JAIN UNIVERSITY

Declared as Deemed-to-be University u/s 3 of the UGC Act 1956

School of Engineering & Technology

Jain Global Campus, Kanakapura Taluk – 562 112

Ramanagara District, Karnataka, INDIA

2017-2018

A

Mini Project Report on

3D CAR PARKING

Submitted in partial fulfilment of the 5th semester

Computer Graphics & Visualization – 14CSL59

in

COMPUTER SCIENCE & ENGINEERING

Submitted by

Karthik C – 15BT6CS028.

Kanva L M – 15BT6CS027.

Lakshmi Srinivas Reddy – 15BT6CS029.

UNDER THE GUIDANCE OF:

Mr. Manjunath C R	**Sindhu Madhuri G**
Associate Professor	**Assistant Professor**

Department of Computer Science and Engineering
School of Engineering and Technology
JAIN University
Jakkasandra post, Kanakapura
July – December '17

JAIN UNIVERSITY

Declared as Deemed-to-be University u/s 3 of the UGC Act 1956

School of Engineering & Technology

Jain Global Campus, Kanakapura Taluk – 562 112

Ramanagara District, Karnataka, INDIA

2017-2018

CERTIFICATE

This is to certify that the Mini Project report of Computer Graphics & Visualization - 14CSL59 titled **"3D CAR PARKING "**is carried out by **Karthik C (15BT6CS028), Kanva L M (15BT6CS027), Lakshmi Srinivas Reddy (15BT6CS029)**, a bonafied student of 3rd year in the Department of Computer Science and Engineering, School of Engineering & Technology, Jain University, during the academic year 2017-2018.

Date: 08-12-2017

Signature of Faculty **Signature of HOD**

Table of Contents

Chapter 1: Introduction to OpenGL

1.1 Introduction

OpenGL is a low-level software interface to graphics hardware, No commands for performing windowing tasks or obtaining user input are included. No high-level commands for describing models of 3D objects are provided. Why low level? You can access graphics hardware directly Independent to window system, cross platform. High level enough, so it’s independent to graphics hardware and Industry standard

OpenGL is a library for doing computer graphics. We can create interactive applications which render high-quality color images composed of 3D geometric objects and images. OpenGL is window and operating system independent. The below figure 1.1 shows the OpenGL architecture which represents the flow of graphical information, as it is processed from CPU to the frame buffer.

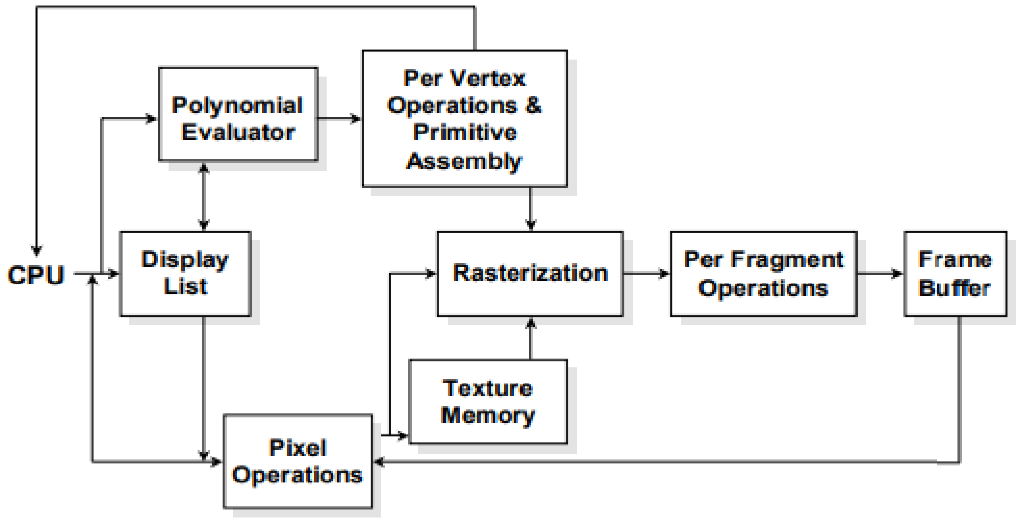


Figure 1.1: OpenGL Architecture

There are two pipelines of data flow. The upper pipeline is for geometric, vertex-based primitives. The lower pipeline is for pixel-based, image primitives. Texturing combines the two types of primitives together.

There are two operations that you do with OpenGL:
• Draw something
• Change the state of how OpenGL draws.

OpenGL has two types of things that it can render: geometric primitives and image primitives. Geometric primitives are points, lines and polygons. Image primitives are bitmaps and graphics. Additionally, OpenGL links image and geometric primitives together using texture mapping.

OpenGL is window and operating system independent. To integrate it into various window systems, additional libraries are used to modify a native window into an OpenGL capable window. Every window system has its own unique library and functions as shown in below figure 1.2:

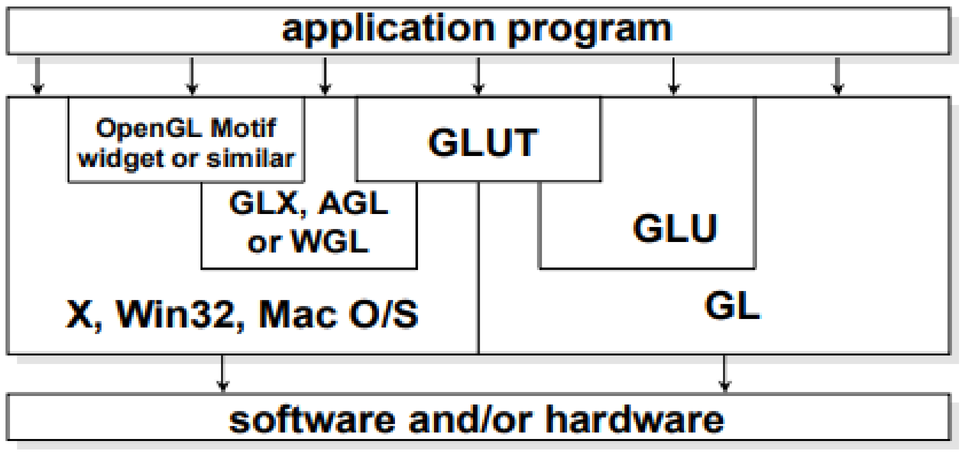


Figure 1.2: OpenGL Libraries

OpenGL also includes a utility library, GLU, to simplify common tasks such as:

- **OpenGL:** provides most of the graphics functionality.
- **GLU:** provides support for some additional operations and primitive types, and is implemented using OpenGL function calls.
- **glut:** designed specifically to be used with OpenGL and it takes care of things like opening windows, redraw events, and keyboard and mouse input. It effectively hides all the windowing system dependencies for OpenGL.

There are a few required elements which an application must do:
• **Header files**: describe all of the function calls, their parameters and defined constant values to the compiler. OpenGL has header files for GL (the core library), GLU (the utility library), and GLUT (freeware windowing toolkit).
Note: glut.h includes gl.h and glu.h. On Microsoft Windows, including only glut.h is recommended to avoid warnings about redefining Windows macros.
• **Libraries** are the operating system dependent implementation of OpenGL on the system you're using. Each operating system has its own set of libraries. For Unix systems, the OpenGL library is commonly named libGL. So and for Microsoft Windows, it's named opengl32.lib.•
Enumerated Types are definitions for the basic types (i.e., float, double, int, etc) whoch the program uses to store variables.

OpenGL provides:

- Draw with points, lines, and polygons.
- Matrix Operations (Transformations)
- Hidden Surface Removal (Z-Buffer)
- Lighting (Phong model)
- Gouraud Shading
- Texture mapping
- Pixels operations

The Buffers:

- A buffer is a memory area in the graphics hardware for some special purposes.
- An OpenGL system can manipulate the four buffers:
  - Color buffers
  - Depth buffer (Z-Buffer)
  - Stencil buffer
  - Accumulation buffer

OpenGL Execution Model:

- Streamlined
- State machine
- In-order execution

OpenGL Utility Toolkit (GLUT):

- A window system-independent toolkit to hide the complexity of differing window system APIs.
- Providing following operations:
  - Initializing and creating window
  - Handling window and input events
  - Drawing basic 3D objects
  - Running the program

GLUT Objects:

GLUT provides the follow objects in wireframe and Solid:

- Sphere
- Cube
- Torus
- Icosahedrons
- Ocrtahedron
- Tetrahedron
- Teapot
- Dodecahedron
- Cone

Ex: glutSolidSphere(1.0, 24, 24), glutWireCube(1.0)

Chapter 2: Problem Statement

The aim of this project is to create a 3-D/VIRTUAL CAR PARK. The viewer is allowed to roam around in the parking area and see the cars closely and to drive a car and park it in the car park area. The parking area is surrounded by a number of houses.

First the co-ordinates of the car is calculated and then using the OPENGL PRIMITIVES the car is constructed. The PRIMITIVES used are:-

1. GL_LINES
2. GL_POLYGON
3. GL_QUADS
4. GL_TRIANGLE

In a similar way the 3D house is constructed.

A display list is constructed for each of these objects. These display lists are used everytime a car or house has to be constructed.

So, to create the 36 cars in the parking lot the "carr_display_list " is called 36 times from within a loop and are translated each time by suitable values to place them correctly.

Similarly, to construct the houses "house_display_list" is called and are suitably translated, scaled and rotated to place them properly.

For the movement of camera GluLookAt( ) function is used .

The controls are:-

1.	UP KEY	- > to move the viewer in forward direction.
2.	DOWN KEY	- > to move the viewer in backwards direction.
3.	LEFT KEY	- > to rotate the camera to the left of the viewer.
4.	RIGHT KEY	- > to rotate the camera to the right of the viewer.
5.	T	- > top view.
6.	S	- > to move away.
7.	W	- >to move near.
8.	D	- > to move right.
9.	A	- > to move left.
10.	Q	- >quit.

Chapter 3: Design & Implementation

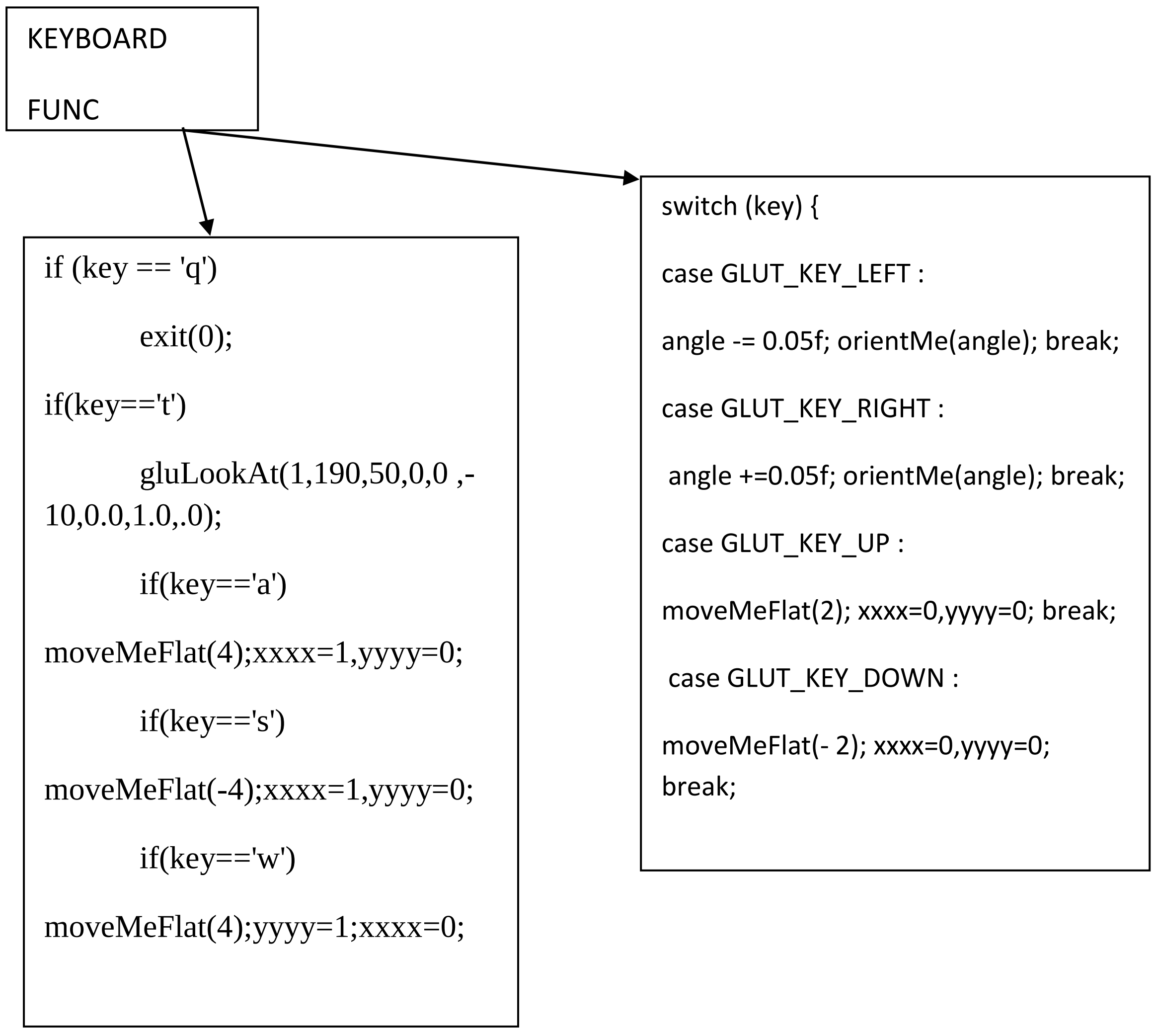
3.1: IMPLEMENTATION

3.1.1: Keyboard function.

3.1.2: Display function.

3.1.3: Reshape function.

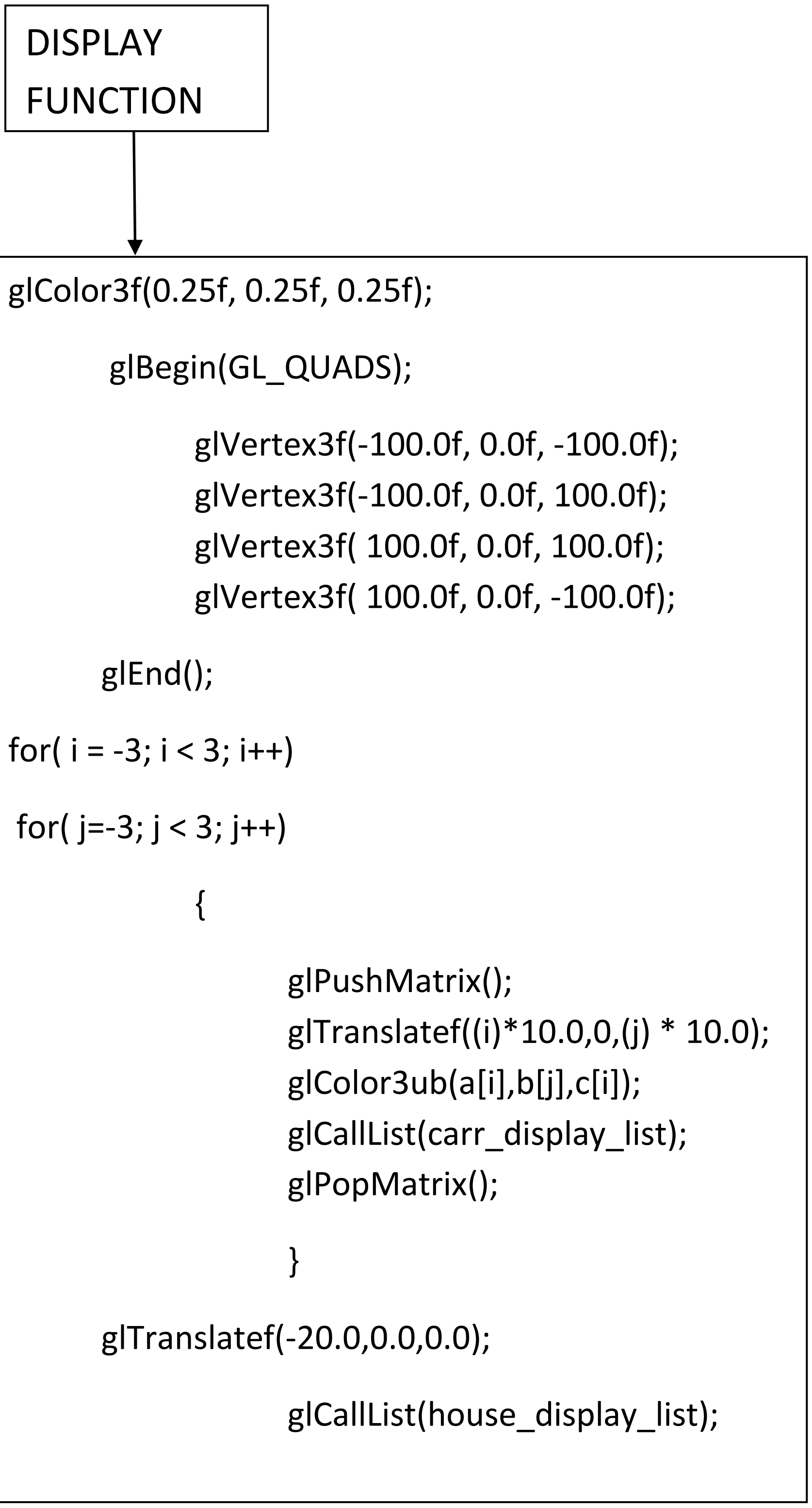
3.1.1: Keyboard Function.



```
if (key == 'q')

            exit(0);

if(key=='t')

            gluLookAt(1,190,50,0,0 ,-
10,0.0,1.0,.0);

            if(key=='a')

moveMeFlat(4);xxxx=1,yyyy=0;

            if(key=='s')

moveMeFlat(-4);xxxx=1,yyyy=0;

            if(key=='w')

moveMeFlat(4);yyyy=1;xxxx=0;
```

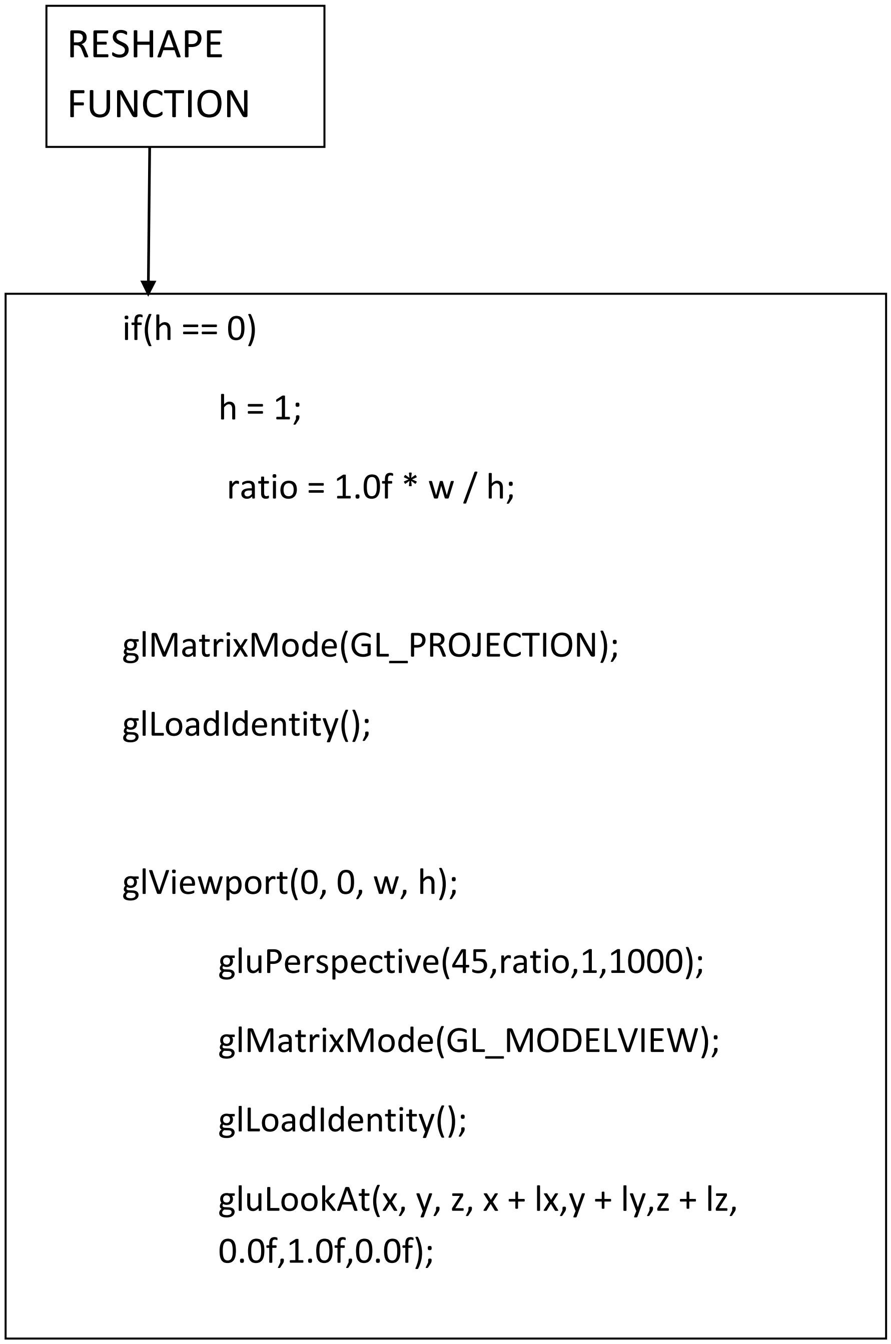
```
switch (key) {

case GLUT_KEY_LEFT :

angle -= 0.05f; orientMe(angle); break;

case GLUT_KEY_RIGHT :

 angle +=0.05f; orientMe(angle); break;

case GLUT_KEY_UP :

moveMeFlat(2); xxxx=0,yyyy=0; break;

 case GLUT_KEY_DOWN :

moveMeFlat(- 2); xxxx=0,yyyy=0;
break;
```

3.1.2: Display Function:



```
glColor3f(0.25f, 0.25f, 0.25f);

          glBegin(GL_QUADS);

                   glVertex3f(-100.0f, 0.0f, -100.0f);
                   glVertex3f(-100.0f, 0.0f, 100.0f);
                   glVertex3f( 100.0f, 0.0f, 100.0f);
                   glVertex3f( 100.0f, 0.0f, -100.0f);

         glEnd();

for( i = -3; i < 3; i++)

 for( j=-3; j < 3; j++)

                   {

                             glPushMatrix();
                             glTranslatef((i)*10.0,0,(j) * 10.0);
                             glColor3ub(a[i],b[j],c[i]);
                             glCallList(carr_display_list);
                             glPopMatrix();

                             }

         glTranslatef(-20.0,0.0,0.0);

                             glCallList(house_display_list);
```

3.1.3: Reshape Function:



```
if(h == 0)
        h = 1;
         ratio = 1.0f * w / h;


glMatrixMode(GL_PROJECTION);
glLoadIdentity();


glViewport(0, 0, w, h);
        gluPerspective(45,ratio,1,1000);
        glMatrixMode(GL_MODELVIEW);
        glLoadIdentity();
        gluLookAt(x, y, z, x + lx,y + ly,z + lz,
        0.0f,1.0f,0.0f);
```

3.2: DESIGN

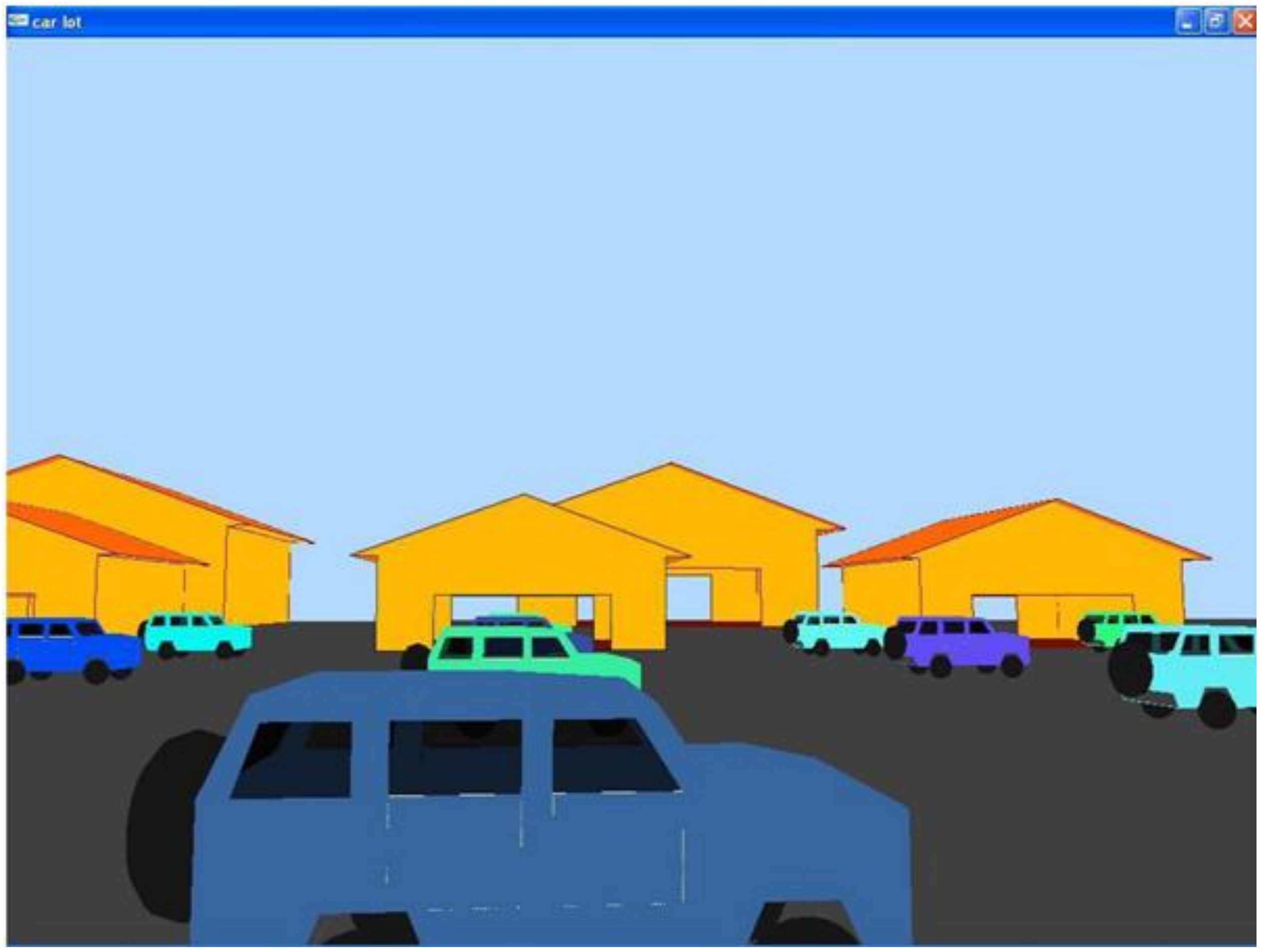
3.2.1: Initial Scene.

3.2.2: Front View.

3.2.3: Top View.

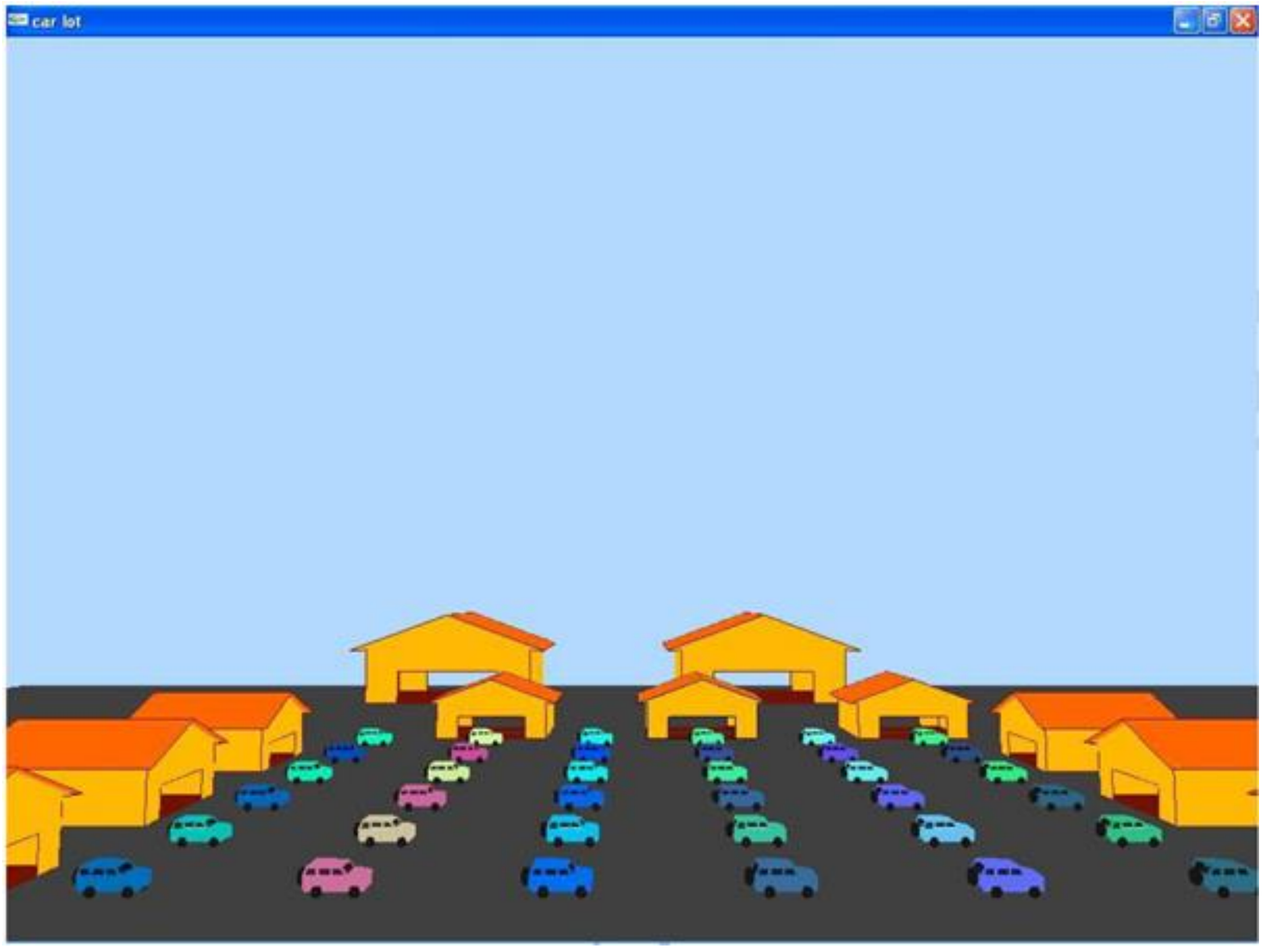
3.2.1: Initial Scene:

This is the first scene which appears when the program is executed.



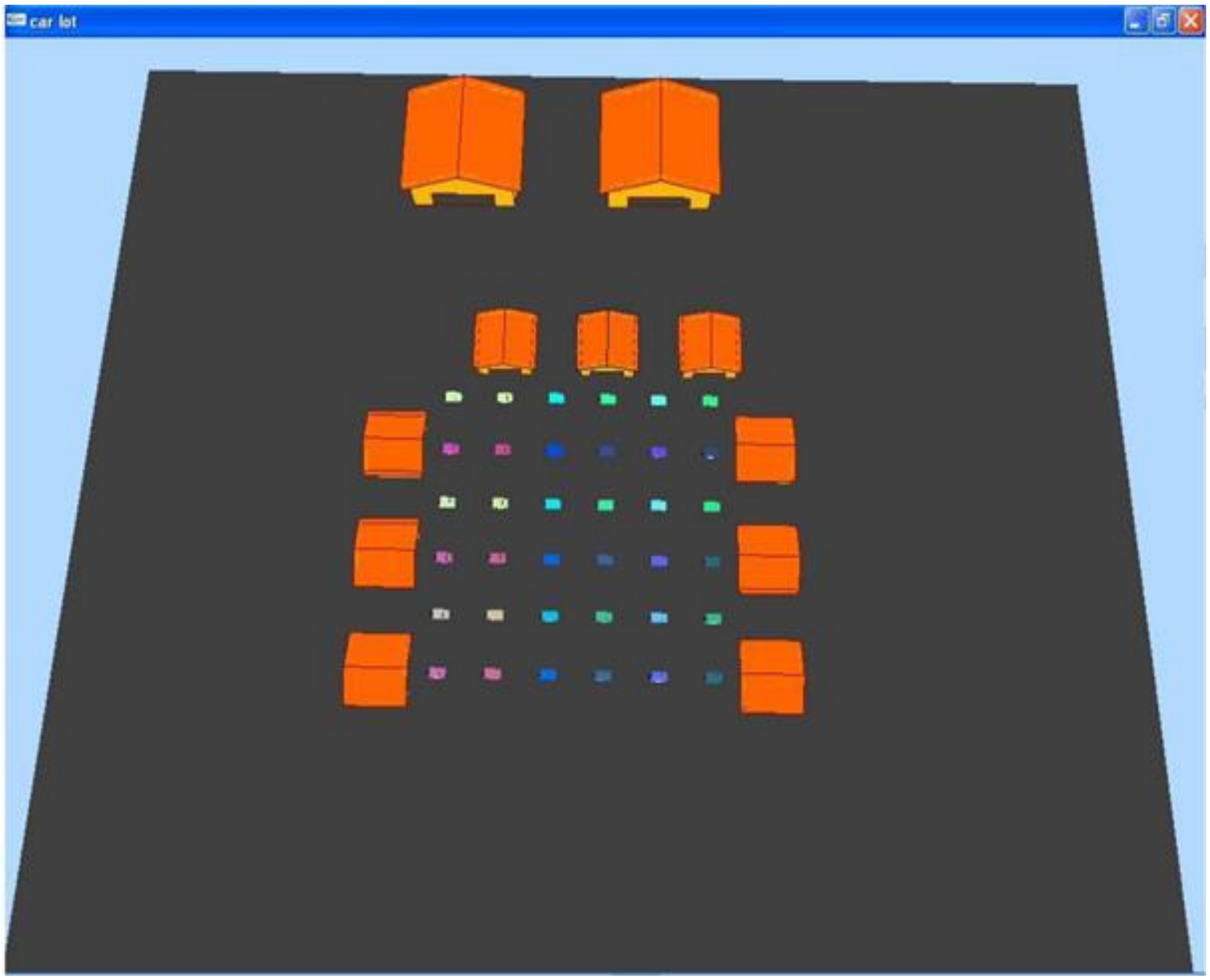
3.2.2: Front View:

On pressing the “S” key the camera moves backwards and upwards simultaneously. The user can press “W” key to move the camera in the front direction in the same way but the path traversed is exactly opposite.



3.2.3: Top View:

On pressing the “t” key the camera changes to a position so that the user can see the top view of the whole parking area.



Chapter 4: Conclusion and Future Scope

4.1: Conclusion

This project allows the user to rove in the parking lot and can even enter the houses that are present along the parking area. So, it’s like a virtualization of that area.

4.2: Future scope

1. **3-D MAP:**

   This project can be modified and a lot of other objects can be added for

example:-

   Trees , boundary walls, multiplexes ,roads etc.

   THUS A WHOLE CITY CAN BE CONSTRUCTED.

2. **GAME:**

   This program can be developed in to a fully-fledged game like Counter Strike, IGI etc

References

1. Interactive Computer Graphics by Edward Angel.
2. Vtucs.com/computer-graphics-projects/
3. en.wikipedia.org/wiki/OpenGL.
4. www.opengl.org.
5. www.glprogramming.com.