**Computer Graphics (UCS505)**

**Project on**

**SMART VILLAGE USING OPENGL**

**Submitted by:**

| Guntas Dhillon | 102016067 |
| --- | --- |
| Manan Singh Mehra | 102016066 |

**Group No. 11 (3CS11)**

**B.E. Third Year – CSE**

**Submitted to:**

**Ms. Jasmine Kaur**



**Computer Science and Engineering Department Thapar Institute of Engineering and Technology Patiala – 147001**





**Table of Contents**

| **Sr. No.** | **Description** | **Page No.** |
| --- | --- | --- |
| 1. | Introduction to Project | 2 |
| 2. | Computer Graphics concepts used | 3 |
| 3. | User Defined Functions | 4 |
| 4. | Code | 6 |
| 5. | Output / Screen shots | 21 |



**INTRODUCTION**

Village Scenery is a 2D graphics mini project developed using OpenGL, a powerful graphics library that enables developers to create visually stunning 2D and 3D graphics. This project is designed to showcase the beauty of a rural village setting through a series of carefully crafted graphical elements.

The project includes a variety of graphical components such as windmills, sun, mountains, trees, houses, and clouds, each with its unique characteristics and movements. The village's scenery is designed to be captivating, with a dynamic representation of nature in the village.

The project has been developed using modern programming techniques and optimized for performance, ensuring smooth and seamless rendering of graphics on different devices. The village scenery project is not only entertaining but also serves as an excellent educational tool for learning OpenGL and computer graphics concepts.

Potential applications of this OpenGL project include tourism and in urban planning and design. This could include evaluating the impact of new buildings, streets, or public spaces on traffic flow, pedestrian safety, and overall urban design.



**COMPUTER GRAPHICS CONCEPTS USED**

This is a piece of C code that uses OpenGL library to create a graphical scene of a sky with moving clouds and a sun. The scene is 2D and is drawn on a window with size 1000 x 500 pixels. The code starts by including the necessary libraries and defining some constants and global variables. Then, it defines some functions to draw the different objects in the scene.

1. **OpenGL:** OpenGL is a graphics rendering API (Application Programming Interface) used for creating 2D and 3D computer graphics. It provides a set of functions and tools for rendering high-quality graphics. The code uses OpenGL for rendering the graphics on the screen.
2. **2D graphics:** The code uses 2D graphics to display the game. This is done using the orthographic projection and the various 2D drawing functions provided by OpenGL.
3. **Orthographic Projection:** Orthographic projection is a type of 3D projection that preserves the relative size of objects in the scene, regardless of their distance from the viewer. In the code, the orthographic projection is set up using the gluOrtho2D() function, which sets up a 2D orthographic projection.
4. **Color:** The code uses RGB color values to specify the colors used in the graphics. RGB color values are defined using three parameters: red, green, and blue.
5. **Coordinates:** The code uses a Cartesian coordinate system to specify the location of the objects in the scene. The coordinate system has x, y, and z axes, which correspond to the horizontal, vertical, and depth dimensions, respectively.
6. **Projection:** The code uses the orthographic projection to display the game screen. This is done using the glOrtho() function.
7. **Rendering:** The code uses OpenGL to render the objects on the screen. This is done using the various OpenGL functions such as glClear(), glColor3f(), glPushMatrix(), and glPopMatrix().
8. **Mouse interaction:** The code uses the mouse to resume/pause the movement of scenery with the help of left/right mouse click respectively.

**USER-DEFINED FUNCTIONS**

Here is a detailed explanation of the functions used in the code:

* void init(void): The init() function is called once at the beginning of the program and is used to initialize various parameters of the OpenGL graphics system. In this code, it sets the background color of the window using glClearColor() and specifies the projection mode using gluOrtho2D().
* void circle(GLdouble rad): The circle() function is used to draw a circle using OpenGL. It takes a radius as a parameter and uses the glBegin() and glEnd() functions to define the circle as a set of connected points using the glVertex2f() function.
* void Sun\_Model(): The Sun\_Model() function is used to draw the sun. It uses the circle() function to draw a circle with a radius of 30 units and then uses glPushMatrix() and glTranslatef() functions to move the sun to the center of the window.
* void Moving\_Sun\_Model() It uses the glRotatef() functions to create a transformation matrix that rotates the sun around the center of the window. The rotation angle is determined by the `sun\_spin` variable, which is incremented in the main loop of the program.
* void cloud\_model\_one(), void cloud\_model\_two(), cloud\_model\_three(): They draw different cloud shapes using the circle() function to draw several circles of different radii at different positions to form the shape of a cloud.
* void hill\_big(): It defines the graphics for drawing a big hill with snow on top in the background of the scene. The function first defines the hill shape as a polygon using the "glBegin" function. The "glColor3f" function is used to set the color of the hill to a grayish-green color. Next, the function defines the snow on top of the hill as another polygon using the "glBegin" function. The "glColor3f" function is used to set the color of the snow to a gray-white color.
* void hill\_small(): It defines the graphics for drawing the smaller hill with snow on top in the background of the scene. The function first defines the hill shape as a polygon using the "glBegin" function. The "glColor3f" function is used to set the color of the hill to a grayish-green color. Next, the function defines the snow on top of the hill as another polygon using the "glBegin" function. The "glColor3f" function is used to set the color of the snow to a gray-white color.
* void Hill\_One\_Model(), void Hill\_Two\_Model(): It is used to draw the main body and the two wheels of the hills. First uses “glBegin” to start drawing a polygon and then specifies the color with `glColor3f`. The second one uses `glPushMatrix()` and `glPopMatrix()` to push and pop the current matrix stack so that each wheel can be drawn independently. It also uses `glTranslatef()` to translate the wheels to their desired positions.
* void house(): It defines the elements of the house, including the roof, fence, door, windows and shadows.
* void field(): Defines the green field on which the house and trees are situated.
* void Tree\_Model\_One(), void Tree\_Model\_Two(), void Tree\_Model\_Three(): These are three functions that draw three different models of trees using OpenGL primitives such as circles and polygons.
* void Windmill\_Stand\_Model(): This function draws the stand of the windmill.
* void Windmill\_Blade(): This function draws the three blades of the windmill, and rotates them around a pivot point.
* void Windmill(): This function is used to call the above two functions and draws the complete windmill.
* void display(void): The display function is responsible for rendering all the graphics objects in the scene. It first clears the color buffer and sets the color to blue. Then, it calls several functions to draw various objects such as the sun, windmills, hills, clouds, trees, houses, and fields. Finally, it flushes the buffer to display the rendered objects on the screen.
* void mouse(int key, int state, int x, int y): The mouse function is called whenever the user interacts with the mouse. In this case, it is used to control the animation. When the left mouse button is clicked, the move\_right function is called repeatedly using the glutIdleFunc function, which updates the position of the objects and redraws the scene. When the middle or right mouse button is clicked, the animation is stopped by setting the idle function to NULL.
* The move\_right function is responsible for updating the positions of the objects to create the illusion of motion. It does this by incrementing the x-coordinates of the objects and the angle of rotation of the sun. When an object reaches the right edge of the screen, it is wrapped around to the left edge to create an infinite loop. Finally, it calls glutPostRedisplay to update the display with the new positions of the objects.

**CODE**

*#include<windows.h>*

*#include<GL\glut.h>*

*#include <GL/glu.h>*

*#include<math.h>*

*#include <stdlib.h>*

*#include<stdio.h>*

*#define PI 3.1416*

*GLint i, j, k;*

*GLfloat sun\_spin = 0, sun\_x = 0, sun\_y = 0;*

*GLfloat ax = 0, bx = 0, cx = 0, dx = 0, str = 500.0, mn = 500.0;*

*GLfloat sr = 0.0, sg = 0.749, sb = 1.0;*

*GLfloat spin = 0.0;*

*void init(void)*

*{*

*glClearColor(.20, 0.80, 1.80, 0.0);*

*glMatrixMode(GL\_PROJECTION);*

*gluOrtho2D(0.0, 1000.0, 0.0, 500.0);*

*}*

*///\*\*\* Circle\_Model\*\*\*///*

*void circle(GLdouble rad)*

*{*

*GLint points = 50;*

*GLdouble delTheta = (2.0 \* PI) / (GLdouble)points;*

*GLdouble theta = 0.0;*

*glBegin(GL\_POLYGON);*

*{*

*for (i = 0; i <= 50; i++, theta += delTheta)*

*{*

*glVertex2f(rad \* cos(theta), rad \* sin(theta));*

*}*

*}*

*glEnd();*

*}*

*/// \*\*\* Sun\_Model \*\*///*

*void Sun\_Model() {*

*glPushMatrix();*

*glTranslatef(500, 0, 0);*

*circle(30);*

*glPopMatrix();*

*}*

*void Moving\_Sun\_Model() {*

*glPushMatrix();*

*glRotatef(-sun\_spin, 0, 0, -.009);*

*Sun\_Model();*

*glPopMatrix();*

*}*

*///\*\*\* Cloud\_Model\*\*\*///*

*void cloud\_model\_one() {*

*glColor3f(1.25, 0.924, 0.930);*

*///Top\_Left*

*glPushMatrix();*

*glTranslatef(320, 210, 0);*

*circle(15);*

*glPopMatrix();*

*///Top*

*glPushMatrix();*

*glTranslatef(340, 225, 0);*

*circle(16);*

*glPopMatrix();*

*///Right*

*glPushMatrix();*

*glTranslatef(360, 210, 0);*

*circle(16);*

*glPopMatrix();*

*///middle\_Fill*

*glPushMatrix();*

*glTranslatef(355, 210, 0);*

*circle(16);*

*glPopMatrix();*

*glPushMatrix();*

*glTranslatef(350, 210, 0);*

*circle(16);*

*glPopMatrix();*

*glPushMatrix();*

*glTranslatef(345, 204, 0);*

*circle(10);*

*glPopMatrix();*

*glPushMatrix();*

*glTranslatef(340, 204, 0);*

*circle(10);*

*glPopMatrix();*

*glPushMatrix();*

*glTranslatef(335, 204, 0);*

*circle(10);*

*glPopMatrix();*

*glPushMatrix();*

*glTranslatef(330, 204, 0);*

*circle(10);*

*glPopMatrix();*

*glPushMatrix();*

*glTranslatef(325, 204, 0);*

*circle(10);*

*glPopMatrix();*

*glPushMatrix();*

*glTranslatef(320, 204, 0);*

*circle(10);*

*glPopMatrix();*

*glPushMatrix();*

*glTranslatef(315, 204, 0);*

*circle(10);*

*glPopMatrix();*

*glPushMatrix();*

*glTranslatef(310, 204, 0);*

*circle(10);*

*glPopMatrix();*

*glPushMatrix();*

*glTranslatef(305, 204, 0);*

*circle(10);*

*glPopMatrix();*

*///\*\*\*\*Fill End\*\*\*\**

*}*

*void cloud\_model\_Two() {*

*glColor3f(1.25, 0.924, 0.930);*

*///Left\_Part*

*glPushMatrix();*

*glTranslatef(305, 205, 0);*

*circle(10);*

*glPopMatrix();*

*///Top*

*glPushMatrix();*

*glTranslatef(320, 210, 0);*

*circle(15);*

*glPopMatrix();*

*///Right\_Part*

*glPushMatrix();*

*glTranslatef(334, 207, 0);*

*circle(10);*

*glPopMatrix();*

*///Bottom\_Part*

*glPushMatrix();*

*glTranslatef(320, 207, 0);*

*circle(10);*

*glPopMatrix();*

*}*

*void cloud\_model\_Three() {*

*glColor3f(1.25, 0.924, 0.930);*

*///Left\_Part*

*glPushMatrix();*

*glTranslatef(300, 200, 0);*

*circle(15);*

*glPopMatrix();*

*///Top\_Left*

*glPushMatrix();*

*glTranslatef(320, 210, 0);*

*circle(15);*

*glPopMatrix();*

*///Top*

*glPushMatrix();*

*glTranslatef(340, 220, 0);*

*circle(16);*

*glPopMatrix();*

*///Top\_Right*

*glPushMatrix();*

*glTranslatef(360, 210, 0);*

*circle(15);*

*glPopMatrix();*

*///Right\_Part*

*glPushMatrix();*

*glTranslatef(380, 200, 0);*

*circle(15);*

*glPopMatrix();*

*///Bottom\_Right*

*glPushMatrix();*

*glTranslatef(360, 190, 0);*

*circle(20);*

*glPopMatrix();*

*///Bottom\_Left*

*glPushMatrix();*

*glTranslatef(320, 190, 0);*

*circle(20);*

*glPopMatrix();*

*///Bottom*

*glPushMatrix();*

*glTranslatef(340, 190, 0);*

*circle(20);*

*glPopMatrix();*

*///\*\*\*\*Fill End\*\*\*\**

*}*

*///\*\*\* Hill\_Model\*\*\*///*

*void hill\_big() {*

*///Hill*

*glBegin(GL\_POLYGON);*

*glColor3f(0.38, 0.41, 0.36);*

*glVertex2i(70, 70);*

*glVertex2i(200, 225);*

*glVertex2i(330, 70);*

*glEnd();*

*///Hill\_Snow*

*glBegin(GL\_POLYGON);*

*glColor3f(1.25, 0.924, 0.930);*

*glVertex2i(200, 225);*

*glVertex2i(230, 190);*

*glVertex2i(220, 180);*

*glVertex2i(200, 190);*

*glVertex2i(190, 180);*

*glVertex2i(170, 190);*

*glEnd();*

*}*

*void hill\_small() {*

*///Hill\_Small*

*glBegin(GL\_POLYGON);*

*glColor3f(0.11, 0.23, 0.36);*

*glVertex2i(250, 100);*

*glVertex2i(310, 175);*

*glVertex2i(370, 100);*

*glEnd();*

*///Hill\_Small\_Snow*

*glBegin(GL\_POLYGON);*

*glColor3f(1.25, 0.924, 0.930);*

*glVertex2i(290, 150);*

*glVertex2i(310, 175);*

*glVertex2i(330, 150);*

*glVertex2i(325, 140);*

*glVertex2i(310, 150);*

*glVertex2i(300, 140);*

*glEnd();*

*}*

*///\*\*\* Hill\_Model \*\*\*///*

*void Hill\_One\_Model() {*

*///Hill*

*glBegin(GL\_POLYGON);*

*glColor3f(0.1, 0.63, 0.10);*

*glVertex2i(125, 70);*

*glVertex2i(150, 80);*

*glVertex2i(160, 90);*

*glVertex2i(170, 90);*

*glVertex2i(180, 100);*

*glVertex2i(190, 105);*

*glVertex2i(200, 108);*

*glVertex2i(300, 110);*

*glVertex2i(300, 70);*

*glEnd();*

*}*

*void Hill\_Two\_Model() {*

*glColor3f(0.1, 0.693, 0.10);*

*/// Left\_Part*

*glPushMatrix();*

*glTranslatef(430, 90, 0);*

*circle(30);*

*glPopMatrix();*

*glPushMatrix();*

*glTranslatef(420, 87, 0);*

*circle(30);*

*glPopMatrix();*

*glPushMatrix();*

*glTranslatef(410, 80, 0);*

*circle(30);*

*glPopMatrix();*

*glPushMatrix();*

*glTranslatef(400, 80, 0);*

*circle(30);*

*glPopMatrix();*

*glPushMatrix();*

*glTranslatef(390, 70, 0);*

*circle(30);*

*glPopMatrix();*

*///Right\_Part*

*glPushMatrix();*

*glTranslatef(445, 80, 0);*

*circle(30);*

*glPopMatrix();*

*glPushMatrix();*

*glTranslatef(455, 75, 0);*

*circle(30);*

*glPopMatrix();*

*glPushMatrix();*

*glTranslatef(465, 70, 0);*

*circle(30);*

*glPopMatrix();*

*glPushMatrix();*

*glTranslatef(470, 65, 0);*

*circle(30);*

*glPopMatrix();*

*glPushMatrix();*

*glTranslatef(480, 60, 0);*

*circle(30);*

*glPopMatrix();*

*glPushMatrix();*

*glTranslatef(485, 55, 0);*

*circle(20);*

*glPopMatrix();*

*}*

*///\*\*\* House\_Model \*\*\*///*

*void house() {*

*///House\_Roof*

*glBegin(GL\_POLYGON);*

*glColor3f(.790, 0.10, 0.0);*

*glVertex2i(285, 104);*

*glVertex2i(295, 130);*

*glVertex2i(380, 115);*

*glVertex2i(380, 104);*

*glEnd();*

*///House\_Roof\_Shadow*

*glBegin(GL\_POLYGON);*

*glColor3f(.690, 0.10, 0.0);*

*glVertex2i(285, 104);*

*glVertex2i(292, 120);*

*glVertex2i(380, 104);*

*glVertex2i(380, 104);*

*glEnd();*

*///House*

*glBegin(GL\_POLYGON);*

*glColor3f(.426, 0.507, 0.60);*

*glVertex2i(290, 70);*

*glVertex2i(290, 104);*

*glVertex2i(375, 104);*

*glVertex2i(375, 70);*

*glEnd();*

*///House\_Shadow*

*glBegin(GL\_POLYGON);*

*glColor3f(.555, 0.94, 0.930);*

*glVertex2i(310, 70);*

*glVertex2i(350, 104);*

*glVertex2i(375, 104);*

*glVertex2i(375, 70);*

*glEnd();*

*///House\_Door*

*glBegin(GL\_POLYGON);*

*glColor3f(0.38, 0.41, 0.6);*

*glVertex2i(330, 70);*

*glVertex2i(330, 100);*

*glVertex2i(350, 100);*

*glVertex2i(350, 70);*

*glEnd();*

*///House\_Window1*

*glBegin(GL\_POLYGON);*

*glColor3f(0.38, 0.21, 0.26);*

*glVertex2i(295, 75);*

*glVertex2i(295, 90);*

*glVertex2i(310, 90);*

*glVertex2i(310, 75);*

*glEnd();*

*///House\_Window2*

*glBegin(GL\_POLYGON);*

*glColor3f(0.38, 0.21, 0.26);*

*glVertex2i(312, 75);*

*glVertex2i(312, 90);*

*glVertex2i(327, 90);*

*glVertex2i(327, 75);*

*glEnd();*

*///House\_Window3*

*glBegin(GL\_POLYGON);*

*glColor3f(0.38, 0.21, 0.26);*

*glVertex2i(355, 75);*

*glVertex2i(355, 90);*

*glVertex2i(370, 90);*

*glVertex2i(370, 75);*

*glEnd();*

*///House\_Small\_Roof*

*glBegin(GL\_POLYGON);*

*glColor3f(.790, 0.10, 0.0);*

*glVertex2i(250, 90);*

*glVertex2i(257, 104);*

*glVertex2i(290, 104);*

*glVertex2i(290, 90);*

*glEnd();*

*///House\_Small\_Fence*

*glBegin(GL\_POLYGON);*

*glColor3f(.555, .724, .830);*

*glVertex2i(255, 70);*

*glVertex2i(255, 90);*

*glVertex2i(290, 90);*

*glVertex2i(290, 70);*

*glEnd();*

*///House\_Small\_Door*

*glBegin(GL\_POLYGON);*

*glColor3f(0.11, 0.33, 0.36);*

*glVertex2i(260, 70);*

*glVertex2i(260, 80);*

*glVertex2i(285, 80);*

*glVertex2i(285, 70);*

*glEnd();*

*}*

*///\*\*\* Field\_Model \*\*\*///*

*void field() {*

*///Field*

*glBegin(GL\_POLYGON);*

*glColor3f(0.4, 0.83, 0.2);*

*glVertex2i(0, 50);*

*glVertex2i(0, 70);*

*glVertex2i(1000, 70);*

*glVertex2i(1000, 50);*

*glEnd();*

*///Field\_Shadow*

*glBegin(GL\_POLYGON);*

*glColor3f(0.3, 0.73, 0.20);*

*glVertex2i(0, 0);*

*glVertex2i(0, 50);*

*glVertex2i(1000, 50);*

*glVertex2i(1000, 0);*

*glEnd();*

*}*

*///\*\*\* Tree\_Model \*\*\*///*

*void Tree\_Model\_One() {*

*glPushMatrix();*

*glTranslatef(110, 110, 0);*

*circle(15);*

*glPopMatrix();*

*glPushMatrix();*

*glTranslatef(110, 100, 0);*

*circle(15);*

*glPopMatrix();*

*glBegin(GL\_POLYGON);*

*glColor3f(0.38, 0.11, 0.26);*

*glVertex2f(109, 70);*

*glVertex2f(109, 88);*

*glVertex2f(111, 88);*

*glVertex2f(111, 70);*

*glEnd();*

*}*

*void Tree\_Model\_Two() {*

*glPushMatrix();*

*glTranslatef(130, 130, 0);*

*circle(5);*

*glPopMatrix();*

*glPushMatrix();*

*glTranslatef(125, 126, 0);*

*circle(5);*

*glPopMatrix();*

*glPushMatrix();*

*glTranslatef(135, 126, 0);*

*circle(5);*

*glPopMatrix();*

*glPushMatrix();*

*glTranslatef(130, 125, 0);*

*circle(5);*

*glPopMatrix();*

*glBegin(GL\_POLYGON);*

*glColor3f(0.38, 0.21, 0.26);*

*glVertex2f(129, 110);*

*glVertex2f(129, 122);*

*glVertex2f(131, 122);*

*glVertex2f(131, 110);*

*glEnd();*

*}*

*void Tree\_Model\_Three() {*

*glBegin(GL\_POLYGON);*

*glVertex2f(125, 123);*

*glVertex2f(133, 145);*

*glVertex2f(141, 123);*

*glEnd();*

*glBegin(GL\_POLYGON);*

*glColor3f(0.38, 0.21, 0.26);*

*glVertex2f(132, 110);*

*glVertex2f(132, 122);*

*glVertex2f(134, 122);*

*glVertex2f(134, 110);*

*glEnd();*

*}*

*/// \*\*\* Windmill\_Stand\_Model \*\*\*///*

*void Windmill\_Stand\_Model() {*

*glColor3f(0.38, 0.41, 0.36);*

*glBegin(GL\_POLYGON);*

*glVertex2i(375, 100);*

*glVertex2i(380, 240);*

*glVertex2i(384, 240);*

*glVertex2i(390, 100);*

*glEnd();*

*}*

*///\*\*\* Windmill\_Blades\_Model \*\*\*///*

*void Windmill\_Blade() {*

*///Blade\_One*

*glPushMatrix();*

*glRotatef(spin, 0, 0, 90);*

*glBegin(GL\_POLYGON);*

*glVertex2i(-5, 0);*

*glVertex2i(-85, -36);*

*glVertex2i(-83, -37);*

*glVertex2i(-3, -8);*

*glEnd();*

*glPopMatrix();*

*///Blade\_Two*

*glPushMatrix();*

*glRotatef(spin, 0, 0, 90);*

*glBegin(GL\_POLYGON);*

*glVertex2i(0, 5);*

*glVertex2i(45, 70);*

*glVertex2i(50, 73);*

*glVertex2i(5, 0);*

*glEnd();*

*glPopMatrix();*

*///Blade\_Three*

*glPushMatrix();*

*glRotatef(spin, 0, 0, 90);*

*glBegin(GL\_POLYGON);*

*glVertex2i(68, -78);*

*glVertex2i(0, 0);*

*glVertex2i(5, 5);*

*glVertex2i(70, -77);*

*glEnd();*

*glPopMatrix();*

*}*

*///\*\*\* Windmill\_Final\_Model \*\*\*///*

*void Windmill() {*

*///Windmill\_Stand*

*glColor3f(0.38, 0.41, 0.36);*

*glPushMatrix();*

*Windmill\_Stand\_Model();*

*glPopMatrix();*

*///Windmill\_Motor*

*glColor3f(0.11, 0.23, 0.36);*

*glPushMatrix();*

*glTranslatef(380, 250, 0);*

*circle(10);*

*glPopMatrix();*

*///Windmill\_Rotary\_Blades*

*glColor3f(0.11, 0.23, 0.36);*

*glPushMatrix();*

*glTranslatef(380, 251, 0);*

*Windmill\_Blade();*

*glPopMatrix();*

*}*

*///\*\*\* Sun \*\*\*///*

*void Sun() {*

*glColor3f(1, 1, 0);*

*glPushMatrix();*

*Moving\_Sun\_Model();*

*glPopMatrix();*

*}*

*///\*\*\* Cloud\_One\_Model\_One \*\*\*///*

*void cloud\_one() {*

*glPushMatrix();*

*glTranslatef(cx, -40, 0);*

*cloud\_model\_one();*

*glPopMatrix();*

*}*

*///\*\*\* Cloud\_Two\_Model\_one \*\*\*///*

*void cloud\_two() {*

*glPushMatrix();*

*glTranslatef(bx + 100, 150, 0);*

*cloud\_model\_one();*

*glPopMatrix();*

*}*

*///\*\*\* Cloud\_Three\_Model\_Two \*\*\*///*

*void cloud\_three() {*

*glPushMatrix();*

*glTranslatef(ax - 80, 80, 0);*

*cloud\_model\_Two();*

*glPopMatrix();*

*}*

*///\*\*\* Cloud\_Four\_Model\_Two \*\*\*///*

*void cloud\_four() {*

*glPushMatrix();*

*glTranslatef(dx + 300, 125, 0);*

*cloud\_model\_Two();*

*glPopMatrix();*

*}*

*///\*\*\* Cloud\_Five\_Model\_Three \*\*\*///*

*void cloud\_five() {*

*glPushMatrix();*

*glTranslatef(ax + -300, 170, 0);*

*cloud\_model\_Three();*

*glPopMatrix();*

*}*

*///\*\*\* Cloud\_Six\_Model\_Three \*\*\*///*

*void cloud\_six() {*

*glPushMatrix();*

*glTranslatef(cx + -500, 20, 0);*

*cloud\_model\_Three();*

*glPopMatrix();*

*}*

*///\*\*\* House\_One \*\*\*///*

*void house\_one() {*

*glPushMatrix();*

*glTranslatef(0, 0, 0);*

*house();*

*glPopMatrix();*

*}*

*///\*\*\* House\_Two \*\*\*///*

*void house\_two() {*

*glPushMatrix();*

*glTranslatef(450, 0, 0);*

*house();*

*glPopMatrix();*

*}*

*///\*\*\* House\_Two \*\*\*///*

*void house\_three() {*

*glPushMatrix();*

*glTranslatef(320, 37, 0);*

*house();*

*glPopMatrix();*

*}*

*///\*\*\* Hill\_big\_One \*\*\*///*

*void Hill\_Big\_One() {*

*glPushMatrix();*

*glTranslatef(0, 0, 0);*

*hill\_big();*

*glPopMatrix();*

*}*

*///\*\*\* Hill\_big\_Two \*\*\*///*

*void Hill\_Big\_Two() {*

*glPushMatrix();*

*glTranslatef(550, -20, 0);*

*hill\_big();*

*glPopMatrix();*

*}*

*///\*\*\* Hill\_Small\_One \*\*\*///*

*void Hill\_Small\_One() {*

*glPushMatrix();*

*glTranslatef(0, 0, 0);*

*hill\_small();*

*glPopMatrix();*

*}*

*/// \*\*\* Hill\_One\_Model\_One \*\*\*///*

*void Hill\_One() {*

*glPushMatrix();*

*glTranslatef(0, 0, 0);*

*Hill\_One\_Model();*

*glPopMatrix();*

*}*

*/// \*\*\* Hill\_Two\_Model\_Two \*\*\*///*

*void Hill\_Two() {*

*glPushMatrix();*

*glTranslatef(0, 0, 0);*

*Hill\_Two\_Model();*

*glPopMatrix();*

*}*

*/// \*\*\* Hill\_Three\_Model\_Two \*\*\*///*

*void Hill\_Three() {*

*glPushMatrix();*

*glTranslatef(400, 0, 0);*

*Hill\_Two\_Model();*

*glPopMatrix();*

*}*

*/// \*\*\* Hill\_Four\_Model\_One \*\*\*///*

*void Hill\_Four() {*

*glColor3f(0.833, 1., 0.0);*

*glPushMatrix();*

*glTranslatef(380, 0, 0);*

*Hill\_One\_Model();*

*glPopMatrix();*

*}*

*///\*\*\* Tree\_1 \*\*\*///*

*void Tree\_One() {*

*glColor3f(0.533, 0.793, 0.0);*

*glPushMatrix();*

*glTranslatef(0, 0, 0);*

*Tree\_Model\_One();*

*glPopMatrix();*

*}*

*///\*\*\* Tree\_2 \*\*\*///*

*void Tree\_Two() {*

*glColor3f(0.533, 0.793, 0.0);*

*glPushMatrix();*

*glTranslatef(540, 0, 0);*

*Tree\_Model\_One();*

*glPopMatrix();*

*}*

*///\*\*\* Tree\_3 \*\*\*///*

*void Tree\_Three() {*

*glColor3f(0.533, .793, 0.0);*

*glPushMatrix();*

*glTranslatef(750, 0, 0);*

*Tree\_Model\_One();*

*glPopMatrix();*

*}*

*///\*\*\* Tree\_4 \*\*\*///*

*void Tree\_Four() {*

*glColor3f(0.533, .793, 0.0);*

*glPushMatrix();*

*glTranslatef(292, 40, 0);*

*Tree\_Model\_One();*

*glPopMatrix();*

*}*

*///\*\*\* Tree\_5 \*\*\*///*

*void Tree\_Five() {*

*glColor3f(0.533, .793, 0.0);*

*glPushMatrix();*

*glTranslatef(30, -20, 0);*

*Tree\_Model\_Two();*

*glPopMatrix();*

*}*

*///\*\*\* Tree\_6 \*\*\*///*

*void Tree\_Six() {*

*glColor3f(0.533, 0.793, 0.0);*

*glPushMatrix();*

*glTranslatef(50, -10, 0);*

*Tree\_Model\_Two();*

*glPopMatrix();*

*}*

*///\*\*\* Tree\_7 \*\*\*///*

*void Tree\_Seven() {*

*glColor3f(0.533, .793, 0.0);*

*glPushMatrix();*

*glTranslatef(322, 0, 0);*

*Tree\_Model\_Two();*

*glPopMatrix();*

*}*

*///\*\*\* Tree\_8 \*\*\*///*

*void Tree\_Eight() {*

*glColor3f(0.533, .793, 0.0);*

*glPushMatrix();*

*glTranslatef(350, -15, 0);*

*Tree\_Model\_Two();*

*glPopMatrix();*

*}*

*///\*\*\* Tree\_9 \*\*\*///*

*void Tree\_Nine() {*

*glColor3f(0.533, .793, 0.0);*

*glPushMatrix();*

*glTranslatef(760, -25, 0);*

*Tree\_Model\_Two();*

*glPopMatrix();*

*}*

*///\*\*\* Tree\_10 \*\*\*///*

*void Tree\_Ten() {*

*glColor3f(0.533, .793, 0.0);*

*glPushMatrix();*

*glTranslatef(90, -2, 0);*

*Tree\_Model\_Three();*

*glPopMatrix();*

*}*

*///\*\*\* Tree\_11 \*\*\*///*

*void Tree\_Eleven() {*

*glColor3f(0.533, .793, 0.0);*

*glPushMatrix();*

*glTranslatef(125, 0, 0);*

*Tree\_Model\_Three();*

*glPopMatrix();*

*}*

*///\*\*\* Tree\_12 \*\*\*///*

*void Tree\_Twelve() {*

*glColor3f(0.533, .793, 0.0);*

*glPushMatrix();*

*glTranslatef(408, -22, 0);*

*Tree\_Model\_Three();*

*glPopMatrix();*

*}*

*/// \*\*\* Windmill \*\*\*///*

*void Windmill\_One() {*

*glColor3f(0.11, 0.23, 0.36);*

*glPushMatrix();*

*glTranslatef(0, -10, 0);*

*Windmill();*

*glPopMatrix();*

*}*

*void Windmill\_Two() {*

*glColor3f(0.11, 0.23, 0.36);*

*glPushMatrix();*

*glTranslatef(508, -70, 0);*

*Windmill();*

*glPopMatrix();*

*}*

*void Windmill\_Three() {*

*glColor3f(0.11, 0.23, 0.36);*

*glPushMatrix();*

*glTranslatef(108, -90, 0);*

*Windmill();*

*glPopMatrix();*

*}*

*/// Model End*

*///=========================================================================================================///*

*///========================///*

*///\*\*\* Display Function \*\*\*///*

*///========================///*

*void display(void)*

*{*

*glClear(GL\_COLOR\_BUFFER\_BIT);*

*glColor3f(0.0, 0.0, 1.0);*

*///\*\*\* Object\_Layer \*\*\*///*

*Sun();*

*Windmill\_Three();*

*Hill\_Big\_One();*

*Hill\_Four();*

*house\_three();*

*Hill\_Big\_Two();*

*Hill\_Small\_One();*

*cloud\_three();*

*cloud\_four();*

*Windmill\_One();*

*Windmill\_Two();*

*Hill\_One();*

*Hill\_Two();*

*Hill\_Three();*

*house\_one();*

*cloud\_one();*

*house\_two();*

*Tree\_One();*

*Tree\_Two();*

*Tree\_Three();*

*Tree\_Four();*

*Tree\_Five();*

*Tree\_Six();*

*Tree\_Seven();*

*Tree\_Eight();*

*Tree\_Nine();*

*Tree\_Ten();*

*Tree\_Eleven();*

*Tree\_Twelve();*

*cloud\_two();*

*cloud\_five();*

*cloud\_six();*

*field();*

*glFlush();*

*}*

*///========================///*

*///\*\*\* Speed & Movement \*\*\*///*

*///========================///*

*///\*\*\* Sun\_Move \*\*\*///*

*void sun\_move() {*

*sun\_spin = sun\_spin + 0.002;*

*}*

*void move\_right() {*

*sun\_move();*

*spin = spin + .1;*

*ax = ax + .05;*

*bx = bx + .08;*

*cx = cx + .10;*

*dx = dx + .15;*

*if (cx > 1000) {*

*cx = -300;*

*}*

*if (bx > 1000) {*

*bx = -400;*

*}*

*if (cx > 1000) {*

*cx = -400;*

*}*

*if (dx > 1000) {*

*dx = -500;*

*}*

*glutPostRedisplay();*

*}*

*void mouse(int key, int state, int x, int y) {*

*switch (key)*

*{*

*case GLUT\_LEFT\_BUTTON:*

*if (state == GLUT\_DOWN)*

*{*

*glutIdleFunc(move\_right);*

*}*

*break;*

*case GLUT\_MIDDLE\_BUTTON:*

*case GLUT\_RIGHT\_BUTTON:*

*if (state == GLUT\_DOWN)*

*{*

*glutIdleFunc(NULL);*

*}*

*break;*

*default:*

*break;*

*}*

*}*

*int main(int argc, char\*\* argv)*

*{*

*glutInit(&argc, argv);*

*glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB);*

*glutInitWindowPosition(5, 5);*

*glutInitWindowSize(800, 500);*

*glutCreateWindow("Smart Village");*

*init();*

*glutDisplayFunc(display);*

*glutMouseFunc(mouse);*

*glutMainLoop();*

*}*

**OUTPUT / SCREENSHOTS**

Initial view



Second view



Third view

