NITTE EDUCATION TRUST | N.M.A.M. INSTITUTE OF TECHNOLOGY
(An Autonomous Institution affiliated to Visvesvaraya Technological University, Belagavi)
Nitte – 574 110, Karnataka, India

(ISO 9001:2015 Certified), Accredited with 'A' Grade by NAAC
☎: 08258 - 281039 – 281263, Fax: 08258 – 281265

Department of Computer Science and Engineering

B.E. CSE Program Accredited by NBA, New Delhi from 1-7-2018 to 30-6-2021

Report on Mini Project

"SINEPOLIS"

Course Code: 20CS607

Course Name: Computer Graphics Lab

Semester: VI Section : A

Submitted To,

Dr. Pradeep Kanchan

Asst Prof Gd III,Dept of CSE,NMAMIT

Submitted By:

AMOGH SHET	4NM20CS024
ADARSH ACHARYA	4N20CS009

Date of submission: 12-5-23

Signature of Course Instructor

ABSTRACT

This project is a depiction of graphics created using computers, more generally the representation of image data. It is represented using different primitives available in OpenGL software package and combining them together in a required manner. The project is about the visualization of a 3D neighborhood, dubbed Sinepolis, with a fully functional & user operatable camera.

This project runs on OpenGL. OpenGL is an open-source graphics library widely used in computer graphics and game development. It provides a set of functions and capabilities for rendering 2D and 3D graphics on various platforms. This abstract explores the fundamental aspects and features of OpenGL.

One of the strengths of OpenGL lies in its ability to apply various visual effects and techniques to enhance the rendering process. It provides support for shading through the use of programmable shaders, allowing developers to customize the vertex and fragment processing stages. This flexibility enables the creation of realistic lighting models, textures, and special effects, resulting in visually appealing and immersive graphics.

Moreover, OpenGL facilitates interaction with user input by capturing events such as mouse movements and keyboard inputs. This enables developers to create interactive applications, such as games or simulations, where users can manipulate objects or navigate through virtual environments. OpenGL's cross-platform compatibility is another key advantage. It supports multiple operating systems, including Windows, macOS, and Linux, making it a versatile choice for developing graphics applications that can run on various platforms.

TABLE OF CONTENTS

INTRODUCTION

This CG project aims to create a virtual journey through a fully realized 3D neighborhood using the OpenGL graphics library in a 3D environment. Users can navigate through the various buildings and features of it. The project utilizes OpenGL's 3D rendering capabilities to create visually appealing and interactive elements, allowing users to explore the neighborhood as certain automated features (like the flying of birds) happen on their own.

The modularity, intractability & user freedom are the main highlights of this project. Given the code was written with these principles in mind from the ground up, one can easily extend the scope of this project without much hitch. Theres also the ground framework laid out for certain randomized elements, like grass, for future expansion. Several features can also be turned on or off (like cell shading or the skybox) in real time as per the user’s convenience. The combination of OpenGL's 3D rendering capabilities and a few creative tricks create a visually striking experience for users, allowing participants to virtually explore and appreciate the beauty of the neighborhood.

REQUIREMENTS

Software Requirements:

1. Operating System: Windows 10
2. Programming language: C/C++
3. Graphics library Used: OpenGL
4. IDE Used: Clion
5. Compiler Used: gcc, G++, mingw

Hardware Requirements:

1. 512MB SD/DDR RAM
2. Keyboard, mouse
3. 800 X 600 or higher resolution display with 256 colours

IMPLEMENTATION

1. Sketch out the scene: The first step in the implementation process is to sketch out the neighborhood in a sheet to determine the placement of each element of geometry.

2. Render the geometry: Use OpenGL function calls to issue draw commands that render the geometry to the viewport. This involves binding the appropriate buffers, setting up vertex attribute pointers, and issuing the draw call to render the geometry using the shader program.

3. Code individual props: Once the scene has been sketched out, the next step is to code each individual props such as houses, trees, clouds etc. using the GLUT library in C++ as their own individual function. This will require using basic shapes such as rectangles, circles, and triangles to create the polygons for each prop. For example, the house is consisted of rectangles and triangles.

4. Integrate props into the scene: After each prop has been coded, the next step is to integrate them into the overall scene. This will require positioning each prop accurately within the to create a cohesive environment.

5. Implement the camera: After this step comes coding the camera such that the 3D props can be viewed from multiple different angles as per the user input.

6. Handle input and update: Implement event handling to respond to user input& update the state of the application, whilst make any necessary changes to the rendering or geometry based on the input. This involves rendering or not rendering certain elements, or changing the camera angle. Also adding no user involved updates like the flight path of birds or the randomized location of the fish.

7. Fine-tuning: Finally, the scene will need to be fine-tuned to ensure that all elements are accurately represented and that the overall environment is cohesive. This may involve adjusting the position or size of certain elements, or tweaking lighting and texturing to create the desired atmosphere.

Below are the various functions used in this program:

- **glutInit()** : Interaction between the windowing system and OPENGL is initiated
- **glutCreateWindow()** : This opens the OPENGL window and displays the title at top of the window
- **glutInitWindowSize()** : Specifies the size of the window

- **glutInitWindowPosition()** : Specifies the position of the window in screen coordinates
- **glPushMatrix()** : Set current matrix on the stack
- **glPopMatrix()** : Pop the old matrix without the transformations
- **glutKeyboardFunc()** : Handles normal ascii symbols
- **glutDisplayFunc()** : This handles redrawing of the window
- **glutMainLoop()** : This starts the main loop, it never returns
- **glVertex3f()** : Used to set up the points or vertices in three dimensions
- **glColor3f()** : Used to render color to faces
- **glFlush()** : Used to flush the pipeline
- **glutPostRedisplay()** : Used to trigger an automatic redrawal of the object
- **glMatrixMode()** : Used to set up the required mode of the matrix
- **glLoadIdentity()** : Used to load or initialize to the identity matrix
- **sndPlaySound()** : Used to play music in the background

Below is the complete description against each key and what they do when pressed.:

• W:	Move Forward
• S:	Move Backward
• A:	Rotate Left
• D:	Rotate Right
• J/Q:	Strafe Left
• L/E:	Strafe Right
• I:	Move Up
• K:	Move Down
• 1:	Toggle Cell Shading
• 2:	Toggle Sky Props (such as clouds & sun)
• 3:	Toggle Sky Box
• 4:	Toggle Ground
• 5:	Toggle Road
• 6:	Toggle Creatures (such as the fish & birds)
• 7:	Toggle Pool
• 8:	Toggle Buildings
• =:	Toggle All On
• -:	Toggle All Off
• Backspace:	Reset Camera Position
• Escape:	Exit Program
• Enter:	Toggle Fullscreen

PSUEDOCODE

```
1. Include Libraries
2. Define global variables & flags
3. Define function to handle user input
4. Define functions to draw the various pieces of geometry & props
5. Define a drawScene function that utilizes all the prop functions
6. Define an update function to constantly update the scene
7. Define a main function, with all the necessary outputs for controls, and OpenGL
   declarations
8. Program is executed until terminated
```

SOURCE CODE

SINEPOLIS.CPP

```
#include <stdio.h>
#include <stdlib.h>
#include <GL/glut.h>
#include <math.h>
#include <string.h>
#include <windows.h>

int flag=0;
void *currentfont;
void myKeyboardFunc( unsigned char key, int x, int y );

void setFont(void *font)
{
   currentfont=font;
}

void drawstring(float x,float y,float z,char *string)
{
   char *c;
   int len = (int) strlen(string);
   int i;
   glRasterPos3f(x,y,z);
   for(i = 0;i<len;i++)
   {
      glColor3f(0.0,0.0,0.0);
      glutBitmapCharacter(currentfont,string[i]);
   }
}

void frontscreen(void)
{
   glClear(GL_COLOR_BUFFER_BIT|GL_DEPTH_BUFFER_BIT);
   setFont(GLUT_BITMAP_9_BY_15);
   glColor3f(1,0,0);
   drawstring(-22,80,0.0,"SIMOPOLIS 3D SIMULATION");
   glColor3f(0,0,1);
   drawstring(-23,76,0.0," ~~~~~~~~~~~~~~~~~~~~~~~");
```

```
   drawstring(-65,72,0.0,"A Computer Graphics Mini Project by Amogh Shet (4NM20CS024)
& Adarsh Acharya (4NM20CS009)");
   glColor3f(1,0.5,0);
   drawstring(-20,30,0.0," Under the Guidance of:");
   glColor3f(0.5,0.2,0.2);
   drawstring(-85,24,0.0," Dr. Pradeep Kanchan");
   //glColor3f(0.5,0.2,0.2);
   drawstring(-85,20,0.0," Assistant Professor Gd-III");
   //glColor3f(0.5,0.2,0.2);
   drawstring(-85,16,0.0," NMAMIT, Nitte");
   glColor3f(0.5,0.2,0.2);
   drawstring(40,24,0.0," Mr. Puneeth RP");
   //glColor3f(0.5,0.2,0.2);
   drawstring(40,20,0.0," Assistant Professor Gd-II");
   //glColor3f(0.5,0.2,0.2);
   drawstring(40,16,0.0," NMAMIT, Nitte");
   glColor3f(1,0,0);
   drawstring(-11,-32,0.0,"WEEK FINAL");
   glColor3f(1,0.1,1);
   drawstring(-18,-40,0.0,"PRESS ENTER FOR DEMO");
   glutSwapBuffers();
   glFlush();
}

void mydisplay(void)
{
        glClear(GL_COLOR_BUFFER_BIT);
        if(flag==0)
        {
                frontscreen();
        }
        if(flag==1)
   {
                system("start D:\\CodeBlocks\\TDM-
GCC\\bin\\housetest\\bin\\Debug\\housetest.exe");
                flag = 0;
                exit(0);
   }
}

void myKeyboardFunc( unsigned char key, int x, int y )
{
   switch(key)
   {
        case 13:
            if(flag==0) //Ascii of 'enter' key is 13
                flag=1;
            else
                flag=0;
            break;

        case 27:        //Escape key
            exit(0);
    }
    mydisplay();
}

int main(int argc, char **argv)
{
    printf(" -----------------------------------------------------\n");
    printf("|SIMOPOLIS: A MINIPROJECT BY AMOGH SHET & ADARSH ACHARYA|\n");
    printf(" -----------------------------------------------------");
    printf("\n\n\n");
    printf("Done under the guidance of:\n\n");
    printf("    Dr. Pradeep Kanchan\n");
    printf("    Assistant Professor Gd-III\n");
    printf("    NMAMIT, Nitte\n\n");
    printf("    Mr. Puneeth RP\n");
    printf("    Assistant Professor Gd-II\n");
    printf("    NMAMIT, Nitte\n\n\n");
    printf("Week Final\n");
    printf("Press 'Enter' to begin or 'Escape' to exit\n");
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE|GLUT_RGBA|GLUT_DEPTH);
    glutInitWindowSize(1920, 800);
    glutCreateWindow("SINOPOLIS: CITYSIM");
    glutDisplayFunc(mydisplay);
    glutKeyboardFunc(myKeyboardFunc);
    glClearColor(0.95,0.95,0.95,0.0);
    glOrtho(-100.0,100.0,-100.0,100.0,-50.0,50.0);
    glEnable(GL_DEPTH_TEST);
    glutMainLoop();
    return 0;
}
```

HOUSETEST.CPP

```
#include <iostream>
#include <stdlib.h>
#include <GL/glut.h>
#include <windows.h>
#include <mmsystem.h>
#include <cstdlib>
#include <time.h>

float _angle = 45.0f;
float _cameraAngle = 0.0f;
float _zangle = 1.0f;
float _otherangle = 0.0f;
float _oangle = 1.0f;
float sfactor = 1.0f;
int scaleflag = 1; //1 = reduce, 0 = incrase
float speed = 0.005;
float anglefactor = .25f;

float birdxz = -40;

int xgrassx[999] = {0};
int xgrassz[999] = {0};

int groundflag = 1;
int skyboxflag = 1;
int skypropflag = 1;
int roadflag = 1;
int buildflag = 1;
int poolflag = 1;
int creatureflag = 1;
int cellshadeflag = 1;
int fullscreenflag = -1;


void *currentfont;
void setFont(void *font)
{
   currentfont=font;
}

void drawstring(float x,float y,float z,char *string)
{
   char *c;
   int len = (int) strlen(string);
   int i;
   glRasterPos3f(x,y,z);
   for(i = 0;i<len;i++)
   {
       glColor3f(0.0,0.0,0.0);
       glutBitmapCharacter(currentfont,string[i]);
   }
}

using namespace std;

void handleKeypress(unsigned char key, int x, int y) {
   switch (key) {
   case 27:
       system("start D:\\CodeBlocks\\TDM-GCC\\bin\\Week2\\bin\\Debug\\Week2.exe");
       exit(0);

   case 'w':
   case 'W':
       _otherangle += 0.075;
       //sfactor += 0.075;
       break;

   case 's':
   case 'S':
       _otherangle -= 0.075;
       //if(sfactor > 0.03)
           //sfactor -= 0.075;
       break;

   case 'd':
   case 'D':
       anglefactor -= 5;
       break;

   case 'a':
   case 'A':
       anglefactor += 5;
       break;

   case 'l':
   case 'L':
```

```
case 'e':
case 'E':
   _oangle -= 0.05;
   break;

case 'j':
case 'J':
case 'q':
case 'Q':
   _oangle += 0.05;
   break;

case 'I':
case 'i':
   _zangle -= 0.1;
   break;

case 'k':
case 'K':
   _zangle += 0.1;
   break;

case 8:
   _cameraAngle = 0.0f;
   _zangle = 1.0f;
   _otherangle = 0.0f;
   _oangle = 1.0f;
   sfactor = 1.0f;
   scaleflag = 1; //1 = reduce, 0 = incrase
   speed = 0.005;
   anglefactor = .25f;
   break;

case '1':
   cellshadeflag *= -1;
   break;

case '2':
   skypropflag *= -1;
   break;

case '3':
   skyboxflag *= -1;
   break;
```

```
case '4':
   groundflag *= -1;
   break;

case '5':
   roadflag *= -1;
   break;

case '6':
   creatureflag *= -1;
   break;

case '7':
   poolflag *= -1;
   break;

case '8':
   buildflag *= -1;
   break;


case '=':
   groundflag = 1;
   skyboxflag = 1;
   skypropflag = 1;
   roadflag = 1;
   buildflag = 1;
   poolflag = 1;
   creatureflag = 1;
   cellshadeflag = 1;
   break;

case '-':
   groundflag = -1;
   skyboxflag = -1;
   skypropflag = -1;
   roadflag = -1;
   buildflag = -1;
   poolflag = -1;
   creatureflag = -1;
   cellshadeflag = -1;
   break;
```

```
    case 13:
        fullscreenflag *= -1;
        if(fullscreenflag == 0)
        {
            fullscreenflag = 1;
        }
        break;
    }
}

void initRendering()
{
    glEnable(GL_DEPTH_TEST);
}

void handleResize(int w, int h) {
    glViewport(0, 0, w, h);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluPerspective(45.0, (double) w / (double) h, 1.0, 200.0);
}

//flx = front left x value
//fby = front bottom y value
//ty = top y value
void drawHouse(float flx, float fz)
{
    float fby = 0.0;
    float ty = fby + 0.7;

    //THE HOUSE

        //Front
        glColor3f (.5, .75, .35);
        glBegin(GL_POLYGON);
        glVertex3f (flx, fby, fz);
        glVertex3f (flx+0.7, fby, fz);
        glVertex3f (flx+0.7, fby+0.475, fz);
        glVertex3f (flx, fby+0.475, fz);
        glEnd();

        //Back
        glColor3f (.5, .75, .35);
        glBegin(GL_POLYGON);
```

```
glVertex3f (flx, fby, fz+0.5);
glVertex3f (flx+0.7, fby, fz+0.5);
glVertex3f (flx+0.7, fby+0.475, fz+0.5);
glVertex3f (flx, fby+0.475, fz+0.5);
glEnd();

//Left
glColor3f (.75, 0.75, .25);
glBegin(GL_POLYGON);
glVertex3f (flx, fby, fz+0.5);
glVertex3f (flx, fby, fz);
glVertex3f (flx, fby+0.475, fz);
glVertex3f (flx, fby+0.475, fz+0.5);
glEnd();

//Right
glColor3f (.75, 0.75, .25);
glBegin(GL_POLYGON);
glVertex3f (flx+0.7, fby, fz+0.5);
glVertex3f (flx+0.7, fby, fz);
glVertex3f (flx+0.7, fby+0.475, fz);
glVertex3f (flx+0.7, fby+0.475, fz+0.5);
glEnd();

//Right roof
glColor3f (.4, 0.4, .15);
glBegin(GL_TRIANGLES);
glVertex3f (flx+0.7, fby+0.475, fz);
glVertex3f (flx+0.7, fby+0.475, fz+0.5);
glVertex3f (flx+0.7, ty, fz+0.25);
glEnd();

//Left roof
glColor3f (.4, 0.4, .15);
glBegin(GL_TRIANGLES);
glVertex3f (flx, fby+0.475, fz);
glVertex3f (flx, fby+0.475, fz+0.5);
glVertex3f (flx, ty, fz+0.25);
glEnd();

//Front roof
glColor3f (.55, 0.35, .2);
glBegin(GL_POLYGON);
glVertex3f (flx, fby+0.475, fz);
```

```
glVertex3f (flx+0.7, fby+0.475, fz);
glVertex3f (flx+0.7, ty, fz+0.25);
glVertex3f (flx, ty, fz+0.25);
glEnd();

//Back roof
glColor3f (.55, 0.35, .2);
glBegin(GL_POLYGON);
glVertex3f (flx, fby+0.475, fz+0.5);
glVertex3f (flx+0.7, fby+0.475, fz+0.5);
glVertex3f (flx+0.7, ty, fz+0.25);
glVertex3f (flx, ty, fz+0.25);
glEnd();

//Door
glColor3f (.15, 0.2, .3);
glBegin(GL_POLYGON);
glVertex3f (flx+0.27, fby+0.005, fz-.0001);
glVertex3f (flx+0.45, fby+0.005, fz-.0001);
glVertex3f (flx+0.45, fby+0.36, fz-.0001);
glVertex3f (flx+0.27, fby+0.36, fz-.0001);
glEnd();

//Window 1
glColor3f (.5, .3, .5);
glBegin(GL_POLYGON);
glVertex3f (flx+0.52, fby+0.15, fz-.0001);
glVertex3f (flx+0.63, fby+0.15, fz-.0001);
glVertex3f (flx+0.63, fby+0.3, fz-.0001);
glVertex3f (flx+0.52, fby+0.3, fz-.0001);
glEnd();

//Window 2
glColor3f (.5, 0.3, .5);
glBegin(GL_POLYGON);
glVertex3f (flx+0.07, fby+0.15, fz-.0001);
glVertex3f (flx+0.18, fby+0.15, fz-.0001);
glVertex3f (flx+0.18, fby+0.3, fz-.0001);
glVertex3f (flx+0.07, fby+0.3, fz-.0001);
glEnd();

//TreeTrunk
glColor3f(0.5f, 0.35f, 0.05f);
glBegin(GL_POLYGON);
```

```
    glVertex3f (flx-0.5, fby, fz);
    glVertex3f (flx-0.6, fby, fz);
    glVertex3f (flx-0.55, fby, fz+0.1);
    glVertex3f (flx-0.5, fby+1.0, fz);
    glEnd();

    //Tree Branches
    glColor3f(.0, .6, .0);
    glBegin(GL_POLYGON);
    glVertex3f (flx-0.5, fby+1.0, fz);
    glVertex3f (flx-0.5, fby+1.0, fz+0.3);
    glVertex3f (flx-0.5, fby+1.0, fz-0.3);
    glVertex3f (flx-0.9, fby+1.0, fz);
    glVertex3f (flx-0.1, fby+1.0, fz);
    glVertex3f (flx-0.5, fby+1.5, fz);
    glVertex3f (flx-0.5, fby+1.0, fz+0.3);
    glEnd();


  //OUTLINE
    if(cellshadeflag == 1)
    {
      glLineWidth(1.5);
      //Front
      glColor3f (0, 0, 0);
      glBegin(GL_LINE_LOOP);
      glVertex3f (flx, fby, fz);
      glVertex3f (flx+0.7, fby, fz);
      glVertex3f (flx+0.7, fby+0.475, fz);
      glVertex3f (flx, fby+0.475, fz);
      glEnd();

      //Back
      glColor3f (0, 0, 0);
      glBegin(GL_LINE_LOOP);
      glVertex3f (flx, fby, fz+0.5);
      glVertex3f (flx+0.7, fby, fz+0.5);
      glVertex3f (flx+0.7, fby+0.475, fz+0.5);
      glVertex3f (flx, fby+0.475, fz+0.5);
      glEnd();

      //Left
      glColor3f (0, 0, 0);
      glBegin(GL_LINE_LOOP);
```

```
glVertex3f (flx, fby, fz+0.5);
glVertex3f (flx, fby, fz);
glVertex3f (flx, fby+0.475, fz);
glVertex3f (flx, fby+0.475, fz+0.5);
glEnd();

//Right
glColor3f (0, 0, 0);
glBegin(GL_LINE_LOOP);
glVertex3f (flx+0.7, fby, fz+0.5);
glVertex3f (flx+0.7, fby, fz);
glVertex3f (flx+0.7, fby+0.475, fz);
glVertex3f (flx+0.7, fby+0.475, fz+0.5);
glEnd();

//Right roof
glColor3f (0, 0, 0);
glBegin(GL_LINE_LOOP);
glVertex3f (flx+0.7, fby+0.475, fz);
glVertex3f (flx+0.7, fby+0.475, fz+0.5);
glVertex3f (flx+0.7, ty, fz+0.25);
glEnd();

//Left roof
glColor3f (0, 0, 0);
glBegin(GL_LINE_LOOP);
glVertex3f (flx, fby+0.475, fz);
glVertex3f (flx, fby+0.475, fz+0.5);
glVertex3f (flx, ty, fz+0.25);
glEnd();

//Front roof
glColor3f (0, 0, 0);
glBegin(GL_LINE_LOOP);
glVertex3f (flx, fby+0.475, fz);
glVertex3f (flx+0.7, fby+0.475, fz);
glVertex3f (flx+0.7, ty, fz+0.25);
glVertex3f (flx, ty, fz+0.25);
glEnd();

//Back roof
glColor3f (0, 0, 0);
glBegin(GL_LINE_LOOP);
glVertex3f (flx, fby+0.475, fz+0.5);
```

```
glVertex3f (flx+0.7, fby+0.475, fz+0.5);
glVertex3f (flx+0.7, ty, fz+0.25);
glVertex3f (flx, ty, fz+0.25);
glEnd();

//Door
glColor3f (0, 0, 0);
glBegin(GL_LINE_LOOP);
glVertex3f (flx+0.27, fby+0.005, fz-.0001);
glVertex3f (flx+0.45, fby+0.005, fz-.0001);
glVertex3f (flx+0.45, fby+0.36, fz-.0001);
glVertex3f (flx+0.27, fby+0.36, fz-.0001);
glEnd();

//Window 1
glColor3f (0, 0, 0);
glBegin(GL_LINE_LOOP);
glVertex3f (flx+0.52, fby+0.15, fz-.0001);
glVertex3f (flx+0.63, fby+0.15, fz-.0001);
glVertex3f (flx+0.63, fby+0.3, fz-.0001);
glVertex3f (flx+0.52, fby+0.3, fz-.0001);
glEnd();

//Window 2
glColor3f (0, 0, 0);
glBegin(GL_LINE_LOOP);
glVertex3f (flx+0.07, fby+0.15, fz-.0001);
glVertex3f (flx+0.18, fby+0.15, fz-.0001);
glVertex3f (flx+0.18, fby+0.3, fz-.0001);
glVertex3f (flx+0.07, fby+0.3, fz-.0001);
glEnd();

//TreeTrunk
glColor3f(0, 0, 0);
glBegin(GL_LINE_LOOP);
glVertex3f (flx-0.5, fby, fz);
glVertex3f (flx-0.6, fby, fz);
glVertex3f (flx-0.55, fby, fz+0.1);
glVertex3f (flx-0.5, fby+1.0, fz);
glEnd();

//Tree Branches
glColor3f(0, 0, 0);
glBegin(GL_LINE_LOOP);
```

```
            glVertex3f (flx-0.5, fby+1.0, fz);
            glVertex3f (flx-0.5, fby+1.0, fz+0.3);
            glVertex3f (flx-0.5, fby+1.0, fz-0.3);
            glVertex3f (flx-0.9, fby+1.0, fz);
            glVertex3f (flx-0.1, fby+1.0, fz);
            glVertex3f (flx-0.5, fby+1.5, fz);
            glVertex3f (flx-0.5, fby+1.0, fz+0.3);
            glEnd();
        }
}

void drawOffice(float x, float z)
{
    float y = x;

    //THE OFFICE
        //Front
        glColor3f(.5, .5, .5);
        glBegin(GL_POLYGON);
        glVertex3f (x, y, z);
        glVertex3f (x+0.7, y, z);
        glVertex3f (x+0.7, y+1.7, z);
        glVertex3f (x, y+1.7, z);
        glEnd();

        //Back
        glColor3f(.5, .5, .5);
        glBegin(GL_POLYGON);
        glVertex3f (x, y, z+0.5);
        glVertex3f (x+0.7, y, z+0.5);
        glVertex3f (x+0.7, y+1.7, z+0.5);
        glVertex3f (x, y+1.7, z+0.5);
        glEnd();

        //Left
        glColor3f(.5, .5, .5);
        glBegin(GL_POLYGON);
        glVertex3f (x, y, z);
        glVertex3f (x, y, z+0.5);
        glVertex3f (x, y+1.7, z+0.5);
        glVertex3f (x, y+1.7, z);
        glEnd();

        //Right
```

```
glColor3f(.5, .5, .5);
glBegin(GL_POLYGON);
glVertex3f (x+0.7, y, z);
glVertex3f (x+0.7, y, z+0.5);
glVertex3f (x+0.7, y+1.7, z+0.5);
glVertex3f (x+0.7, y+1.7, z);
glEnd();

//Windows Rightmost
glColor3f(1, 1, 1);
glBegin(GL_POLYGON);
glVertex3f (x+0.07, y+.21, z-.0001);
glVertex3f (x+0.07, y+.28, z-.0001);
glVertex3f (x+.14, y+.28, z-.0001);
glVertex3f (x+.14, y+.21, z-.0001);
glEnd();
glColor3f(1, 1, 1);
glBegin(GL_POLYGON);
glVertex3f (x+0.07, y+.36, z-.0001);
glVertex3f (x+0.07, y+.43, z-.0001);
glVertex3f (x+.14, y+.43, z-.0001);
glVertex3f (x+.14, y+.36, z-.0001);
glEnd();
glColor3f(1, 1, 1);
glBegin(GL_POLYGON);
glVertex3f (x+0.07, y+.51, z-.0001);
glVertex3f (x+0.07, y+.58, z-.0001);
glVertex3f (x+.14, y+.58, z-.0001);
glVertex3f (x+.14, y+.51, z-.0001);
glEnd();
glColor3f(1, 1, 1);
glBegin(GL_POLYGON);
glVertex3f (x+0.07, y+.66, z-.0001);
glVertex3f (x+0.07, y+.73, z-.0001);
glVertex3f (x+.14, y+.73, z-.0001);
glVertex3f (x+.14, y+.66, z-.0001);
glEnd();
glColor3f(1, 1, 1);
glBegin(GL_POLYGON);
glVertex3f (x+0.07, y+.81, z-.0001);
glVertex3f (x+0.07, y+.88, z-.0001);
glVertex3f (x+.14, y+.88, z-.0001);
glVertex3f (x+.14, y+.81, z-.0001);
glEnd();
```

```
glColor3f(1, 1, 1);
glBegin(GL_POLYGON);
glVertex3f (x+0.07, y+.96, z-.0001);
glVertex3f (x+0.07, y+1.03, z-.0001);
glVertex3f (x+.14, y+1.03, z-.0001);
glVertex3f (x+.14, y+.96, z-.0001);
glEnd();
glColor3f(1, 1, 1);
glBegin(GL_POLYGON);
glVertex3f (x+0.07, y+1.11, z-.0001);
glVertex3f (x+0.07, y+1.18, z-.0001);
glVertex3f (x+.14, y+1.18, z-.0001);
glVertex3f (x+.14, y+1.11, z-.0001);
glEnd();
glColor3f(1, 1, 1);
glBegin(GL_POLYGON);
glVertex3f (x+0.07, y+1.26, z-.0001);
glVertex3f (x+0.07, y+1.33, z-.0001);
glVertex3f (x+.14, y+1.33, z-.0001);
glVertex3f (x+.14, y+1.26, z-.0001);
glEnd();
glColor3f(1, 1, 1);
glBegin(GL_POLYGON);
glVertex3f (x+0.07, y+1.41, z-.0001);
glVertex3f (x+0.07, y+1.48, z-.0001);
glVertex3f (x+.14, y+1.48, z-.0001);
glVertex3f (x+.14, y+1.41, z-.0001);
glEnd();
glColor3f(1, 1, 1);
glBegin(GL_POLYGON);
glVertex3f (x+0.07, y+1.56, z-.0001);
glVertex3f (x+0.07, y+1.63, z-.0001);
glVertex3f (x+.14, y+1.63, z-.0001);
glVertex3f (x+.14, y+1.56, z-.0001);
glEnd();


//Windows Leftmost
glColor3f(1, 1, 1);
glBegin(GL_POLYGON);
glVertex3f (x+0.55, y+.21, z-.0001);
glVertex3f (x+0.55, y+.28, z-.0001);
glVertex3f (x+.62, y+.28, z-.0001);
glVertex3f (x+.62, y+.21, z-.0001);
```

```
glEnd();
glColor3f(1, 1, 1);
glBegin(GL_POLYGON);
glVertex3f (x+0.55, y+.36, z-.0001);
glVertex3f (x+0.55, y+.43, z-.0001);
glVertex3f (x+.62, y+.43, z-.0001);
glVertex3f (x+.62, y+.36, z-.0001);
glEnd();
glColor3f(1, 1, 1);
glBegin(GL_POLYGON);
glVertex3f (x+0.55, y+.51, z-.0001);
glVertex3f (x+0.55, y+.58, z-.0001);
glVertex3f (x+.62, y+.58, z-.0001);
glVertex3f (x+.62, y+.51, z-.0001);
glEnd();
glColor3f(1, 1, 1);
glBegin(GL_POLYGON);
glVertex3f (x+0.55, y+.66, z-.0001);
glVertex3f (x+0.55, y+.73, z-.0001);
glVertex3f (x+.62, y+.73, z-.0001);
glVertex3f (x+.62, y+.66, z-.0001);
glEnd();
glColor3f(1, 1, 1);
glBegin(GL_POLYGON);
glVertex3f (x+0.55, y+.81, z-.0001);
glVertex3f (x+0.55, y+.88, z-.0001);
glVertex3f (x+.62, y+.88, z-.0001);
glVertex3f (x+.62, y+.81, z-.0001);
glEnd();
glColor3f(1, 1, 1);
glBegin(GL_POLYGON);
glVertex3f (x+0.55, y+.96, z-.0001);
glVertex3f (x+0.55, y+1.03, z-.0001);
glVertex3f (x+.62, y+1.03, z-.0001);
glVertex3f (x+.62, y+.96, z-.0001);
glEnd();
glColor3f(1, 1, 1);
glBegin(GL_POLYGON);
glVertex3f (x+0.55, y+1.11, z-.0001);
glVertex3f (x+0.55, y+1.18, z-.0001);
glVertex3f (x+.62, y+1.18, z-.0001);
glVertex3f (x+.62, y+1.11, z-.0001);
glEnd();
glColor3f(1, 1, 1);
```

```
glBegin(GL_POLYGON);
glVertex3f (x+0.55, y+1.26, z-.0001);
glVertex3f (x+0.55, y+1.33, z-.0001);
glVertex3f (x+.62, y+1.33, z-.0001);
glVertex3f (x+.62, y+1.26, z-.0001);
glEnd();
glColor3f(1, 1, 1);
glBegin(GL_POLYGON);
glVertex3f (x+0.55, y+1.41, z-.0001);
glVertex3f (x+0.55, y+1.48, z-.0001);
glVertex3f (x+.62, y+1.48, z-.0001);
glVertex3f (x+.62, y+1.41, z-.0001);
glEnd();
glColor3f(1, 1, 1);
glBegin(GL_POLYGON);
glVertex3f (x+0.55, y+1.56, z-.0001);
glVertex3f (x+0.55, y+1.63, z-.0001);
glVertex3f (x+.62, y+1.63, z-.0001);
glVertex3f (x+.62, y+1.56, z-.0001);
glEnd();


//Windows Rightinner
glColor3f(1, 1, 1);
glBegin(GL_POLYGON);
glVertex3f (x+0.23, y+.36, z-.0001);
glVertex3f (x+0.23, y+.43, z-.0001);
glVertex3f (x+.3, y+.43, z-.0001);
glVertex3f (x+.3, y+.36, z-.0001);
glEnd();
glColor3f(1, 1, 1);
glBegin(GL_POLYGON);
glVertex3f (x+0.23, y+.51, z-.0001);
glVertex3f (x+0.23, y+.58, z-.0001);
glVertex3f (x+.3, y+.58, z-.0001);
glVertex3f (x+.3, y+.51, z-.0001);
glEnd();
glColor3f(1, 1, 1);
glBegin(GL_POLYGON);
glVertex3f (x+0.23, y+.66, z-.0001);
glVertex3f (x+0.23, y+.73, z-.0001);
glVertex3f (x+.3, y+.73, z-.0001);
glVertex3f (x+.3, y+.66, z-.0001);
glEnd();
```

```
glColor3f(1, 1, 1);
glBegin(GL_POLYGON);
glVertex3f (x+0.23, y+.81, z-.0001);
glVertex3f (x+0.23, y+.88, z-.0001);
glVertex3f (x+.3, y+.88, z-.0001);
glVertex3f (x+.3, y+.81, z-.0001);
glEnd();
glColor3f(1, 1, 1);
glBegin(GL_POLYGON);
glVertex3f (x+0.23, y+.96, z-.0001);
glVertex3f (x+0.23, y+1.03, z-.0001);
glVertex3f (x+.3, y+1.03, z-.0001);
glVertex3f (x+.3, y+.96, z-.0001);
glEnd();
glColor3f(1, 1, 1);
glBegin(GL_POLYGON);
glVertex3f (x+0.23, y+1.11, z-.0001);
glVertex3f (x+0.23, y+1.18, z-.0001);
glVertex3f (x+.3, y+1.18, z-.0001);
glVertex3f (x+.3, y+1.11, z-.0001);
glEnd();
glColor3f(1, 1, 1);
glBegin(GL_POLYGON);
glVertex3f (x+0.23, y+1.26, z-.0001);
glVertex3f (x+0.23, y+1.33, z-.0001);
glVertex3f (x+.3, y+1.33, z-.0001);
glVertex3f (x+.3, y+1.26, z-.0001);
glEnd();
glColor3f(1, 1, 1);
glBegin(GL_POLYGON);
glVertex3f (x+0.23, y+1.41, z-.0001);
glVertex3f (x+0.23, y+1.48, z-.0001);
glVertex3f (x+.3, y+1.48, z-.0001);
glVertex3f (x+.3, y+1.41, z-.0001);
glEnd();
glColor3f(1, 1, 1);
glBegin(GL_POLYGON);
glVertex3f (x+0.23, y+1.56, z-.0001);
glVertex3f (x+0.23, y+1.63, z-.0001);
glVertex3f (x+.3, y+1.63, z-.0001);
glVertex3f (x+.3, y+1.56, z-.0001);
glEnd();
```

```
//Windows Leftinner
glColor3f(1, 1, 1);
glBegin(GL_POLYGON);
glVertex3f (x+0.39, y+.36, z-.0001);
glVertex3f (x+0.39, y+.43, z-.0001);
glVertex3f (x+.46, y+.43, z-.0001);
glVertex3f (x+.46, y+.36, z-.0001);
glEnd();
glColor3f(1, 1, 1);
glBegin(GL_POLYGON);
glVertex3f (x+0.39, y+.51, z-.0001);
glVertex3f (x+0.39, y+.58, z-.0001);
glVertex3f (x+.46, y+.58, z-.0001);
glVertex3f (x+.46, y+.51, z-.0001);
glEnd();
glColor3f(1, 1, 1);
glBegin(GL_POLYGON);
glVertex3f (x+0.39, y+.66, z-.0001);
glVertex3f (x+0.39, y+.73, z-.0001);
glVertex3f (x+.46, y+.73, z-.0001);
glVertex3f (x+.46, y+.66, z-.0001);
glEnd();
glColor3f(1, 1, 1);
glBegin(GL_POLYGON);
glVertex3f (x+0.39, y+.81, z-.0001);
glVertex3f (x+0.39, y+.88, z-.0001);
glVertex3f (x+.46, y+.88, z-.0001);
glVertex3f (x+.46, y+.81, z-.0001);
glEnd();
glColor3f(1, 1, 1);
glBegin(GL_POLYGON);
glVertex3f (x+0.39, y+.96, z-.0001);
glVertex3f (x+0.39, y+1.03, z-.0001);
glVertex3f (x+.46, y+1.03, z-.0001);
glVertex3f (x+.46, y+.96, z-.0001);
glEnd();
glColor3f(1, 1, 1);
glBegin(GL_POLYGON);
glVertex3f (x+0.39, y+1.11, z-.0001);
glVertex3f (x+0.39, y+1.18, z-.0001);
glVertex3f (x+.46, y+1.18, z-.0001);
glVertex3f (x+.46, y+1.11, z-.0001);
glEnd();
glColor3f(1, 1, 1);
```

```
    glBegin(GL_POLYGON);
    glVertex3f (x+0.39, y+1.26, z-.0001);
    glVertex3f (x+0.39, y+1.33, z-.0001);
    glVertex3f (x+.46, y+1.33, z-.0001);
    glVertex3f (x+.46, y+1.26, z-.0001);
    glEnd();
    glColor3f(1, 1, 1);
    glBegin(GL_POLYGON);
    glVertex3f (x+0.39, y+1.41, z-.0001);
    glVertex3f (x+0.39, y+1.48, z-.0001);
    glVertex3f (x+.46, y+1.48, z-.0001);
    glVertex3f (x+.46, y+1.41, z-.0001);
    glEnd();
    glColor3f(1, 1, 1);
    glBegin(GL_POLYGON);
    glVertex3f (x+0.39, y+1.56, z-.0001);
    glVertex3f (x+0.39, y+1.63, z-.0001);
    glVertex3f (x+.46, y+1.63, z-.0001);
    glVertex3f (x+.46, y+1.56, z-.0001);
    glEnd();


    //Door
    glColor3f(.1, .1, .1);
    glBegin(GL_POLYGON);
    glVertex3f (x+0.25, y, z-.0001);
    glVertex3f (x+0.25, y+.28, z-.0001);
    glVertex3f (x+.42, y+.28, z-.0001);
    glVertex3f (x+.42, y, z-.0001);
    glEnd();


  //OUTLINE
    if(cellshadeflag == 1)
    {
       //Front
       glColor3f(0, 0, 0);
       glBegin(GL_LINE_LOOP);
       glVertex3f (x, y, z);
       glVertex3f (x+0.7, y, z);
       glVertex3f (x+0.7, y+1.7, z);
       glVertex3f (x, y+1.7, z);
       glEnd();
```

```
//Back
glColor3f(0, 0, 0);
glBegin(GL_LINE_LOOP);
glVertex3f (x, y, z+0.5);
glVertex3f (x+0.7, y, z+0.5);
glVertex3f (x+0.7, y+1.7, z+0.5);
glVertex3f (x, y+1.7, z+0.5);
glEnd();

//Left
glColor3f(0, 0, 0);
glBegin(GL_LINE_LOOP);
glVertex3f (x, y, z);
glVertex3f (x, y, z+0.5);
glVertex3f (x, y+1.7, z+0.5);
glVertex3f (x, y+1.7, z);
glEnd();

//Right
glColor3f(0, 0, 0);
glBegin(GL_LINE_LOOP);
glVertex3f (x+0.7, y, z);
glVertex3f (x+0.7, y, z+0.5);
glVertex3f (x+0.7, y+1.7, z+0.5);
glVertex3f (x+0.7, y+1.7, z);
glEnd();

//Windows Rightmost
glColor3f(0, 0, 0);
glBegin(GL_LINE_LOOP);
glVertex3f (x+0.07, y+.21, z-.0001);
glVertex3f (x+0.07, y+.28, z-.0001);
glVertex3f (x+.14, y+.28, z-.0001);
glVertex3f (x+.14, y+.21, z-.0001);
glEnd();
glColor3f(0, 0, 0);
glBegin(GL_LINE_LOOP);
glVertex3f (x+0.07, y+.36, z-.0001);
glVertex3f (x+0.07, y+.43, z-.0001);
glVertex3f (x+.14, y+.43, z-.0001);
glVertex3f (x+.14, y+.36, z-.0001);
glEnd();
glColor3f(0, 0, 0);
glBegin(GL_LINE_LOOP);
```

```
glVertex3f (x+0.07, y+.51, z-.0001);
glVertex3f (x+0.07, y+.58, z-.0001);
glVertex3f (x+.14, y+.58, z-.0001);
glVertex3f (x+.14, y+.51, z-.0001);
glEnd();
glColor3f(0, 0, 0);
glBegin(GL_LINE_LOOP);
glVertex3f (x+0.07, y+.66, z-.0001);
glVertex3f (x+0.07, y+.73, z-.0001);
glVertex3f (x+.14, y+.73, z-.0001);
glVertex3f (x+.14, y+.66, z-.0001);
glEnd();
glColor3f(0, 0, 0);
glBegin(GL_LINE_LOOP);
glVertex3f (x+0.07, y+.81, z-.0001);
glVertex3f (x+0.07, y+.88, z-.0001);
glVertex3f (x+.14, y+.88, z-.0001);
glVertex3f (x+.14, y+.81, z-.0001);
glEnd();
glColor3f(0, 0, 0);
glBegin(GL_LINE_LOOP);
glVertex3f (x+0.07, y+.96, z-.0001);
glVertex3f (x+0.07, y+1.03, z-.0001);
glVertex3f (x+.14, y+1.03, z-.0001);
glVertex3f (x+.14, y+.96, z-.0001);
glEnd();
glColor3f(0, 0, 0);
glBegin(GL_LINE_LOOP);
glVertex3f (x+0.07, y+1.11, z-.0001);
glVertex3f (x+0.07, y+1.18, z-.0001);
glVertex3f (x+.14, y+1.18, z-.0001);
glVertex3f (x+.14, y+1.11, z-.0001);
glEnd();
glColor3f(0, 0, 0);
glBegin(GL_LINE_LOOP);
glVertex3f (x+0.07, y+1.26, z-.0001);
glVertex3f (x+0.07, y+1.33, z-.0001);
glVertex3f (x+.14, y+1.33, z-.0001);
glVertex3f (x+.14, y+1.26, z-.0001);
glEnd();
glColor3f(0, 0, 0);
glBegin(GL_LINE_LOOP);
glVertex3f (x+0.07, y+1.41, z-.0001);
glVertex3f (x+0.07, y+1.48, z-.0001);
```

```
glVertex3f (x+.14, y+1.48, z-.0001);
glVertex3f (x+.14, y+1.41, z-.0001);
glEnd();
glColor3f(0, 0, 0);
glBegin(GL_LINE_LOOP);
glVertex3f (x+0.07, y+1.56, z-.0001);
glVertex3f (x+0.07, y+1.63, z-.0001);
glVertex3f (x+.14, y+1.63, z-.0001);
glVertex3f (x+.14, y+1.56, z-.0001);
glEnd();


//Windows Leftmost
glColor3f(0, 0, 0);
glBegin(GL_LINE_LOOP);
glVertex3f (x+0.55, y+.21, z-.0001);
glVertex3f (x+0.55, y+.28, z-.0001);
glVertex3f (x+.62, y+.28, z-.0001);
glVertex3f (x+.62, y+.21, z-.0001);
glEnd();
glColor3f(0, 0, 0);
glBegin(GL_LINE_LOOP);
glVertex3f (x+0.55, y+.36, z-.0001);
glVertex3f (x+0.55, y+.43, z-.0001);
glVertex3f (x+.62, y+.43, z-.0001);
glVertex3f (x+.62, y+.36, z-.0001);
glEnd();
glColor3f(0, 0, 0);
glBegin(GL_LINE_LOOP);
glVertex3f (x+0.55, y+.51, z-.0001);
glVertex3f (x+0.55, y+.58, z-.0001);
glVertex3f (x+.62, y+.58, z-.0001);
glVertex3f (x+.62, y+.51, z-.0001);
glEnd();
glColor3f(0, 0, 0);
glBegin(GL_LINE_LOOP);
glVertex3f (x+0.55, y+.66, z-.0001);
glVertex3f (x+0.55, y+.73, z-.0001);
glVertex3f (x+.62, y+.73, z-.0001);
glVertex3f (x+.62, y+.66, z-.0001);
glEnd();
glColor3f(0, 0, 0);
glBegin(GL_LINE_LOOP);
glVertex3f (x+0.55, y+.81, z-.0001);
```

```
glVertex3f (x+0.55, y+.88, z-.0001);
glVertex3f (x+.62, y+.88, z-.0001);
glVertex3f (x+.62, y+.81, z-.0001);
glEnd();
glColor3f(0, 0, 0);
glBegin(GL_LINE_LOOP);
glVertex3f (x+0.55, y+.96, z-.0001);
glVertex3f (x+0.55, y+1.03, z-.0001);
glVertex3f (x+.62, y+1.03, z-.0001);
glVertex3f (x+.62, y+.96, z-.0001);
glEnd();
glColor3f(0, 0, 0);
glBegin(GL_LINE_LOOP);
glVertex3f (x+0.55, y+1.11, z-.0001);
glVertex3f (x+0.55, y+1.18, z-.0001);
glVertex3f (x+.62, y+1.18, z-.0001);
glVertex3f (x+.62, y+1.11, z-.0001);
glEnd();
glColor3f(0, 0, 0);
glBegin(GL_LINE_LOOP);
glVertex3f (x+0.55, y+1.26, z-.0001);
glVertex3f (x+0.55, y+1.33, z-.0001);
glVertex3f (x+.62, y+1.33, z-.0001);
glVertex3f (x+.62, y+1.26, z-.0001);
glEnd();
glColor3f(0, 0, 0);
glBegin(GL_LINE_LOOP);
glVertex3f (x+0.55, y+1.41, z-.0001);
glVertex3f (x+0.55, y+1.48, z-.0001);
glVertex3f (x+.62, y+1.48, z-.0001);
glVertex3f (x+.62, y+1.41, z-.0001);
glEnd();
glColor3f(0, 0, 0);
glBegin(GL_LINE_LOOP);
glVertex3f (x+0.55, y+1.56, z-.0001);
glVertex3f (x+0.55, y+1.63, z-.0001);
glVertex3f (x+.62, y+1.63, z-.0001);
glVertex3f (x+.62, y+1.56, z-.0001);
glEnd();


//Windows Rightinner
glColor3f(0, 0, 0);
glBegin(GL_LINE_LOOP);
```

```
glVertex3f (x+0.23, y+.36, z-.0001);
glVertex3f (x+0.23, y+.43, z-.0001);
glVertex3f (x+.3, y+.43, z-.0001);
glVertex3f (x+.3, y+.36, z-.0001);
glEnd();
glColor3f(0, 0, 0);
glBegin(GL_LINE_LOOP);
glVertex3f (x+0.23, y+.51, z-.0001);
glVertex3f (x+0.23, y+.58, z-.0001);
glVertex3f (x+.3, y+.58, z-.0001);
glVertex3f (x+.3, y+.51, z-.0001);
glEnd();
glColor3f(0, 0, 0);
glBegin(GL_LINE_LOOP);
glVertex3f (x+0.23, y+.66, z-.0001);
glVertex3f (x+0.23, y+.73, z-.0001);
glVertex3f (x+.3, y+.73, z-.0001);
glVertex3f (x+.3, y+.66, z-.0001);
glEnd();
glColor3f(0, 0, 0);
glBegin(GL_LINE_LOOP);
glVertex3f (x+0.23, y+.81, z-.0001);
glVertex3f (x+0.23, y+.88, z-.0001);
glVertex3f (x+.3, y+.88, z-.0001);
glVertex3f (x+.3, y+.81, z-.0001);
glEnd();
glColor3f(0, 0, 0);
glBegin(GL_LINE_LOOP);
glVertex3f (x+0.23, y+.96, z-.0001);
glVertex3f (x+0.23, y+1.03, z-.0001);
glVertex3f (x+.3, y+1.03, z-.0001);
glVertex3f (x+.3, y+.96, z-.0001);
glEnd();
glColor3f(0, 0, 0);
glBegin(GL_LINE_LOOP);
glVertex3f (x+0.23, y+1.11, z-.0001);
glVertex3f (x+0.23, y+1.18, z-.0001);
glVertex3f (x+.3, y+1.18, z-.0001);
glVertex3f (x+.3, y+1.11, z-.0001);
glEnd();
glColor3f(0, 0, 0);
glBegin(GL_LINE_LOOP);
glVertex3f (x+0.23, y+1.26, z-.0001);
glVertex3f (x+0.23, y+1.33, z-.0001);
```

```
glVertex3f (x+.3, y+1.33, z-.0001);
glVertex3f (x+.3, y+1.26, z-.0001);
glEnd();
glColor3f(0, 0, 0);
glBegin(GL_LINE_LOOP);
glVertex3f (x+0.23, y+1.41, z-.0001);
glVertex3f (x+0.23, y+1.48, z-.0001);
glVertex3f (x+.3, y+1.48, z-.0001);
glVertex3f (x+.3, y+1.41, z-.0001);
glEnd();
glColor3f(0, 0, 0);
glBegin(GL_LINE_LOOP);
glVertex3f (x+0.23, y+1.56, z-.0001);
glVertex3f (x+0.23, y+1.63, z-.0001);
glVertex3f (x+.3, y+1.63, z-.0001);
glVertex3f (x+.3, y+1.56, z-.0001);
glEnd();


//Windows Leftinner
glColor3f(0, 0, 0);
glBegin(GL_LINE_LOOP);
glVertex3f (x+0.39, y+.36, z-.0001);
glVertex3f (x+0.39, y+.43, z-.0001);
glVertex3f (x+.46, y+.43, z-.0001);
glVertex3f (x+.46, y+.36, z-.0001);
glEnd();
glColor3f(0, 0, 0);
glBegin(GL_LINE_LOOP);
glVertex3f (x+0.39, y+.51, z-.0001);
glVertex3f (x+0.39, y+.58, z-.0001);
glVertex3f (x+.46, y+.58, z-.0001);
glVertex3f (x+.46, y+.51, z-.0001);
glEnd();
glColor3f(0, 0, 0);
glBegin(GL_LINE_LOOP);
glVertex3f (x+0.39, y+.66, z-.0001);
glVertex3f (x+0.39, y+.73, z-.0001);
glVertex3f (x+.46, y+.73, z-.0001);
glVertex3f (x+.46, y+.66, z-.0001);
glEnd();
glColor3f(0, 0, 0);
glBegin(GL_LINE_LOOP);
glVertex3f (x+0.39, y+.81, z-.0001);
```

```
glVertex3f (x+0.39, y+.88, z-.0001);
glVertex3f (x+.46, y+.88, z-.0001);
glVertex3f (x+.46, y+.81, z-.0001);
glEnd();
glColor3f(0, 0, 0);
glBegin(GL_LINE_LOOP);
glVertex3f (x+0.39, y+.96, z-.0001);
glVertex3f (x+0.39, y+1.03, z-.0001);
glVertex3f (x+.46, y+1.03, z-.0001);
glVertex3f (x+.46, y+.96, z-.0001);
glEnd();
glColor3f(0, 0, 0);
glBegin(GL_LINE_LOOP);
glVertex3f (x+0.39, y+1.11, z-.0001);
glVertex3f (x+0.39, y+1.18, z-.0001);
glVertex3f (x+.46, y+1.18, z-.0001);
glVertex3f (x+.46, y+1.11, z-.0001);
glEnd();
glColor3f(0, 0, 0);
glBegin(GL_LINE_LOOP);
glVertex3f (x+0.39, y+1.26, z-.0001);
glVertex3f (x+0.39, y+1.33, z-.0001);
glVertex3f (x+.46, y+1.33, z-.0001);
glVertex3f (x+.46, y+1.26, z-.0001);
glEnd();
glColor3f(0, 0, 0);
glBegin(GL_LINE_LOOP);
glVertex3f (x+0.39, y+1.41, z-.0001);
glVertex3f (x+0.39, y+1.48, z-.0001);
glVertex3f (x+.46, y+1.48, z-.0001);
glVertex3f (x+.46, y+1.41, z-.0001);
glEnd();
glColor3f(0, 0, 0);
glBegin(GL_LINE_LOOP);
glVertex3f (x+0.39, y+1.56, z-.0001);
glVertex3f (x+0.39, y+1.63, z-.0001);
glVertex3f (x+.46, y+1.63, z-.0001);
glVertex3f (x+.46, y+1.56, z-.0001);
glEnd();


//Door
glColor3f(0, 0, 0);
glBegin(GL_LINE_LOOP);
```

```
            glVertex3f (x+0.25, y, z-.0001);
            glVertex3f (x+0.25, y+.28, z-.0001);
            glVertex3f (x+.42, y+.28, z-.0001);
            glVertex3f (x+.42, y, z-.0001);
            glEnd();
        }

}

void drawRoad(float x1, float z1, float x2, float z2, float width)
{
    //ROAD
        glColor3f(.2, .2, .2);
        glBegin(GL_POLYGON);
        glVertex3f (x1, 0, z1);
        glVertex3f (x2, 0, z2);
        glVertex3f (x2+width, 0, z2);
        glVertex3f (x1+width, 0, z1);
        glEnd();

    //OUTLINE
        if(cellshadeflag == 1)
        {
            glLineWidth(0.5);
            glColor3f(0, 0, 0);
            glBegin(GL_LINE_LOOP);
            glVertex3f (x1, 0, z1);
            glVertex3f (x2, 0, z2);
            glVertex3f (x2+width, 0, z2);
            glVertex3f (x1+width, 0, z1);
            glEnd();
        }
}

void drawPool(float x, float z)
{
    //POOL
        glColor3f(.12, .56, 1);
        glBegin(GL_POLYGON);
        glVertex3f (x, 0, z);
        glVertex3f (x+.4, 0, z+.2);
        glVertex3f (x+.8, 0, z+.5);
        glVertex3f (x+1, 0, z+.7);
        glVertex3f (x+1.1, 0, z+1.6);
        glVertex3f (x+1.2, 0, z+1.84);
        glVertex3f (x+1.1, 0, z+1.82);
        glVertex3f (x+.5, 0, z+1.62);
        glVertex3f (x+.4, 0, z+1.3);
        glVertex3f (x, 0, z+1);
        glVertex3f (x-.2, 0, z+1);
        glVertex3f (x-.5, 0, z+1);
        glEnd();

    //OUTLINE
        if(cellshadeflag == 1)
        {
            glLineWidth(3);
            glColor3f(0, 0, 0);
            glBegin(GL_LINE_LOOP);
            glVertex3f (x, 0, z);
            glVertex3f (x+.4, 0, z+.2);
            glVertex3f (x+.8, 0, z+.5);
            glVertex3f (x+1, 0, z+.7);
            glVertex3f (x+1.1, 0, z+1.6);
            glVertex3f (x+1.2, 0, z+1.84);
            glVertex3f (x+1.1, 0, z+1.82);
            glVertex3f (x+.5, 0, z+1.62);
            glVertex3f (x+.4, 0, z+1.3);
            glVertex3f (x, 0, z+1);
            glVertex3f (x-.2, 0, z+1);
            glVertex3f (x-.5, 0, z+1);
            glEnd();
        }
}

void drawBirds()
{
    //BIRDS
        glColor3f(1,1,1);
        glBegin(GL_POLYGON);
        glVertex3f (birdxz, 8, birdxz);
        glVertex3f (birdxz + .2 , 8 , birdxz + .5);
        glVertex3f (birdxz, 8 , birdxz + .2);
        glVertex3f (birdxz - .2, 8 , birdxz +.5);
        glEnd();

        glColor3f(1,1,1);
        glBegin(GL_POLYGON);
        glVertex3f (birdxz+1, 8, birdxz+1);
        glVertex3f (birdxz+1 + .2 , 8 , birdxz+1 + .5);
        glVertex3f (birdxz+1, 8 , birdxz+1 + .2);
        glVertex3f (birdxz+1 - .2, 8 , birdxz+1 +.5);
        glEnd();

        glColor3f(1,1,1);
        glBegin(GL_POLYGON);
        glVertex3f (birdxz+1, 8, birdxz-1);
        glVertex3f (birdxz+1 + .2 , 8 , birdxz-1 + .5);
        glVertex3f (birdxz+1, 8 , birdxz-1 + .2);
        glVertex3f (birdxz+1 - .2, 8 , birdxz-1 +.5);
        glEnd();

    //OUTLINE
        if(cellshadeflag == 1)
        {
            glLineWidth(0.05);
            glColor3f(.5,.5,.5);
            glBegin(GL_LINE_LOOP);
            glVertex3f (birdxz, 8, birdxz);
            glVertex3f (birdxz + .2 , 8 , birdxz + .5);
            glVertex3f (birdxz, 8 , birdxz + .2);
            glVertex3f (birdxz - .2, 8 , birdxz +.5);
            glEnd();

            glColor3f(.5,.5,.5);
            glBegin(GL_LINE_LOOP);
            glVertex3f (birdxz+1, 8, birdxz+1);
            glVertex3f (birdxz+1 + .2 , 8 , birdxz+1 + .5);
            glVertex3f (birdxz+1, 8 , birdxz+1 + .2);
            glVertex3f (birdxz+1 - .2, 8 , birdxz+1 +.5);
            glEnd();

            glColor3f(.3,.3,.3);
            glBegin(GL_LINE_LOOP);
            glVertex3f (birdxz+1, 8, birdxz-1);
            glVertex3f (birdxz+1 + .2 , 8 , birdxz-1 + .5);
            glVertex3f (birdxz+1, 8 , birdxz-1 + .2);
            glVertex3f (birdxz+1 - .2, 8 , birdxz-1 +.5);
            glEnd();
        }
}
```

```
void drawCloud(int x, int y, int z)
{
   //CLOUD
      glColor3f(.8, .8, 8);
      glBegin(GL_POLYGON);
      glVertex3f (x, y, z);
      glVertex3f (x+.4, y, z+.2);
      glVertex3f (x+.8, y, z+.5);
      glVertex3f (x+1, y, z+.7);
      glVertex3f (x+1.1, y, z+1.6);
      glVertex3f (x+1.2, y, z+1.84);
      glVertex3f (x+1.1, y, z+1.82);
      glVertex3f (x+.5, y, z+1.62);
      glVertex3f (x+.4, y, z+1.3);
      glVertex3f (x, y, z+1);
      glVertex3f (x-.2, y, z+1);
      glVertex3f (x-.5, y, z+1);
      glVertex3f (x+.4, y+.5, z+.2);
      glVertex3f (x+.8, y+.25, z+.57);
      glVertex3f (x+1, y+.5, z+.7);
      glVertex3f (x+.9, y+.25, z+.6);
      glVertex3f (x+.5, y+.25, z+1.62);
      glVertex3f (x+.4, y+.25, z+.5);
      glVertex3f (x, y+.5, z+1);
      glVertex3f (x-.5, y+.5, z+1);
      glVertex3f (x, y, z);
      glVertex3f (x+.4, y, z+.2);
      glVertex3f (x+.8, y-.25, z+.5);
      glVertex3f (x+1, y, z+.7);
      glVertex3f (x+.5, y-.5, z+1.6);
      glVertex3f (x+1.2, y, z+.84);
      glVertex3f (x+.5, y, z+2);
      glVertex3f (x+.5, y-.5, z+1.62);
      glVertex3f (x+.76, y, z+1.3);
      glVertex3f (x, y, z+1);
      glVertex3f (x-.2, y, z+1);
      glVertex3f (x-.2, y, z+.5);
      glVertex3f (x+1, y, z);
      glVertex3f (x+1, y, z+.5);
      glVertex3f (x+1.2, y, z+.6);
      glEnd();

   //OUTLINE
      if(cellshadeflag == 1)
```

```
        {
            glLineWidth(0.5);
            glColor3f(0, 0, 0);
            glBegin(GL_LINE_LOOP);
            glVertex3f (x, y, z);
            glVertex3f (x+.4, y, z+.2);
            glVertex3f (x+.8, y, z+.5);
            glVertex3f (x+1, y, z+.7);
            glVertex3f (x+1.1, y, z+1.6);
            glVertex3f (x+1.2, y, z+1.84);
            glVertex3f (x+1.1, y, z+1.82);
            glVertex3f (x+.5, y, z+1.62);
            glVertex3f (x+.4, y, z+1.3);
            glVertex3f (x, y, z+1);
            glVertex3f (x-.2, y, z+1);
            glVertex3f (x-.5, y, z+1);
            glVertex3f (x+.4, y+.5, z+.2);
            glVertex3f (x+.8, y+.25, z+.57);
            glVertex3f (x+1, y+.5, z+.7);
            glVertex3f (x+.9, y+.25, z+.6);
            glVertex3f (x+.5, y+.25, z+1.62);
            glVertex3f (x+.4, y+.25, z+.5);
            glVertex3f (x, y+.5, z+1);
            glVertex3f (x-.5, y+.5, z+1);
            glVertex3f (x, y, z);
            glVertex3f (x+.4, y, z+.2);
            glVertex3f (x+.8, y-.25, z+.5);
            glVertex3f (x+1, y, z+.7);
            glVertex3f (x+.5, y-.5, z+1.6);
            glVertex3f (x+1.2, y, z+.84);
            glVertex3f (x+.5, y, z+2);
            glVertex3f (x+.5, y-.5, z+1.62);
            glVertex3f (x+.76, y, z+1.3);
            glVertex3f (x, y, z+1);
            glVertex3f (x-.2, y, z+1);
            glVertex3f (x-.2, y, z+.5);
            glEnd();
        }
}

void drawSun(int y)
{
    //SUN
    glColor3f(.8, .8, 0);
    glBegin(GL_POLYGON);
    glVertex3f (0-15, y+10, 100); //A
    glVertex3f (-7.5-15, y+6.25, 100); //B
    glVertex3f (-10-15, y, 100); //C
    glVertex3f (-7.5-15, y-6.25, 100); //D
    glVertex3f (0-15, y-10, 100); //E
    glVertex3f (7.5-15, y-6.25, 100); //F
    glVertex3f (10-15, y, 100); //G
    glVertex3f (7.5-15, y+6.25, 100); //H
    glEnd();

    //OUTLINE
    if(cellshadeflag == 1)
    {
        glLineWidth(4.0);
        glColor3f(0, 0, 0);
        glBegin(GL_LINE_LOOP);
        glVertex3f (0-15, y+10, 100); //A
        glVertex3f (-7.5-15, y+6.25, 100); //B
        glVertex3f (-10-15, y, 100); //C
        glVertex3f (-7.5-15, y-6.25, 100); //D
        glVertex3f (0-15, y-10, 100); //E
        glVertex3f (7.5-15, y-6.25, 100); //F
        glVertex3f (10-15, y, 100); //G
        glVertex3f (7.5-15, y+6.25, 100); //H
        glEnd();
    }
}

void drawTree(float flx, float fz)
{
    float fby = 0.0;
    //TREE
        //TreeTrunk
        glColor3f(0.5f, 0.35f, 0.05f);
        glBegin(GL_POLYGON);
        glVertex3f (flx-0.5, fby, fz);
        glVertex3f (flx-0.6, fby, fz);
        glVertex3f (flx-0.55, fby, fz+0.1);
        glVertex3f (flx-0.5, fby+1.0, fz);
        glEnd();

        //Tree Branches
        glColor3f(.0, .6, .0);
        glBegin(GL_POLYGON);
        glVertex3f (flx-0.5, fby+1.0, fz);
        glVertex3f (flx-0.5, fby+1.0, fz+0.3);
        glVertex3f (flx-0.5, fby+1.0, fz-0.3);
        glVertex3f (flx-0.9, fby+1.0, fz);
        glVertex3f (flx-0.1, fby+1.0, fz);
        glVertex3f (flx-0.5, fby+1.5, fz);
        glVertex3f (flx-0.5, fby+1.0, fz+0.3);
        glEnd();


    //OUTLINE
        if(cellshadeflag == 1)
        {
            glLineWidth(1.5);
            //TreeTrunk
            glColor3f(0, 0, 0);
            glBegin(GL_LINE_LOOP);
            glVertex3f (flx-0.5, fby, fz);
            glVertex3f (flx-0.6, fby, fz);
            glVertex3f (flx-0.55, fby, fz+0.1);
            glVertex3f (flx-0.5, fby+1.0, fz);
            glEnd();

            //Tree Branches
            glColor3f(0, 0, 0);
            glBegin(GL_LINE_LOOP);
            glVertex3f (flx-0.5, fby+1.0, fz);
            glVertex3f (flx-0.5, fby+1.0, fz+0.3);
            glVertex3f (flx-0.5, fby+1.0, fz-0.3);
            glVertex3f (flx-0.9, fby+1.0, fz);
            glVertex3f (flx-0.1, fby+1.0, fz);
            glVertex3f (flx-0.5, fby+1.5, fz);
            glVertex3f (flx-0.5, fby+1.0, fz+0.3);
            glEnd();
        }
}

void drawGrass(int x, int z)
{
    glColor3f(0,0,0);
    glBegin(GL_LINE_LOOP);
    glVertex3f(x,0,z);
    glVertex3f(x+0.02,0.02,z);
    glVertex3f(x+0.04,0.03,z);
    glVertex3f(x+0.02,0.04,z);
    glVertex3f(x,0.02,z);
    glVertex3f(x-0.02,0.04,z);
    glVertex3f(x-0.04,0.03,z);
    glVertex3f(x-0.02,0.02,z);
    glEnd();

    glColor3f(0,0,0);
    glBegin(GL_LINE_LOOP);
    glVertex3f(z,0,x);
    glVertex3f(z,0.02,x+0.02);
    glVertex3f(z,0.03,x+0.04);
    glVertex3f(z,0.04,x+0.02);
    glVertex3f(z,0.02,x);
    glVertex3f(z,0.04,x-0.02);
    glVertex3f(z,0.03,x-0.04);
    glVertex3f(z,0.02,x-0.02);
    glEnd();
}

void drawGround()
{
    glColor3f(.1, .3, .0);
    glBegin(GL_POLYGON);
    glVertex3f (999, -0.001, -999);
    glVertex3f (999, -0.001, 999);
    glVertex3f (-999, -0.001, 999);
    glVertex3f (-999, -0.001, -999);
    glEnd();

    if(cellshadeflag == 1)
    {
        for(int i = 0 ; i < 100 ; i++)
        {
            glLineWidth(0.1);
            drawGrass(xgrassx[i], xgrassz[i]);
        }
    }
}

void drawFish(int x, int y, int z)
{
    //BIG FISH
```

```
        glColor3f(1, .64, .0);
        glBegin(GL_POLYGON);
        glVertex3f (x, y, z);
        glVertex3f (x-0.05, y+0.05, z);
        glVertex3f (x-0.1, y-0.02, z);
        glVertex3f (x-0.1, y+0.02, z);
        glVertex3f (x-0.05, y-0.05, z);
        glEnd();

    //SMALL FISH
        glColor3f(1, .64, .0);
        glBegin(GL_POLYGON);
        glVertex3f (x-.3, y, z-.3);
        glVertex3f (z-0.3, y+(0.05/2), x-(0.05/2)-0.3);
        glVertex3f (z-0.3, y-(0.02/2), x-(0.1/2)-0.3);
        glVertex3f (z-0.3, y+(0.02/2), x-(0.1/2)-0.3);
        glVertex3f (z-0.3, y-(0.05/2), x-(0.05/2)-0.3);
        glEnd();

    //OUTLINE
        if(cellshadeflag == 1)
        {
            glColor3f(0, 0, 0);
            glBegin(GL_LINE_LOOP);
            glVertex3f (x, y, z);
            glVertex3f (x-0.05, y+0.05, z);
            glVertex3f (x-0.1, y-0.02, z);
            glVertex3f (x-0.1, y+0.02, z);
            glVertex3f (x-0.05, y-0.05, z);
            glEnd();

            glColor3f(0, 0, 0);
            glBegin(GL_LINE_LOOP);
            glVertex3f (x-.3, y, z-.3);
            glVertex3f (z-0.3, y+(0.05/2), x-(0.05/2)-0.3);
            glVertex3f (z-0.3, y-(0.02/2), x-(0.1/2)-0.3);
            glVertex3f (z-0.3, y+(0.02/2), x-(0.1/2)-0.3);
            glVertex3f (z-0.3, y-(0.05/2), x-(0.05/2)-0.3);
            glEnd();
        }
}

void drawScene()
{
```

```
if(fullscreenflag == 1)
{
   glutFullScreen();
}
else
{
   if(fullscreenflag == -1)
   {
      glutPositionWindow(100,100);
      glutReshapeWindow(960, 600);
      fullscreenflag = 0;
   }
}
glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
glRotatef(-_cameraAngle, 0.0f, _zangle, 0.0f);
glTranslatef(-1.0f, -1.5f, -2.0f);
glPopMatrix();
glPushMatrix();
glTranslatef(_oangle, _zangle, _otherangle);
glRotatef(_angle, 0.0f, 1.0f, 0.0f);
glScalef(sfactor, sfactor, sfactor);
glColor3f(1.0, 0.25, 1.0);
glColor3f (.5, 0.5, .25);
if(skyboxflag == 1)
{
   glClearColor(0,.5,1,0);
}
else
{
   glClearColor(0,0,0,0);
}
glClear (GL_COLOR_BUFFER_BIT);


if(roadflag == 1)
{
   drawRoad(0,0, 0,8, .5);
   drawRoad(0,0, 5,2, .8);
   drawRoad(5,2, 8,4, .5);
   drawRoad(8,4, 0,9, .8);
}
if(buildflag == 1)
```

```
    {
        drawHouse(1.2, 1);
        drawHouse(1.2, 2.5);
        drawHouse(1.2, 4);
        drawHouse(1.2, 5.5);
        drawHouse(3, 4.8);
        drawHouse(3, 2);
        drawHouse(5, 3);
        drawTree(.3, .5);
        drawTree(.3, 2);
        drawTree(.3, 3.5);
        drawTree(.3, 5);
        drawTree(.3, 6.5);
        drawOffice(0, 8);
    }
    if(poolflag == 1)
    {
        drawPool(3.5, 3);
    }
    if(creatureflag == 1)
    {
        drawFish(4,0,4);
        drawBirds();
    }
    if(skypropflag == 1)
    {
        drawCloud(3.5, 4, 3);
        drawCloud(-.5,3.89,4);
        drawCloud(5.25, 3, 9.5);
        drawSun(30);
    }
    if(groundflag == 1)
    {
        drawGround();
    }

    glFlush ();
    glPopMatrix();
    glutSwapBuffers();
}

void update(int value)
{
    _angle = anglefactor;
    if (_angle > 360)
        _angle -= 360;

    birdxz += .2;
    if (birdxz >= 20)
    {
        birdxz = -40;
    }

    glutPostRedisplay();
    glutTimerFunc(25, update, 0);
}

int main(int argc, char ** argv)
{
    srand(time(0));

    for(int i = 0 ; i < 999 ; i++)
    {
        int mul = 1;
        if (rand()%2)
        {
            mul = 1;
        }
        else
        {
            mul = -1;
        }
        xgrassx[i]= ((rand()%50) + 4) * mul;
    }

    for(int i = 0 ; i < 999 ; i++)
    {
        int mul = 1;
        if (rand()%2)
        {
            mul = 1;
        }
        else
        {
            mul = -1;
        }
        xgrassz[i]= ((rand()%50) + 4) * mul;
    }
    glutInit( & argc, argv);
    glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH);
    glutInitWindowSize(960, 600);
    glutCreateWindow("SIMOPOLIS: CHECK CONSOLE FOR KEY BINDS");
    printf(" -------------------------------------------------------\n");
    printf("|SIMOPOLIS: A MINIPROJECT BY AMOGH SHET & ADARSH ACHARYA|\n");
    printf(" -------------------------------------------------------");
    printf("\n\n\n");
    printf("KEY BINDS:\n");
    printf("~~~~~~~~~~\n\n");
    printf("'W'       -   Move Forward                    '1'    -   Toggle Cell Shading\n");
    printf("'S'       -   Move Backward                   '2'    -   Toggle Sky Props\n");
    printf("'A'       -   Rotate Left                  '3'    -   Toggle Sky Box\n");
    printf("'D'       -   Rotate Right                  '4'    -   Toggle Ground\n\n");
    printf("'J'/'Q'   -   Strafe Left                  '5'    -   Toggle Road\n");
    printf("'L'/'E'   -   Strafe Right                  '6'    -   Toggle Creatures\n");
    printf("'I'       -   Move Up                     '7'    -   Toggle Pool\n");
    printf("'K'       -   Move Down                     '8'    -   Toggle Buildings\n\n");
    printf("Backspace -   Reset Camera Pos                  '='    -   Toggle All On\n");
    printf("Escape    -   Exit Program                    '-'    -   Toggle All Off\n");
    printf("Enter     -   Toggle Fullscreen");
    initRendering();
    glutDisplayFunc(drawScene);
    glutKeyboardFunc(handleKeypress);
    glutReshapeFunc(handleResize);
    glutTimerFunc(25, update, 0);
    sndPlaySound(TEXT("OMH.wav"), SND_ASYNC);
    glutMainLoop();
    return 0;
}
```

CONCLUSION

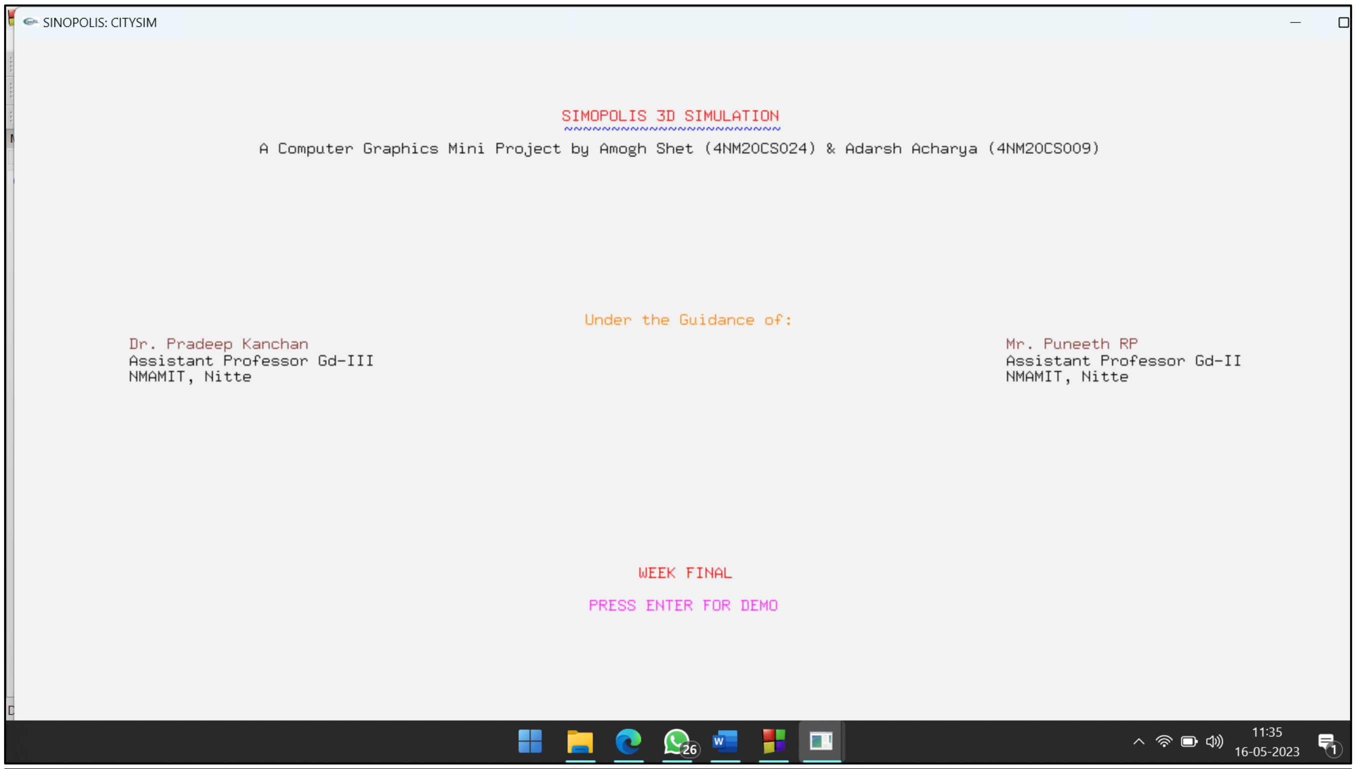
In conclusion, OpenGL is a versatile and widely-used graphics library that enables developers to create visually appealing and interactive 2D and 3D graphics. With its programmable pipeline, support for shaders, and cross-platform compatibility, OpenGL provides the necessary tools and capabilities to render realistic scenes, apply visual effects, and interact with user input.

The concept of an OpenGL neighborhood leverages the capabilities of this graphics library to offer a virtual tour of a neighbourhood fully realized in 3D. By combining the power of OpenGL with some clever workounds, the program creates a captivating and engaging experience, showcasing the potential of computer graphics to transport and inspire users. It lays the foundation for a project of much greater scope, and allows for users to see some of the more fundamental steps in 3D modeling.

REFERENCES

1. GeeksforGeeks: https://www.geeksforgeeks.org
2. Stack Overflow: https://stackoverflow.com
3. OpenGL: https://www.opengl.org
4. Adding Sound: https://www.youtube.com/watch?v=pCGyiBG8514

APPENDIX / RESULTS



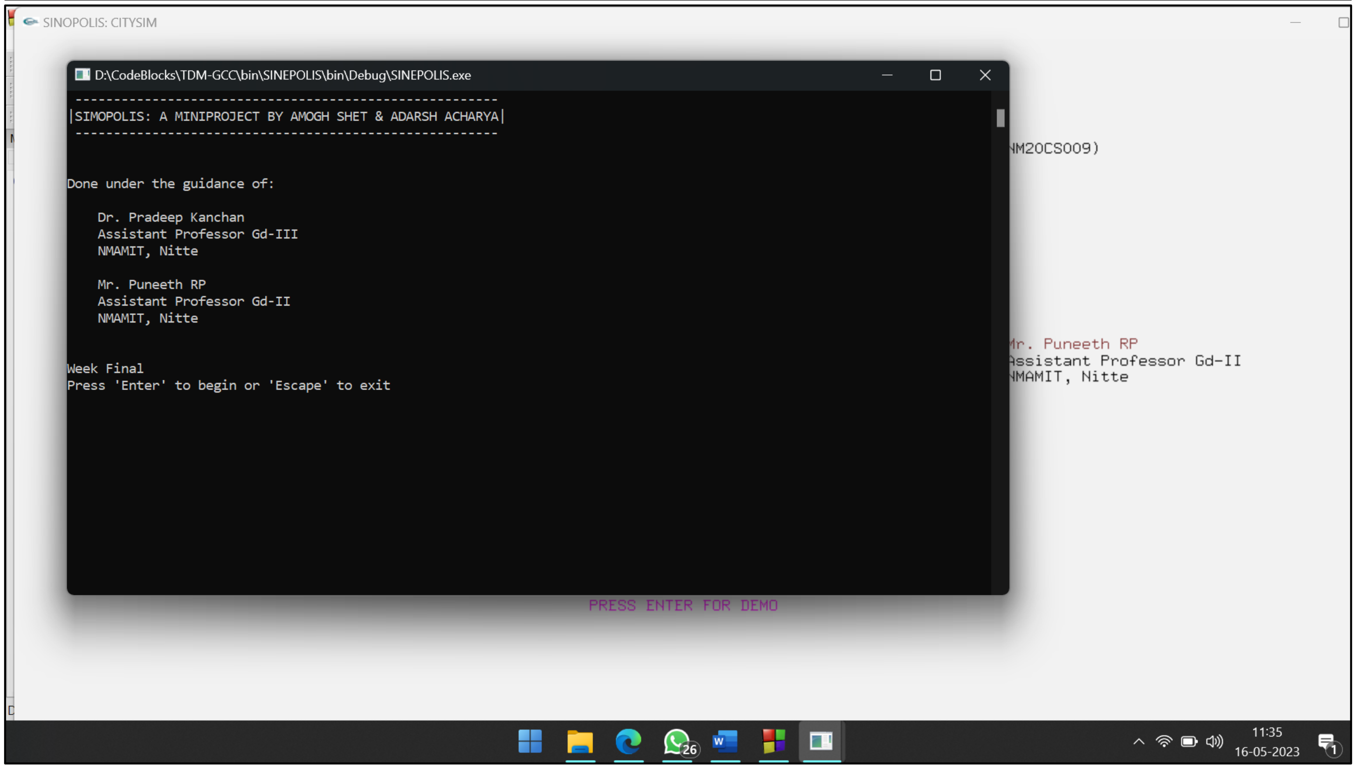
```
SINOPOLIS: CITYSIM

SIMOPOLIS 3D SIMULATION
A Computer Graphics Mini Project by Amogh Shet (4NM20CS024) & Adarsh Acharya (4NM20CS009)

Under the Guidance of:

Dr. Pradeep Kanchan                                  Mr. Puneeth RP
Assistant Professor Gd-III                           Assistant Professor Gd-II
NMAMIT, Nitte                                        NMAMIT, Nitte

WEEK FINAL
PRESS ENTER FOR DEMO
```



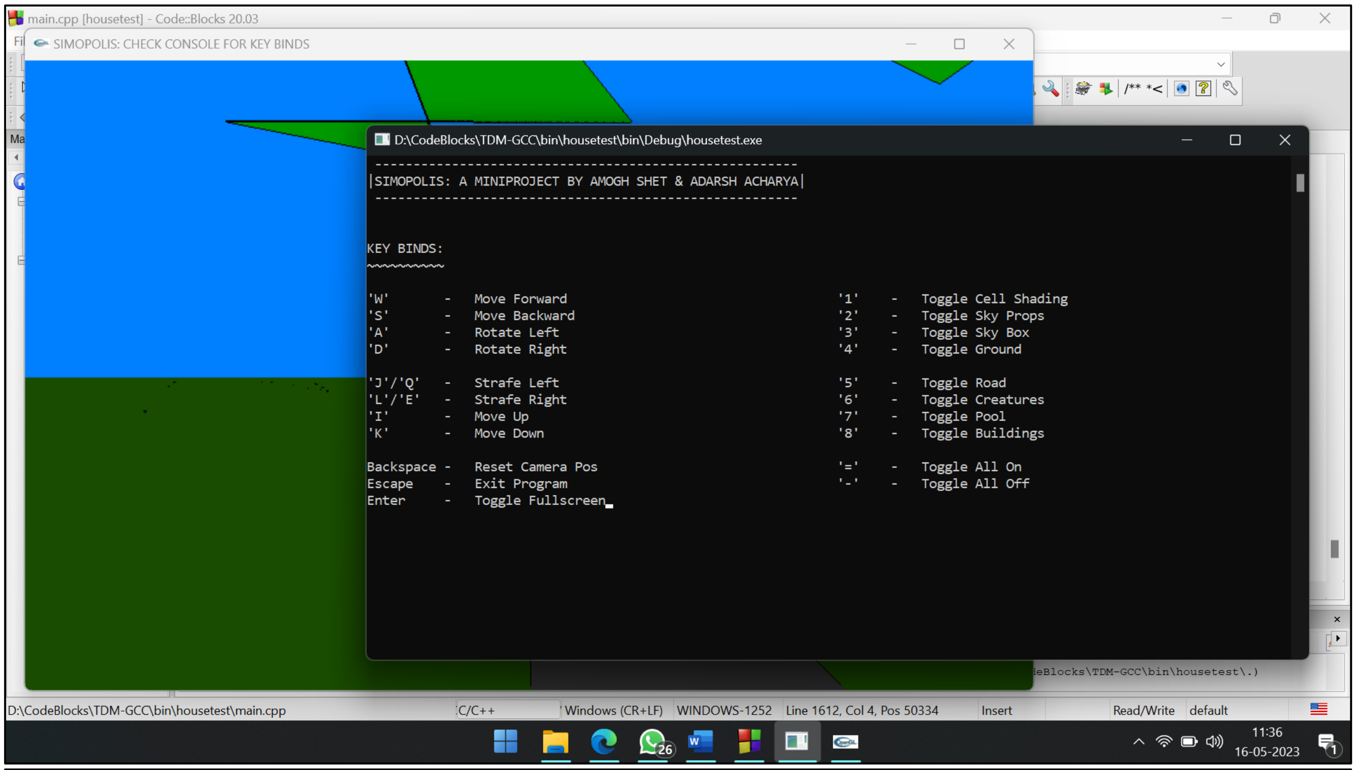
```
D:\CodeBlocks\TDM-GCC\bin\SINEPOLIS\bin\Debug\SINEPOLIS.exe
 -------------------------------------------------------
|SIMOPOLIS: A MINIPROJECT BY AMOGH SHET & ADARSH ACHARYA|
 -------------------------------------------------------

Done under the guidance of:

    Dr. Pradeep Kanchan
    Assistant Professor Gd-III
    NMAMIT, Nitte

    Mr. Puneeth RP
    Assistant Professor Gd-II
    NMAMIT, Nitte

Week Final
Press 'Enter' to begin or 'Escape' to exit
```


main.cpp [housetest] - Code::Blocks 20.03
SIMOPOLIS: CHECK CONSOLE FOR KEY BINDS
D:\CodeBlocks\TDM-GCC\bin\housetest\bin\Debug\housetest.exe
D:\CodeBlocks\TDM-GCC\bin\housetest\main.cpp
C/C++
Windows (CR+LF)
WINDOWS-1252
Line 1612, Col 4, Pos 50334
Insert
Read/Write
default
11:36
16-05-2023

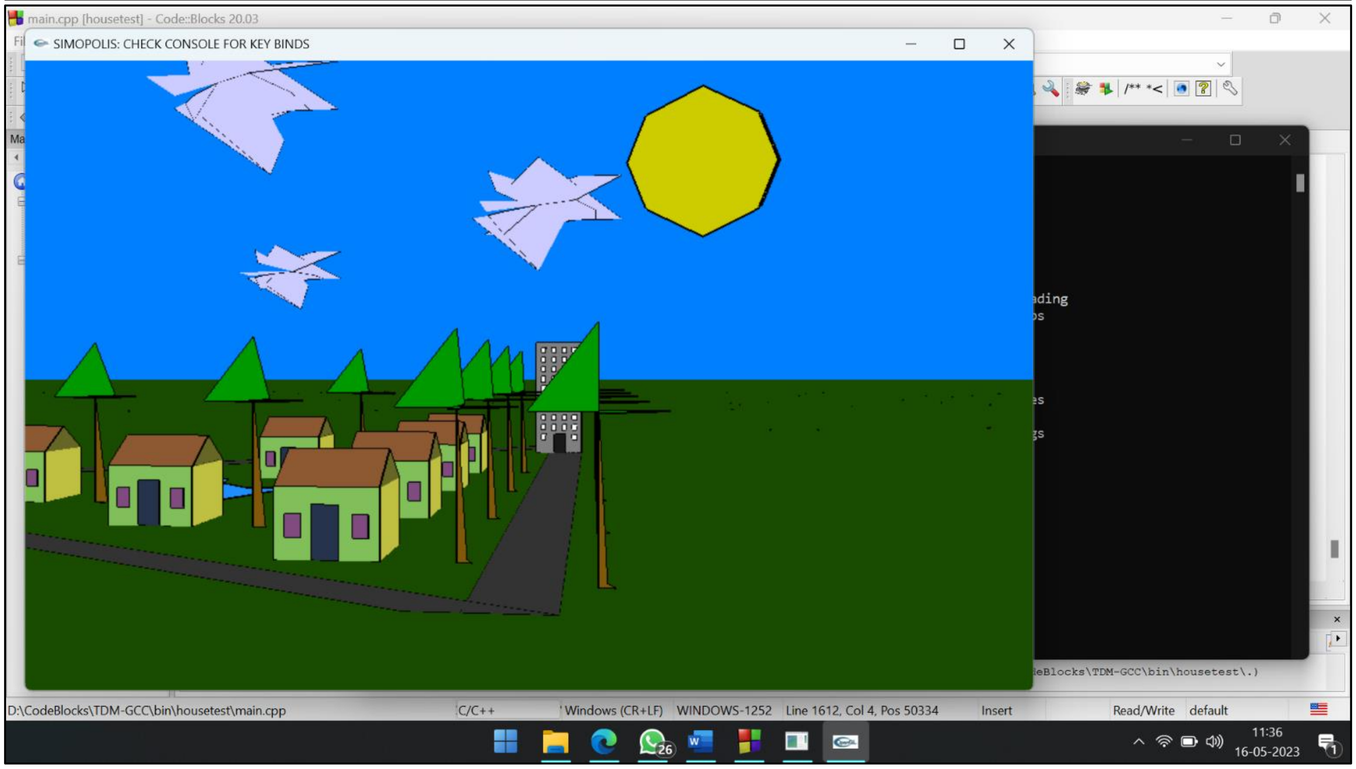
```
 ------------------------------------------------------
|SIMOPOLIS: A MINIPROJECT BY AMOGH SHET & ADARSH ACHARYA|
 ------------------------------------------------------


KEY BINDS:
~~~~~~~~~~

'W'       -   Move Forward                                 '1'    -   Toggle Cell Shading
'S'       -   Move Backward                                '2'    -   Toggle Sky Props
'A'       -   Rotate Left                                  '3'    -   Toggle Sky Box
'D'       -   Rotate Right                                 '4'    -   Toggle Ground

'J'/'Q'   -   Strafe Left                                  '5'    -   Toggle Road
'L'/'E'   -   Strafe Right                                 '6'    -   Toggle Creatures
'I'       -   Move Up                                      '7'    -   Toggle Pool
'K'       -   Move Down                                    '8'    -   Toggle Buildings

Backspace -   Reset Camera Pos                             '='    -   Toggle All On
Escape    -   Exit Program                                 '-'    -   Toggle All Off
Enter     -   Toggle Fullscreen_
```


main.cpp [housetest] - Code::Blocks 20.03
SIMOPOLIS: CHECK CONSOLE FOR KEY BINDS
D:\CodeBlocks\TDM-GCC\bin\housetest\main.cpp
C/C++
Windows (CR+LF)
WINDOWS-1252
Line 1612, Col 4, Pos 50334
Insert
Read/Write
default
11:36
16-05-2023