the view .

CONTENTS

CHAPTER 1

INTRODUCTION

1.1 Introduction to Computer Graphics

Computer Graphics started with the display of data on hardcopy plotters and cathode ray tube screens soon after the introduction of computer themselves. Now it has grown to that extent that it has included the creation, storage and manipulation of models and images of objects. Computer Graphics today is largely interactive; it is the user who controls the contents, structure and appearance of objects and of their displayed images by using input devices, such as keyboard, mouse or touch-sensitive panel on the screen. Although early applications in engineering and science had to rely on expensive and cumbersome equipment, advance in computer technology have made interactive graphics as a practical tool.

Computer Graphics is an integral part of all computer user interfaces, and is indispensable for visualizing two dimensional, three-dimensional, and higher dimensional objects: areas as diverse such as education, science, Computer Graphics started with the display of data on hardcopy plotters and cathode ray tube screens soon after the introduction of computer themselves. Now it has grown to that extent that it has included the creation, storage and manipulation of models and images of objects.

Computer Graphics today is largely interactive; it is the user who controls the contents, structure and appearance of objects and of their displayed images by using input devices, such as keyboard, mouse or touch-sensitive panel on the screen. Although early applications in engineering and science had to rely on expensive and cumbersome equipment, advance in computer technology have made interactive graphics as a practical tool. Computer Graphics is an integral part of engineering, medicine, commerce, the military; advertising and entertainment all rely on computer graphics.

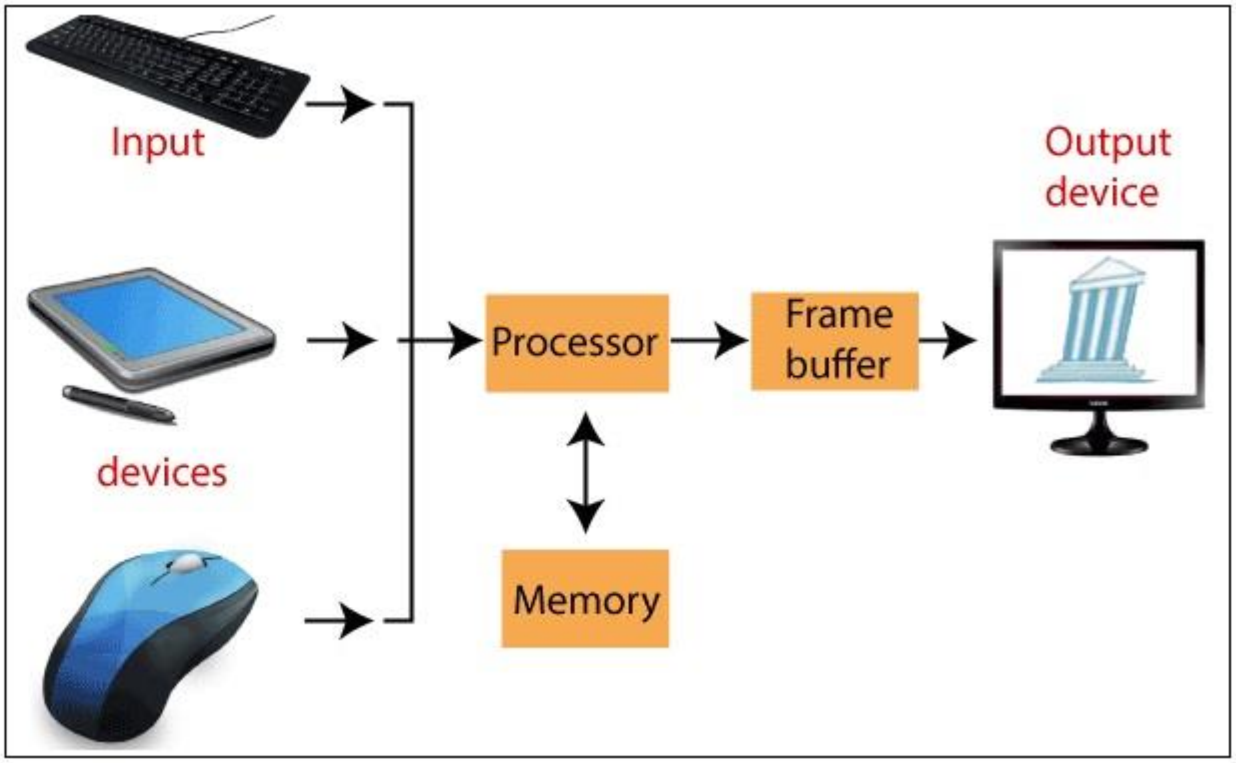


Fig. 1.1: Graphics System

Until the early 1980's, Computer Graphics was a small, specialized field, largely because the hardware was expensive and graphics-based application programs that were easy to use and cost effective were few. Then, personal computers with built in raster graphics displays popularized the use of bitmap graphics for array of points or also called as pixels on the screen. Once bitmap graphics became affordable, an explosion of easyto-use and inexpensive graphics-based applications soon followed. Graphics-based interfaces allowed millions of news users to control simple, low cost application programs, such as spreadsheets, word processors, and drawing programs. Graphics has its own hardcopy technologies, input technologies and also display technologies. Some of the hardcopy technologies are printers, pen plotters etc. Some of the display technologies such direct view storage tube, liquid crystal displays, plasma panels etc. Input technologies like keyboard, mouse, touch panel, tablets etc.

1.2 Applications of Computer Graphics

Computer graphics is used today in mainly different areas of industry, business, government, education, entertainment, and most recently, the home. The list of applications is enormous and is growing rapidly as computers with graphics capabilities become commodity products. Now let us see some of the applications.

- **Computer Aided Design**
  Computer graphics is used in design process, particularly for engineering and architectural systems, but almost all products are now computer designed. Generally referred to as CAD, computer aided design methods are now routinely used in the design of buildings, automobiles, aircraft, watercraft, spacecraft, computers, textiles, and many other products.

- **Image Processing**
  In computer graphics, a computer is used to create a picture. Image processing on the other hand, applies techniques to modify or interpret existing pictures, such as photographs and TV scans. Two principal applications of image processing are improving picture quality and machine perception of visual information, as used in robotics.

- **Education and Training**
  Computer generated models of physical, financial and economic systems are often used as educational aids. Models of physical systems, physiological systems, population trends, or equipment, can help trainees to understand the operation of the system.

- **Entertainment**
  Computer graphics methods are now commonly used in making motion pictures, music videos, and television shows.

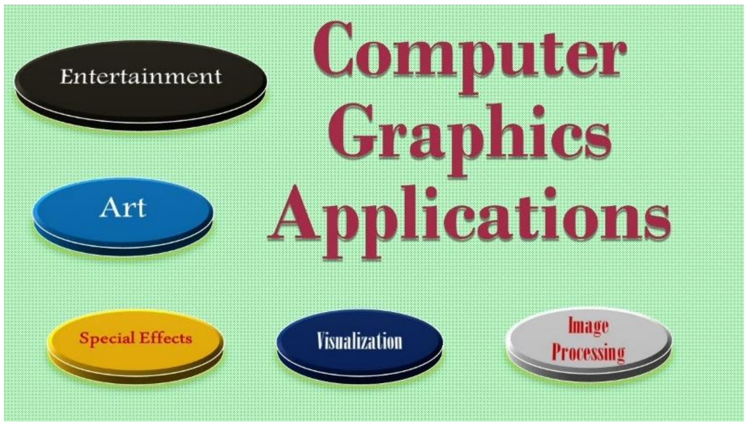


Fig. 1.2: Application of Computer Graphics

1.3 History of OpenGL

In the 1980s, developing software that could function with a wide range of graphics hardware was a real challenge. Software developers wrote custom interfaces and drivers for each piece of hardware. This was expensive and resulted in much duplication of effort.

By the early 1990s, Silicon Graphics (SGI) was a leader in 3D graphics for workstations. Their IRIS GL API was considered the state of the art and became the de facto industry standard, overshadowing the open standards-based PHIGS. This was because IRIS GL was considered easier to use, and because it supported immediate mode rendering. By contrast, PHIGS was considered difficult to use and outdated in terms of functionality.

SGI's competitors (including Sun Microsystems, Hewlett-Packard and IBM) were also able to bring to market 3D hardware, supported by extensions made to the PHIGS standard. This in turn caused SGI market share to weaken as more 3D graphics hardware suppliers entered the market. In an effort to influence the market, SGI decided to turn the IrisGL API into an open standard.

SGI considered that the IrisGL API itself wasn't suitable for opening due to licensing and patent issues. Also, the IrisGL had API functions that were not relevant to 3D graphics. For example, it included a windowing, keyboard and mouse API, in part because it was developed before the X Window System and Sun's News systems were developed.

In addition, SGI had a large number of software customers; by changing to the OpenGL API they planned to keep their customers locked onto SGI (and IBM) hardware for a few years while market support for OpenGL matured. Meanwhile, SGI would continue to try to maintain their customers tied to SGI hardware by developing the advanced and proprietary Iris Inventor and Iris Performer programming APIs.

As a result, SGI released the OpenGL standard. The OpenGL standardized access to hardware, and pushed the development responsibility of hardware interface programs, sometimes called device drivers, to hardware manufacturers and delegated windowing functions to the underlying operating system. With so many different kinds of graphic hardware, getting them all speak the same language in this way had a remarkable impact by giving software developers a higher level platform for 3D-software development.

In 1992 SGI led the creation of the OpenGL architectural review board (OpenGL ARB), the group of companies that would maintain and expand the OpenGL specification for years to come. OpenGL evolved from (and is very similar in style to) SGI's earlier 3D interface, IrisGL. One of the restrictions of IrisGL was that it only provided access to features supported by the underlying hardware. If the graphics hardware did not support a feature, then the application could not use it. OpenGL overcame this problem by providing support in software for features unsupported by hardware, allowing applications to use advanced graphics on relatively low powered systems.

1.3.1 Important Functions of OpenGL

The OpenGL Utility Toolkit is a library of utilities for OpenGL programs, which primarily perform system-level I/O with the host operating system. The glut library is organized as shown

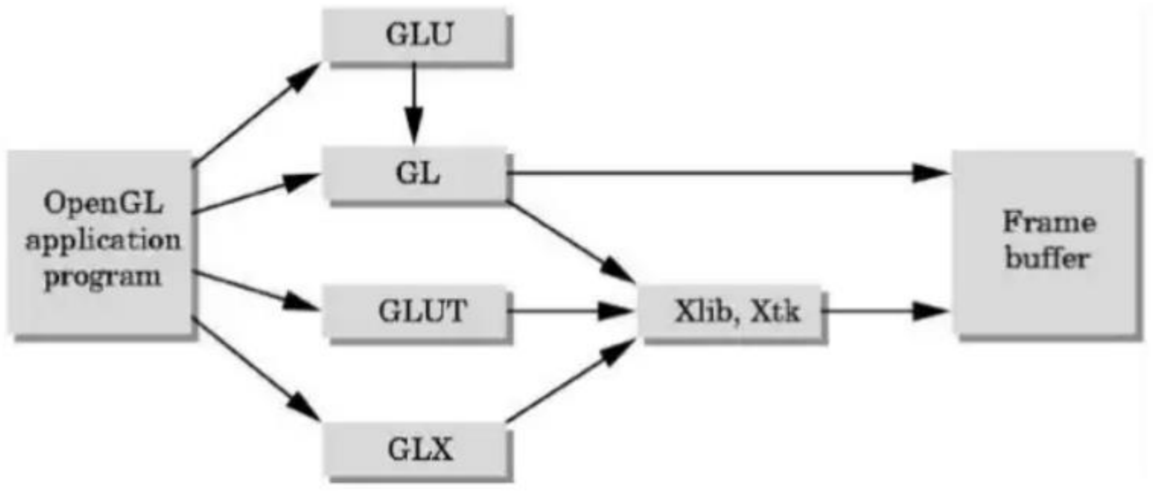


Fig. 1.3: OpenGL libraries

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 both C, C++ (same as C), FORTRAN, and Ada programming bindings. The GLUT source code distribution is portable to nearly all OpenGL implementations and platforms.

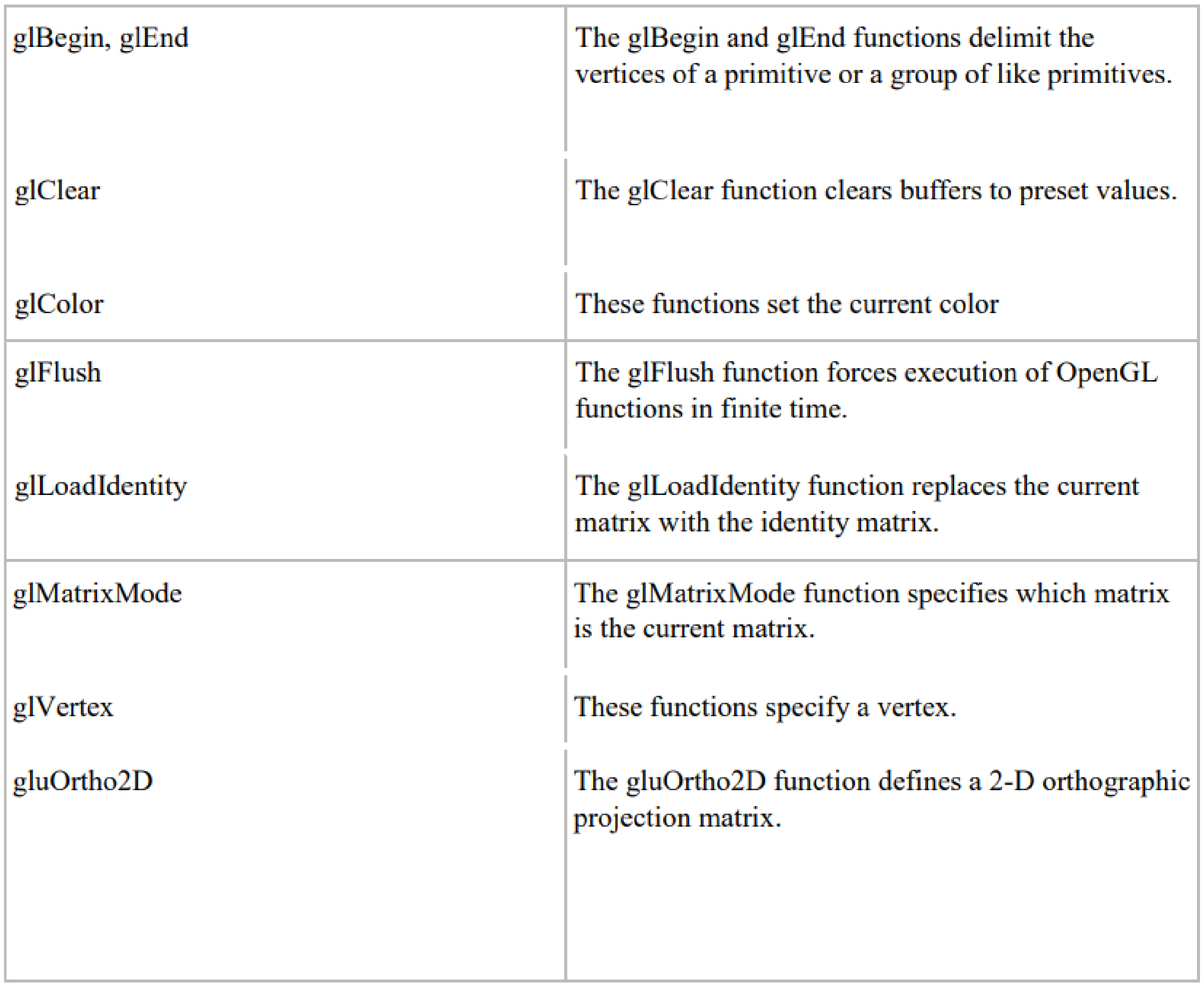
Examples:

gl functions: glBegin(), glEnd(), glBitmap(), glClear(), glFlush(), glOrtho(), glPointSize()

glu functions: gluLookAt(), gluOrtho2D(), gluSphere(), gluPerspective()

glut functions: glutInit(), glutFullScreen(), glutMainLoop(), glutCreateMenu()

Some of the important openGL built in functions with their functionalities are listed in following table:



glBegin, glEnd	The glBegin and glEnd functions delimit the vertices of a primitive or a group of like primitives.
glClear	The glClear function clears buffers to preset values.
glColor	These functions set the current color
glFlush	The glFlush function forces execution of OpenGL functions in finite time.
glLoadIdentity	The glLoadIdentity function replaces the current matrix with the identity matrix.
glMatrixMode	The glMatrixMode function specifies which matrix is the current matrix.
glVertex	These functions specify a vertex.
gluOrtho2D	The gluOrtho2D function defines a 2-D orthographic projection matrix.

Fig 1.4: Important openGL built in functions with their functionalities

1.3.2 Important Features of OpenGL

OpenGL provides a set of commands to render a three-dimensional scene. That means you provide the data in an OpenGL-useable form and OpenGL will show this data on the screen (render it). It is developed by many companies and it is free to use.

OpenGL is an API and system-independent interface. An OpenGL-application will work on every platform, as long as there is an installed implementation.

OpenGL is a collection of several hundred functions that provide access to all of the features that your graphics hardware has to offer. Internally it acts like a state machine- a collection of states that tell OpenGL what to do. Using the API you can set various aspects of this state machine, including current color, blending, lighting effect, etc.

Because it is system independent, there are no functions to create windows etc., but there are helper functions for each platform. A very useful thing is GLUT.

1.4 Advantages of OpenGL

- It is user-friendly and speed up the user's work.
- It is more attractive for non-technical people.
- In general, it looks more professional (but this does not mean it is always the best solution).
- OpenGL is a cross-platform graphics API, which means that the same code can be used on multiple operating system types with minimal changes.
- OpenGL runs on every computer with graphics output capability and requires no extra downloads.

1.5 Drawbacks of OpenGL

- When it is not properly built, it can be very difficult to work with.
- It generally requires more resources than a non-graphical one.
- It might require the installation of additional software e.g., the “runtime environment” in the case of java.

CHAPTER 2

LITERATURE SURVEY

In late 1960's the development of Free-form curves and surfaces for computer graphics begins. Free-form curves and surfaces where developed to describe curved 3-D objects without using polyhedral representations which are bulky and intractable. To get a precise curve with polygons might require thousands of faces, whereas curved surfaces requires much less calculations. The UNISURF CAD system was created for designing cars which utilized the curve theories.

A research was made but was never published so designers get most of the credit. The men were pioneers in Computer Aided Geometric Design(CAGD) for the auto industry, which replaced the use of hand drawn French-curve templates in design of auto bodies. The curves were based on Bernstein Polynomials which had been developed by the mathematician Bernstein much earlier. Another kind of basic curve predated that was the Hermite Curve developed by the mathematician Hermite. Also, in the same era as, Schoenberg, a mathematician at the university of Wisconsin was working on Mathematical Splines, which would influence the work of S.Coons at MIT in Splines, Bicubic Surface Patches, Rational Polynomials around 1968. A surface patch is freeform curved surface defined by two or more curves.

In 1973, designers based their research into parametric B-Splines on Coon's work. The main difference between B-Splines and Bezier curves is the former allows for local control system. B-Splines and Bezier curves is the former allows for local control of key control points and the later has more of a global control system. B-Splines are also faster to calculate for a computer than cubic polynomial based curves like the Hermite and Bezier.Rosenfield's pioneering development of B-Splines later influenced the E.catmull's research at Utah.

Chapter 3

REQUIREMENT SPECIFICATION

3.1 Domain Understanding

The main objective is to develop a suitable OpenGL graphics package to implement basic computer graphics skills. The aim of the project is the animation of “2D Windmill”.

Program to demonstrate the Rendering of a Beautiful 2D Windmill Animation, in which the user can interact with it.

3.2 System Requirements

Here we are using header files namely <GL/glut.h> , functions in the OpenGL in windows are stored in a library usually referred as GL. GLUT uses only GL functions and also contains code for creating objects and simplifying, viewing, rather than using different library for each system here we can use readily available library called the OPENGL UTILITY TOOL KIT(GLUT)

3.2.1 SOFTWARE REQUIREMENTS

Number	Description	Type
1	Operating System	Windows / Ubuntu
2	Libraries	OpenGL
3	Compiler	GNU GCC Compiler / G++ Compiler
4	IDE	Code Blocks, geditor
5	Language	C++

3.2.2 HARDWARE REQUIREMENTS

Number	Description
1	PC with 5 GB or more Hard disk.
2	PC with 1 GB RAM.
3	PC with Pentium 1 and Above.
4	Interface Device: Mouse, Keyboard, Monitor
5	Monitor Screen Resolution 1200x700

CHAPTER 4

SYSTEM ANALYSIS AND DESIGN

4.1 System Analysis

- In this project we are concentrating on using keyboard and mouse.
- Execute the program using openGL and just press esc button to exit

4.2 System Design

The Main function calls are :

```
cout<< " YASHWANTH K(1KS18CS121) , KSIT"<<endl<<endl<<endl;

cout<< " INSTRUCTIONS :  "<<endl;

cout<< " Press  'ESC' to STOP The project "<<endl;

     glutInit(&argc, argv);

     glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB);

     glutInitWindowSize(1050,200);

     glutCreateWindow("City");

     gluOrtho2D(0.0, 2000.0, 0.0, 1500.0);

     glutKeyboardFunc(keyboard);

     glutDisplayFunc(myDisplay);

     glutTimerFunc(20, update, 0);

     glutFullScreen();

     glutMainLoop();

     return 0;
```

CHAPTER 5

IMPLEMENTATION

5.1 Description of Implementation Modules

In this project I have created a simple 2D Windmill Animation using "OpenGL" functional API by the help of built in the functions present in the header file. To provide functionality to the project I have written user defined functions. These functions provide us the efficient way to design the project. In this chapter we are describing the functionality of our project using these functions.

5.2 Headers Files

#include <GL/glut.h> : The glut.h header file contains definition of openGL functions

#include <math.h> : The math.h header file defines various mathematical functions.

#include<stdio.h> : The stdio.h header file for standard input and output.

#include<string.h> : The string.h header file defines various functions for manipulating arrays of characters.

5.3 List of Implementation Functions

User Defined Functions:

void update(int value):

This function is used to update the movement of the objects.

void middleCar():

This function is used add the car in the middle road

void leftCar():

This function is used add the car in the left side road

void rightCar():

This function is used add the car in the right side road

void road ():

This function is used add the road

void rightBoat():

This function is used add the boat in the right side of a river

void leftBoat():

This function is used add the boat in the left side of a river

void river():

This function is used add the river

void Plane():

This function is used to add plane

void building():

This function is used add the Buildings

void normalSky():

This function is used add the only sky

void rightCloud():

This function is used add the cloud which is right side of sky

void leftCloud():

This function is used add the cloud which is left side of sky

void sunMaking():

This function is used add the sun in between clouds.

5.4 Description of inbuilt functions

Window Manager Setup Using glut

glutInit(&argc, argv) - initializes GLUT, processes any command-line arguments for X functions. Call this before anything else.

glutInitDisplayMode(unsigned int mode) - sets the basic display modes. For more than one choice, OR the constants together. Usual mode constants are:
GLUT_SINGLE - single buffering
GLUT_DOUBLE - double buffering
GLUT_RGB - set to full color mode
GLUT_RGBA - same as RGB
GLUT_INDEX - set color index mode

GLUT_DEPTH - set up a depth buffer

glutInitWindowSize(int width, int height) - sets size of screen window in pixels

glutInitWindowPosition(int x, int y) - sets initial window position on the screen for upper left corner

glutCreateWindow(char* title) - creates a top-level window labeled with title.

glutDisplayFunc(void (*func)(void)) - sets the callback function, provided by your program, that GLUT will call when a window must be redisplayed (e.g. on resize). Simply pass the name of a void function that takes no parameters.

glutMainLoop( ) - start the event loop; never returns.

Window Manager Input Handling Using glut

glutReshapeFunc(void (*func)(int width, int height)) - registers the callback function that GLUT calls on window reshape events.

glutKeyboardFunc(void (*func)(unsigned char key, int x, int y) - registers the callback function that GLUT calls on a keyboard event. The parameter key is the ASCII value of the key pressed. Parameters x and y indicate the mouse position at the time of the key press.

glutMouseFunc(void (*func)(int button, int state, int x, int y)) - registers the callback function that GLUT calls on a mouse event. The parameter button may be GLUT_LEFT_BUTTON, GLUT_MIDDLE_BUTTON, or GLUT_RIGHT_BUTTON (a two button mouse may not generate the middle value).
The parameter state may be GLUT_UP or GLUT_DOWN indicating button release or press, respectively. Parameters x and y indicate the mouse position.

glutPostRedisplay(void) - "nudges" the main loop to call display( ); sometimes useful after keyboard or mouse input changes some display parameters.

glFlush( ) - flush the graphics output buffer.

glutSwapBuffers(void) - Performs a buffer swap on the *layer in use* for the *current window*. Specifically, glutSwapBuffers promotes the contents of the back buffer of the *layer in use* of the *current window* to become the contents of the front buffer. The contents of the back buffer then become undefined. The update typically takes place during the vertical retrace of the monitor, rather than immediately after glutSwapBuffers is called.

Viewing and Modeling Transformations

glScalef(float x, float y, float z) - scale by (x, y, z). Also: glScaled( ).

glTranslatef(float x, float y, float z) - translate by (x, y, z).
Also: glTranslated( ).

glRotatef(float angle, float x, float y, float z) - rotate counterclockwise by angle degrees around the line through the origin with direction vector (x, y, z)

glMatrixMode(GLenum mode) - sets the matrix to be worked on. Values for mode are:
GL_MODELVIEW - use the modelview matrix
GL_PROJECTION - use the projection matrix
GL_TEXTURE - use the texture matrix

glLoadIdentity(void) - set the current matrix to the identity

Projection and Viewport Transformations

glOrtho(double left, double right, double bottom, double top, double near, double far) - sets the projection matrix to orthographic (parallel), with the near clipping plane defined by (left, bottom, -near) and (right, top, -near) and the far clipping plane defined by (left, bottom, -far) and (right, top, -far).

glViewport(GLint x, GLint y, GLsizei width, GLsizei height) - Parameters: *x*, *y* Specify the lower left corner of the viewport rectangle, in pixels. The initial value is (0,0). and Parameters: *width*, *height* Specify the width and height of the viewport. When a GL context is first attached to a window, *width* and *height* are set to the dimensions of that window. The glViewport function specifies the affine transformation

Color and Clearing the Screen

glClearColor(float red, float green, float blue, float alpha) - set the clear color to (red, green, blue) with value alpha.

glClear(GLbitfield mask) - clear the buffer indicated by mask using the current clear color. Values for mask are:
GL_COLOR_BUFFER_BIT - color buffer
GL_DEPTH_BUFFER_BIT - depth buffer
GL_ACCUM_BUFFER_BIT - accumulation buffer
GL_STENCIL_BUFFER_BIT - stencil buffer

glColor3f(float red, float green, float blue) - set the current drawing color to (red, green, blue).

OpenGL Drawing Primitives

glVertex{234}{sifd}{v}(…) - specify a vertex in either 2, 3, or 4-D coordinates, using either 16 or 32 bit integers or floats or doubles, and either listing all coordinates or using an array parameter.

glBegin(GLenum mode) - start a vertex list. Parameter mode can be:
GL_POINTS - single points
GL_LINES - pairs of vertices are a line segment
GL_LINE_STRIP - polyline
GL_LINE_LOOP - closed polyline
GL_TRIANGLES - triples indicate a triangle
GL_TRIANGLE_STRIP- linked strip of triangles
GL_TRIANGLE_FAN - linked fan of triangles
GL_QUADS - quadruples indicate a quadrilateral
GL_QUAD_STRIP - linked strip of quadrilaterals
GL_POLYGON - simple convex polygon

glEnd(void) - end of a vertex list

glPointSize(float size) - sets pixel width for a point; default is 1.0

glLineWidth(float width) - sets pixel width for a line; default is 1.0

glRectf(float x1, float y1, float x2, float y2) - draw a rectangle with corners (x1, y1) and (x2, y2).

glRasterPos{234}{sifd}{v}(…) - sets the current raster position (for what comes next) in either 2, 3, or 4-D coordinates (see glVertex( )).

glutBitmapCharacter(void* font, char c) - draws one bitmapped character using the indicated font; some fonts are: GLUT_BITMAP_TIMES_ROMAN_10 (also in 24), GLUT_BITMAP_HELVETICA_10 (also in 12 and 18).

Creating Menus in Glut

glutCreateMenu(void (*func) (int value)) - This function defines the callback that has to be called when a menu item was selected. This callback function has one parameter, the value. This function returns an int, this is the menu identifier. This identifier is needed when you would want to attach this menu as a submenu.

glutAddMenuEntry(char *name , int value) - This adds an entry to the menu with the label defined by name and the second parameter is the value that will be passed to the callback function. The menu is being added to the current menu. Each menu entry that is being added is added at the bottom of the current menu.

glutAttachMenu(int button) - This attaches the current menu to a certain (mouse) event, you can let a menu listen to a specified mouse button, button can be one of the following: GLUT_LEFT_BUTTON , GLUT_RIGHT_BUTTON and GLUT_RIGHT_BUTTON.

Matrix Stack in Glut

There is a stack of matrices for each of the matrix modes. In GL_MODELVIEW mode, the stack depth is at least 32. In the other modes, GL_COLOR, GL_PROJECTION, and GL_TEXTURE, the depth is at least 2. The current matrix in any mode is the matrix on the top of the stack for that mode.

glPushMatrix() - pushes the current matrix stack down by one, duplicating the current matrix. That is, after a glPushMatrix call, the matrix on top of the stack is identical to the one below it.

glPopMatrix() - pops the current matrix stack, replacing the current matrix with the one below it on the stack.

Timer Function in Glut

glutTimerFunc(unsigned int msecs void (*func)(int value), value) - glutTimerFunc registers the timer callback func to be triggered in at least msecs milliseconds. The value parameter to the timer callback will be the value of the value parameter to glutTimerFunc. Multiple timer callbacks at same or differing times may be registered simultaneously.

Chapter 6

SNAPSHOTS

6.1 INITIAL SCREEN

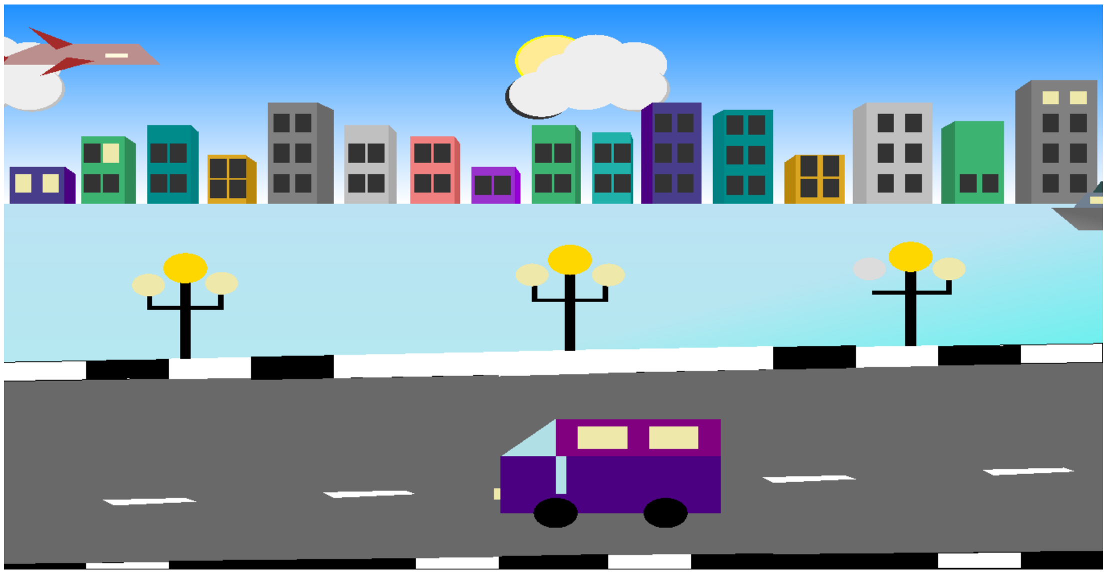


Fig 6.2 Initial screen

Fig 6.2 It include start state of the flight

6.4 MOVING SCREEN

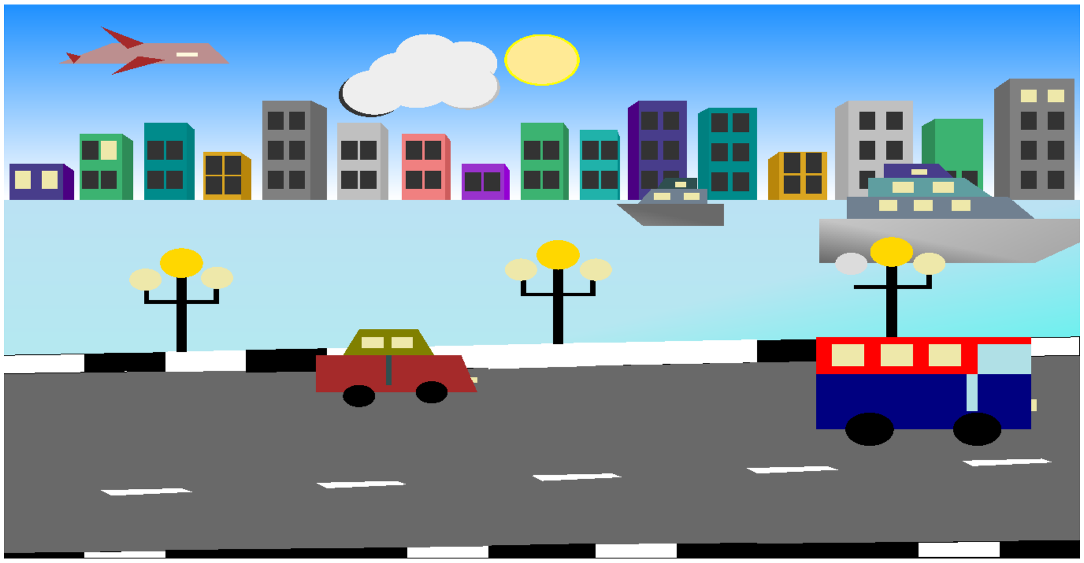
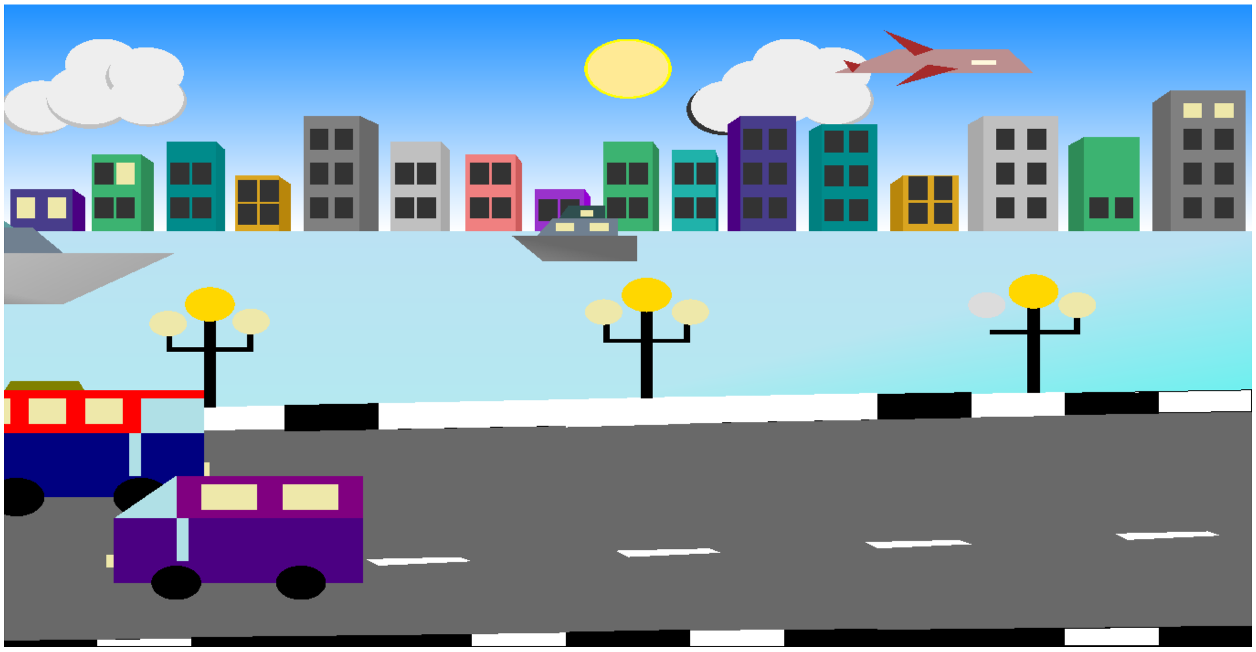
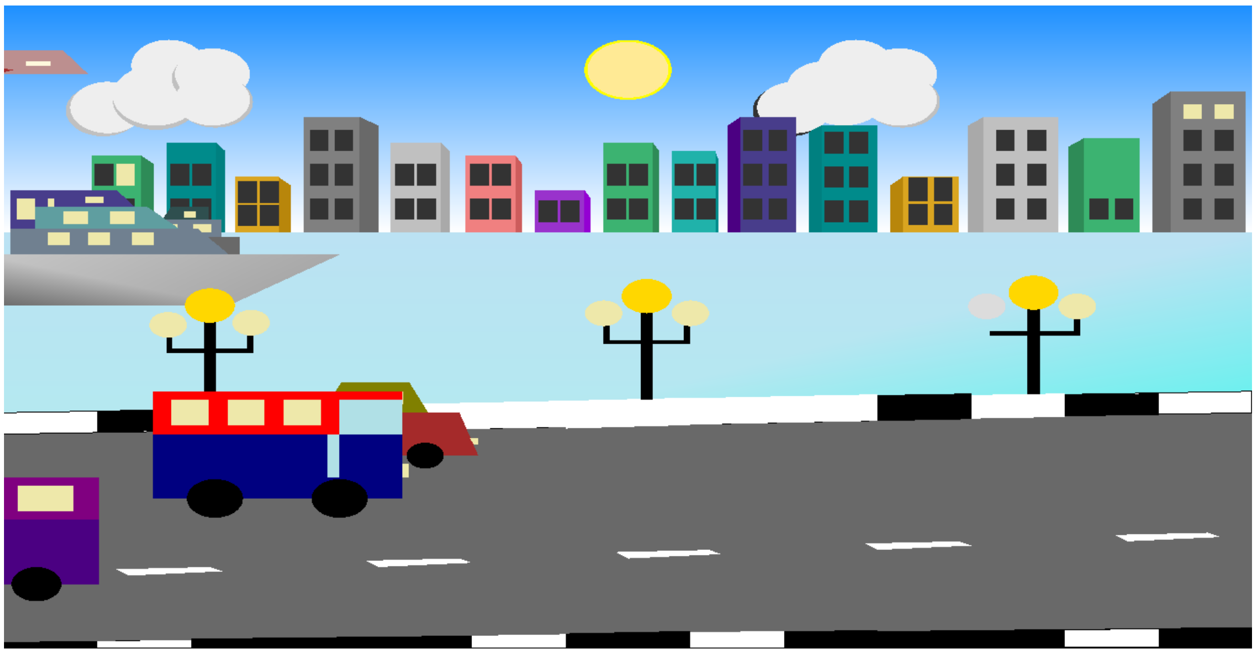


Fig 6.3 Moving Screen

Fig 6.3 It includes two vehicle’s and boat’s

6.5 MORE CLICKS





CONCLUSION

The project was started with modest aim with no prior experience in any programming projects as this, but ended up in learning many things. The project entitled "**A SMART CITY**" is developed with the best of our effort within a limited period of time. , furnished to attain perfection.
The functions in the package have been implemented and tested to ensure the efficiency of operation and they were found to be quite satisfactory.
After the completion of the development and study of the project I have come to the conclusion that computer graphics using OpenGL can be used to developing a much better and complex application that include 2D and 3D image processing.

REFERENCE

Reference Books and E-books

1. **WEBSITES**
   https://www.opengl.org//
   https://www.geeksforgeeks.org/c-programming-language/

2. **EDWARD ANGEL**
   Interactive Computer Graphics, 5th edition, universities of New Mexico.