

**KYUNGDONG UNIVERSITY (KDU)**

**House in Desert**

**Computer Graphics OPENGL Project**

***Bachelor of Smart computing***

**Prepared By: Submitted To:**

**Rasaili Tilak (1824052) Prof. Nur Alam**

**Submission On:**

**2019/06/7**

**TABLE OF CONTENTS**

Abstract………………………………………………………………….1

System Specifications…………………………………………………...2

Introduction to OpenGL…………………………………………………3

Implementation…………………………………………………………..4

Interaction………………………………………………………………..6

Source Code……………………………………………………………...7

Output…………………………………………………………………….86

Conclusion………………………………………………………………..87

Bibliography………………………………………………………………88

# Abstract

* Main aim of this Mini Project is to illustrate the concepts and usage of pre-built functions in OpenGL.
* Display of house in desert with some background like way and toilet is being done using computer graphics.
* The scene can be changed by pressing ‘d’ for day mode and ‘n’ for night mode. At night time sunset and house window color become white and another small house door become white back tree and side tree color also white windows.
* We have used input devices like mouse and key board to interact with program

# System specifications

* **SOFTWARE REQUIREMENTS :** 
  + MICROSOFT VISUAL C++
  + OPENGL
* **HARDWARE REQUIREMENT :** 
  + GRAPHICS SYSTEM,
  + Pentium P4 with 256 of Ram(Min)

# Introduction to openGL

Most of our application will be designed to access OpenGL directly through functions in three libraries. Functions in the main GL (or OpenGL in windows) library have names that begin with the letters gl and are stored in a library usually referred to as GL (or OpenGL in windows). The second is the OpenGL Utility Library (GLU). This library uses only GL functions but contains code for creating common objects and simplifying viewing. All functions in GLU can be created from the core GL library but application programmers prefer not to write the code repeatedly. The GLU library is available in all OpenGL implementations; functions in the GLU library begin with letters glu.

To interface with the window system and to get input from external devices into our programs, we need at least one more system-specific library that provides the “glue” between the window system and OpenGL. For the X window system, this library is functionality that should be expected in any modern windowing system.

Fig 1.1 shows the organization of the libraries for an X Window System environment. For this window system, GLUT will use GLX and the X libraries. The application program, however, can use only GLUT functions and thus can be recompiled with the GLUT library for other window systems.

GLU

Frame

Buffer

GL

Xlib,Xtk

GLUT

OpenGL

Application

program

GLX

**Fig1.1: Library Organization of OpenGL**

# Implementation

This program is implemented using various OpenGL functions which are shown below.

**Various functions used in this program.**

* glutInit() : interaction between the windowing system and OPENGL is initiated
* glutInitDisplayMode() : used when double buffering is required and depth information is required
* 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 co-ordinates
* glutKeyboardFunc() : handles normal ascii symbols
* glutSpecialFunc() : handles special keyboard keys
* glutReshapeFunc() : sets up the callback function for reshaping the window
* glutIdleFunc() : this handles the processing of the background
* glutDisplayFunc() : this handles redrawing of the window
* glutMainLoop() : this starts the main loop, it never returns
* glViewport() : used to set up the viewport
* glVertex3fv() : used to set up the points or vertices in three dimensions
* glColor3fv() : 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
* glTranslatef() : used to translate or move the rotation centre from one point to another in three dimensions
* glRotatef() : used to rotate an object through a specified rotation angle

# Interaction with program

This program includes interaction through keyboard.

* Front display Student Information
* Press “R” for rain.
* Press “D” in Keyboard for day.
* Press “N” in Keyboard for Night.

# Source Code

#include<cstdio>

#include <windows.h>

#include<math.h>

#include <vector>

#include <cstdlib>

# define PI 3.14159265358979323846

#include <GL/gl.h>

#include <GL/glut.h>

#include<MMSystem.h>

void PointLight(const float x, const float y, const float z, const float amb, const float diff, const float spec);

void PointLight(const float x, const float y, const float z, const float amb, const float diff, const float spec)

{

glEnable(GL\_LIGHTING);

GLfloat light\_ambient[] = { amb,amb,amb, 1.0 };

GLfloat light\_position[] = { -0.9,.9,0, 0.0 };

glLightfv(GL\_LIGHT0, GL\_AMBIENT, light\_ambient);

glLightfv(GL\_LIGHT0, GL\_POSITION, light\_position);

glEnable(GL\_LIGHT0); //enable the light after setting the properties

}

GLfloat position22 = 1.0f;

GLfloat speed22 = 0.007f;

void birdd(int value) {

if (position22 > 1.0)

position22 = -1.0f;

position22 += speed22;

glutPostRedisplay();

glutTimerFunc(100, birdd, 0);

}

GLfloat position4 = 0.0f;

GLfloat speed4 = -0.01f;

void sunn(int value)

{

if (position4 > 1.0)

position4 = 0.0f;

position4 += speed4;

glutPostRedisplay();

glutTimerFunc(100, sunn, 0);

}

GLfloat position3 = 0.0f;

GLfloat speed3 = -0.5f;

void rain(int value) {

if (position3 < -1.0)

position3 = 1.0f;

position3 += speed3;

glutPostRedisplay();

glutTimerFunc(100, rain, 0);

}

GLfloat position2 = 0.0f;

GLfloat speed2 = 0.004f;

void cloud(int value) {

if (position2 > 1.0)

position2 = -1.0f;

position2 += speed2;

glutPostRedisplay();

glutTimerFunc(100, cloud, 0);

}

GLfloat position1 = 1.0f;

GLfloat speed1 = -0.005f;

void cloud1()

{

int i;

GLfloat x = .5f; GLfloat y = .86f; GLfloat radius = .05f;

int triangleAmount = 20;

GLfloat twicePi = 2.0f \* PI;

glBegin(GL\_TRIANGLE\_FAN);

glColor3ub(255, 255, 255);

glVertex2f(x, y); // center of circle

for (i = 0; i <= triangleAmount; i++) {

glVertex2f(

x + (radius \* cos(i \* twicePi / triangleAmount)),

y + (radius \* sin(i \* twicePi / triangleAmount))

);

}

glEnd();

GLfloat a = .55f; GLfloat b = .83f;

glBegin(GL\_TRIANGLE\_FAN);

glColor3ub(255, 255, 255);

glVertex2f(a, b); // center of circle

for (i = 0; i <= triangleAmount; i++) {

glVertex2f(

a + (radius \* cos(i \* twicePi / triangleAmount)),

b + (radius \* sin(i \* twicePi / triangleAmount))

);

}

glEnd();

GLfloat c = .45f; GLfloat d = .83f;

glBegin(GL\_TRIANGLE\_FAN);

glColor3ub(255, 255, 255);

glVertex2f(c, d); // center of circle

for (i = 0; i <= triangleAmount; i++) {

glVertex2f(

c + (radius \* cos(i \* twicePi / triangleAmount)),

d + (radius \* sin(i \* twicePi / triangleAmount))

);

}

glEnd();

GLfloat e = .52f; GLfloat f = .8f;

glBegin(GL\_TRIANGLE\_FAN);

glColor3ub(255, 255, 255);

glVertex2f(e, f); // center of circle

for (i = 0; i <= triangleAmount; i++) {

glVertex2f(

e + (radius \* cos(i \* twicePi / triangleAmount)),

f + (radius \* sin(i \* twicePi / triangleAmount))

);

}

glEnd();

GLfloat g = .6f; GLfloat h = .82f;

glBegin(GL\_TRIANGLE\_FAN);

glColor3ub(255, 255, 255);

glVertex2f(g, h); // center of circle

for (i = 0; i <= triangleAmount; i++) {

glVertex2f(

g + (radius \* cos(i \* twicePi / triangleAmount)),

h + (radius \* sin(i \* twicePi / triangleAmount))

);

}

glEnd();

}

void cloud2()

{

// glLoadIdentity();

int i;

GLfloat x = -.5f; GLfloat y = .86f; GLfloat radius = .05f;

int triangleAmount = 20;

GLfloat twicePi = 2.0f \* PI;

glBegin(GL\_TRIANGLE\_FAN);

glColor3ub(245, 245, 245);

glVertex2f(x, y); // center of circle

for (i = 0; i <= triangleAmount; i++) {

glVertex2f(

x + (radius \* cos(i \* twicePi / triangleAmount)),

y + (radius \* sin(i \* twicePi / triangleAmount))

);

}

glEnd();

GLfloat a = -.55f; GLfloat b = .83f;

glBegin(GL\_TRIANGLE\_FAN);

glColor3ub(245, 245, 245);

glVertex2f(a, b); // center of circle

for (i = 0; i <= triangleAmount; i++) {

glVertex2f(

a + (radius \* cos(i \* twicePi / triangleAmount)),

b + (radius \* sin(i \* twicePi / triangleAmount))

);

}

glEnd();

GLfloat c = -.45f; GLfloat d = .83f;

glBegin(GL\_TRIANGLE\_FAN);

glColor3ub(245, 245, 245);

glVertex2f(c, d); // center of circle

for (i = 0; i <= triangleAmount; i++) {

glVertex2f(

c + (radius \* cos(i \* twicePi / triangleAmount)),

d + (radius \* sin(i \* twicePi / triangleAmount))

);

}

glEnd();

GLfloat e = -.52f; GLfloat f = .8f;

glBegin(GL\_TRIANGLE\_FAN);

glColor3ub(245, 245, 245);

glVertex2f(e, f); // center of circle

for (i = 0; i <= triangleAmount; i++) {

glVertex2f(

e + (radius \* cos(i \* twicePi / triangleAmount)),

f + (radius \* sin(i \* twicePi / triangleAmount))

);

}

glEnd();

GLfloat g = -.6f; GLfloat h = .82f;

glBegin(GL\_TRIANGLE\_FAN);

glColor3ub(245, 245, 245);

glVertex2f(g, h); // center of circle

for (i = 0; i <= triangleAmount; i++) {

glVertex2f(

g + (radius \* cos(i \* twicePi / triangleAmount)),

h + (radius \* sin(i \* twicePi / triangleAmount))

);

}

glEnd();

}

void sky()

{

glBegin(GL\_QUADS);

glColor3ub(51, 204, 255);

glVertex2f(-1.0f, 0.45f);

glVertex2f(1.0f, 0.45f);

glVertex2f(1.0f, 1.0f);

glVertex2f(-1.0f, 1.0f);

glEnd();

}

void sky2()

{

glBegin(GL\_QUADS);

glColor3ub(0, 51, 204);

glVertex2f(-1.0f, 0.45f);

glVertex2f(1.0f, 0.45f);

glVertex2f(1.0f, 1.0f);

glVertex2f(-1.0f, 1.0f);

glEnd();

}

void stars()

{

glPointSize(10.5);

glBegin(GL\_POINTS);

glColor3ub(255, 255, 255);

glVertex2f(0.8f, 0.95f);

glVertex2f(0.9f, 0.9f);

glVertex2f(0.95f, 0.7f);

glVertex2f(0.8f, 0.8f);

glVertex2f(0.7f, 0.9f);

glVertex2f(0.6f, 0.8f);

glVertex2f(0.5f, 0.75f);

glVertex2f(0.4f, 0.9f);

glVertex2f(0.3f, 0.7f);

glVertex2f(0.25f, 0.9f);

glVertex2f(0.25f, 0.7f);

glVertex2f(0.1f, 0.9f);

glVertex2f(0.15f, 0.75f);

glVertex2f(0.0f, 0.8f);

glVertex2f(-0.7f, 0.9f);

glVertex2f(-0.8f, 0.95f);

glVertex2f(-0.4f, 0.8f);

glVertex2f(-0.8f, 0.95f);

glVertex2f(-0.9f, 0.9f);

glVertex2f(-0.95f, 0.7f);

glVertex2f(-0.8f, 0.8f);

glVertex2f(-0.7f, 0.9f);

glVertex2f(-0.6f, 0.8f);

glVertex2f(-0.5f, 0.75f);

glVertex2f(-0.4f, 0.9f);

glVertex2f(-0.3f, 0.7f);

glVertex2f(-0.25f, 0.9f);

glVertex2f(-0.25f, 0.7f);

glVertex2f(-0.15f, 0.75f);

glVertex2f(-0.1f, 0.9f);

glEnd();

}

void sun()

{

int i;

GLfloat x = .0f; GLfloat y = .9f; GLfloat radius = .06f;

int triangleAmount = 20;

GLfloat twicePi = 2.0f \* PI;

glBegin(GL\_TRIANGLE\_FAN);

glColor3ub(255, 223, 0);

glVertex2f(x, y); // center of circle

for (i = 0; i <= triangleAmount; i++) {

glVertex2f(

x + (radius \* cos(i \* twicePi / triangleAmount)),

y + (radius \* sin(i \* twicePi / triangleAmount))

);

}

glEnd();

}

void moon()

{

int i;

GLfloat xx = -.5f; GLfloat yx = .93f; GLfloat radiusx = .06f;

int triangleAmount = 20;

GLfloat twicePi = 2.0f \* PI;

glBegin(GL\_TRIANGLE\_FAN);

glColor3ub(245, 245, 245);

glVertex2f(xx, yx); // center of circle

for (i = 0; i <= triangleAmount; i++) {

glVertex2f(

xx + (radiusx \* cos(i \* twicePi / triangleAmount)),

yx + (radiusx \* sin(i \* twicePi / triangleAmount))

);

}

glEnd();

GLfloat x = -.5f; GLfloat y = .9f; GLfloat radius = .06f;

glBegin(GL\_TRIANGLE\_FAN);

glColor3ub(255, 255, 255);

glVertex2f(x, y); // center of circle

for (i = 0; i <= triangleAmount; i++) {

glVertex2f(

x + (radius \* cos(i \* twicePi / triangleAmount)),

y + (radius \* sin(i \* twicePi / triangleAmount))

);

}

glEnd();

}

void ground()

{

glBegin(GL\_POLYGON);

glColor3ub(185, 141, 112);

glVertex2f(-1.0f, -1.0f);

glVertex2f(-1.0f, 0.0f);

glVertex2f(0.0f, 0.0f);

glVertex2f(0.1f, -0.03);

glVertex2f(0.2f, -0.07);

glVertex2f(0.3f, -0.1);//6

glVertex2f(0.2f, -0.13);

glVertex2f(0.1f, -0.17);

glVertex2f(0.2f, -0.2);

glVertex2f(0.35f, -0.23);

glVertex2f(0.25f, -0.25);

glVertex2f(0.18f, -0.28);//12

glVertex2f(0.3f, -0.32);

glVertex2f(0.2f, -0.35);

glVertex2f(0.4f, -0.4);

glVertex2f(0.4f, -0.6);

glVertex2f(0.2f, -0.65);

glVertex2f(0.3f, -0.7);

glVertex2f(0.2f, -0.75);

glVertex2f(0.4f, -0.8);

glVertex2f(0.2f, -0.85);

glVertex2f(0.35f, -0.9);

glVertex2f(0.25f, -0.95);

glVertex2f(0.4f, -1.0);

glEnd();

glBegin(GL\_QUADS);

glColor3ub(185, 141, 112);

glVertex2f(0.17f, -0.19);

glVertex2f(0.19f, -0.2f);

glVertex2f(0.3f, -0.12f);

glVertex2f(0.3f, -0.1);

glEnd();

glBegin(GL\_QUADS);

glColor3ub(185, 141, 112);

glVertex2f(0.23f, -0.295);

glVertex2f(0.25f, -0.305f);

glVertex2f(0.35f, -0.25f);

glVertex2f(0.35f, -0.23);

glEnd();

glBegin(GL\_QUADS);

glColor3ub(185, 141, 112);

glVertex2f(0.3f, -0.32);

glVertex2f(0.3f, -0.34);

glVertex2f(0.25f, -0.365f);

glVertex2f(0.2f, -0.35);

glEnd();

glBegin(GL\_QUADS);

glColor3ub(185, 141, 112);

glVertex2f(0.205f, -0.655);

glVertex2f(0.4f, -0.6);

glVertex2f(0.4f, -0.625);

glVertex2f(0.25f, -0.675);

glEnd();

glBegin(GL\_QUADS);

glColor3ub(185, 141, 112);

glVertex2f(0.3f, -0.7);

glVertex2f(0.3f, -0.72);

glVertex2f(0.24f, -0.7595);

glVertex2f(0.2f, -0.75);

glEnd();

glBegin(GL\_QUADS);

glColor3ub(185, 141, 112);

glVertex2f(0.4f, -0.8);

glVertex2f(0.4f, -0.825);

glVertex2f(0.24f, -0.865);

glVertex2f(0.2f, -0.85);

glEnd();

glBegin(GL\_QUADS);

glColor3ub(185, 141, 112);

glVertex2f(0.35f, -0.9);

glVertex2f(0.35f, -0.925);

glVertex2f(0.27f, -0.958);

glVertex2f(0.25f, -0.948);

glEnd();

}

void river()

{

glBegin(GL\_QUADS);

glColor3ub(185, 141, 112);

//glColor3ub(0,122,204);

glVertex2f(-1.0f, 0.45f);

glVertex2f(1.0f, 0.45f);

glVertex2f(1.0f, -1.0f);

glVertex2f(-1.0f, -1.0f);

glEnd();

}

void backgroundtreeB()

{

//pamtrees

glBegin(GL\_POLYGON);

glColor3ub(102, 51, 0);

glVertex2f(-0.52f, 0.45f);

glVertex2f(-0.48f, 0.45f);

glVertex2f(-0.48f, 0.62f);

glVertex2f(-0.5f, 0.64f);

glVertex2f(-0.52f, 0.62f);

glEnd();

glBegin(GL\_POLYGON);

glColor3ub(250,250,131);

glVertex2f(-0.5f, 0.64f);

glVertex2f(-0.52f, 0.62f);

glVertex2f(-0.58f, 0.6f);

glVertex2f(-0.55f, 0.63f);

glVertex2f(-0.58f, 0.66f);

glVertex2f(-0.52f, 0.66f);

glVertex2f(-0.52f, 0.72f);

glVertex2f(-0.5f, 0.69f);///

glVertex2f(-0.48f, 0.72f);

glVertex2f(-0.48f, 0.66f);

glVertex2f(-0.42f, 0.66f);

glVertex2f(-0.45f, 0.63f);

glVertex2f(-0.42f, 0.6f);

glVertex2f(-0.48f, 0.62f);

glEnd();

glBegin(GL\_POLYGON);

glColor3ub(102, 51, 0);

glVertex2f(0.52f, 0.45f);

glVertex2f(0.48f, 0.45f);

glVertex2f(0.48f, 0.62f);

glVertex2f(0.5f, 0.64f);

glVertex2f(0.52f, 0.62f);

glEnd();

}

void backgroundtreeA()

{

//pamtrees

glBegin(GL\_POLYGON);

glColor3ub(255,255,255);

glVertex2f(-0.52f, 0.45f);

glVertex2f(-0.48f, 0.45f);

glVertex2f(-0.48f, 0.62f);

glVertex2f(-0.5f, 0.64f);

glVertex2f(-0.52f, 0.62f);

glEnd();

glBegin(GL\_POLYGON);

glColor3ub(255,255,255);

glVertex2f(-0.5f, 0.64f);

glVertex2f(-0.52f, 0.62f);

glVertex2f(-0.58f, 0.6f);

glVertex2f(-0.55f, 0.63f);

glVertex2f(-0.58f, 0.66f);

glVertex2f(-0.52f, 0.66f);

glVertex2f(-0.52f, 0.72f);

glVertex2f(-0.5f, 0.69f);///

glVertex2f(-0.48f, 0.72f);

glVertex2f(-0.48f, 0.66f);

glVertex2f(-0.42f, 0.66f);

glVertex2f(-0.45f, 0.63f);

glVertex2f(-0.42f, 0.6f);

glVertex2f(-0.48f, 0.62f);

glEnd();

glBegin(GL\_POLYGON);

glColor3ub(255,255,255);

glVertex2f(0.52f, 0.45f);

glVertex2f(0.48f, 0.45f);

glVertex2f(0.48f, 0.62f);

glVertex2f(0.5f, 0.64f);

glVertex2f(0.52f, 0.62f);

glEnd();

}

void hut()

{

glBegin(GL\_POLYGON);

glColor3ub(132, 55, 42);

glVertex2f(-0.5f, 0.2f);

glVertex2f(-0.1f, 0.2f);

glVertex2f(-0.18f, 0.5f);

glVertex2f(-0.58f, 0.5f);

glEnd();

glBegin(GL\_POLYGON);

glColor3ub(93, 67, 44);

glVertex2f(-0.49f, 0.2f);

glVertex2f(-0.13f, 0.2f);

glVertex2f(-0.13f, -0.2f);

glVertex2f(-0.49f, -0.2f);

glEnd();

glBegin(GL\_POLYGON);

glColor3ub(132, 55, 42);

glVertex2f(-0.58f, 0.5f);

glVertex2f(-0.63f, 0.2f);

glVertex2f(-0.61f, 0.2f);

glVertex2f(-0.61f, -0.15f);

glVertex2f(-0.49f, -0.2f);

glVertex2f(-0.17f, -0.2f);

glEnd();

glLineWidth(2);

glBegin(GL\_LINES);

glColor3ub(0, 0, 0);

glVertex2f(-0.5f, 0.2f);

glVertex2f(-0.1f, 0.2f);

glColor3ub(0, 0, 0);

glVertex2f(-0.49f, 0.2f);

glVertex2f(-0.49f, -0.2f);

glColor3ub(0, 0, 0);

glVertex2f(-0.13f, -0.2f);

glVertex2f(-0.49f, -0.2f);

glColor3ub(0, 0, 0);

glVertex2f(-0.13f, -0.2f);

glVertex2f(-0.13f, 0.2f);

glColor3ub(0, 0, 0);

glVertex2f(-0.5f, 0.2f);

glVertex2f(-0.58f, 0.5f);

glColor3ub(0, 0, 0);

glVertex2f(-0.58f, 0.5f);

glVertex2f(-0.63f, 0.2f);

glColor3ub(0, 0, 0);

glVertex2f(-0.57f, 0.445f);

glVertex2f(-0.61f, 0.2f);

glColor3ub(0, 0, 0);

glVertex2f(-0.61f, 0.2f);

glVertex2f(-0.61f, -0.15f);

glColor3ub(0, 0, 0);

glVertex2f(-0.61f, -0.15f);

glVertex2f(-0.49f, -0.2f);

glColor3ub(0, 0, 0);

glVertex2f(-0.1f, 0.2f);

glVertex2f(-0.18f, 0.5f);

glColor3ub(0, 0, 0);

glVertex2f(-0.18f, 0.5f);

glVertex2f(-0.58f, 0.5f);

glEnd();

glBegin(GL\_POLYGON);

glColor3ub(181, 101, 29);

glVertex2f(-0.35f, 0.1f);

glVertex2f(-0.22f, 0.1f); //main door

glVertex2f(-0.22f, -0.2f);

glVertex2f(-0.35f, -0.2f);

glEnd();

glLineWidth(2);

glBegin(GL\_LINES);

glColor3ub(0, 0, 0);

glVertex2f(-0.35f, 0.1f);

glVertex2f(-0.22f, 0.1f);

glVertex2f(-0.22f, 0.1f);

glVertex2f(-0.22f, -0.2f);

glVertex2f(-0.35f, 0.1f);// main door line

glVertex2f(-0.35f, -0.2f);

glVertex2f(-0.285f, 0.1f);

glVertex2f(-0.285f, -0.2f);

glEnd();

glBegin(GL\_POLYGON);

glColor3ub(255, 255, 255);

glVertex2f(-0.44f, 0.05f);

glVertex2f(-0.38f, 0.05f); //left window

glVertex2f(-0.38f, -0.05f);

glVertex2f(-0.44f, -0.05f);

glEnd();

glLineWidth(2);

glBegin(GL\_LINES);

glColor3ub(0, 0, 0);

glVertex2f(-0.44f, 0.05f);

glVertex2f(-0.44f, -0.05f);

glVertex2f(-0.38f, 0.05f);

glVertex2f(-0.38f, -0.05f);

glVertex2f(-0.44f, 0.05f);// window draw line

glVertex2f(-0.38f, 0.05f);

glVertex2f(-0.44f, -0.05f);

glVertex2f(-0.38f, -0.05f);

glVertex2f(-0.41f, 0.05f);

glVertex2f(-0.41f, -0.05f);

glEnd();

/\*

glBegin(GL\_POLYGON);

glColor3ub(181, 101, 29);

glVertex2f(-0.51f, 0.12f); //2nd door

glVertex2f(-0.58f, 0.14f);

glVertex2f(-0.58f, -0.17f);

glVertex2f(-0.51f, -0.2f);

glEnd();

glLineWidth(2);

glBegin(GL\_LINES);

glColor3ub(0, 0, 0);

glVertex2f(-0.58f, 0.14f);

glVertex2f(-0.58f, -0.17f);

glVertex2f(-0.51f, 0.12f);

glVertex2f(-0.51f, -0.2f);

glVertex2f(-0.58f, 0.14f);

glVertex2f(-0.51f, 0.12f);

glVertex2f(-0.545f, 0.13f);

glVertex2f(-0.545f, -0.185f);

glEnd();\*/

glBegin(GL\_POLYGON);

glColor3ub(255, 255, 255);

glVertex2f(-0.14f, 0.05f);

glVertex2f(-0.2f, 0.05f); //rightwindow

glVertex2f(-0.2f, -0.05f);

glVertex2f(-0.14f, -0.05f);

glEnd();

glLineWidth(2);

glBegin(GL\_LINES);

glColor3ub(0, 0, 0);

glVertex2f(-0.14f, 0.05f);

glVertex2f(-0.14f, -0.05f);

glVertex2f(-0.2f, 0.05f);

glVertex2f(-0.2f, -0.05f);

glVertex2f(-0.14f, 0.05f);

glVertex2f(-0.2f, 0.05f);

glVertex2f(-0.14f, -0.05f);

glVertex2f(-0.2f, -0.05f);

glVertex2f(-0.17f, 0.05f);

glVertex2f(-0.17f, -0.05f);

glEnd();

}

void hut1()

{

glLineWidth(2);

glBegin(GL\_LINES);

glColor3ub(0, 0, 0);

glVertex2f(-0.35f, 0.1f);

glVertex2f(-0.22f, 0.1f);

glVertex2f(-0.22f, 0.1f);

glVertex2f(-0.22f, -0.2f);

glVertex2f(-0.35f, 0.1f);

glVertex2f(-0.35f, -0.2f);

glVertex2f(-0.285f, 0.1f);

glVertex2f(-0.285f, -0.2f);

glEnd();/\*

glBegin(GL\_POLYGON);

glColor3ub(153, 115, 0);

glVertex2f(-0.51f,0.12f); //left door

glVertex2f(-0.58f,0.14f);

glVertex2f(-0.58f,-0.17f);

glVertex2f(-0.51f,-0.2f);

glEnd();\*/

glLineWidth(2);

glBegin(GL\_LINES);

glColor3ub(0, 0, 0);

glVertex2f(-0.35f, 0.1f);

glVertex2f(-0.22f, 0.1f);

glVertex2f(-0.22f, 0.1f);

glVertex2f(-0.22f, -0.2f);

glVertex2f(-0.35f, 0.1f);

glVertex2f(-0.35f, -0.2f);

glVertex2f(-0.285f, 0.1f);

glVertex2f(-0.285f, -0.2f);

glEnd();

glBegin(GL\_POLYGON);

glColor3ub(255,255,255);

glVertex2f(-0.44f,0.05f);

glVertex2f(-0.38f,0.05f); //left window

glVertex2f(-0.38f,-0.05f);

glVertex2f(-0.44f,-0.05f);

glEnd();

glLineWidth(2);

glBegin(GL\_LINES);

glColor3ub(0, 0, 0);

glVertex2f(-0.44f, 0.05f);

glVertex2f(-0.44f, -0.05f);

glVertex2f(-0.38f, 0.05f);

glVertex2f(-0.38f, -0.05f);

glVertex2f(-0.44f, 0.05f);

glVertex2f(-0.38f, 0.05f);

glVertex2f(-0.44f, -0.05f);

glVertex2f(-0.38f, -0.05f);

glVertex2f(-0.41f, 0.05f);

glVertex2f(-0.41f, -0.05f);

glEnd();

/\*glLineWidth(2);

glBegin(GL\_LINES);

glColor3ub(0, 0, 0);

glVertex2f(-0.58f, 0.14f);

glVertex2f(-0.58f, -0.17f);

glVertex2f(-0.51f, 0.12f);

glVertex2f(-0.51f, -0.2f);

glVertex2f(-0.58f, 0.14f);

glVertex2f(-0.51f, 0.12f);

glVertex2f(-0.545f, 0.13f);

glVertex2f(-0.545f, -0.185f);

glEnd();\*/

glBegin(GL\_POLYGON);

glColor3ub(255,255,255);

glVertex2f(-0.14f,0.05f); ///right window

glVertex2f(-0.2f,0.05f);

glVertex2f(-0.2f,-0.05f);

glVertex2f(-0.14f,-0.05f);

glEnd();

glLineWidth(2);

glBegin(GL\_LINES);

glColor3ub(0, 0, 0);

glVertex2f(-0.14f, 0.05f);

glVertex2f(-0.14f, -0.05f);

glVertex2f(-0.2f, 0.05f);

glVertex2f(-0.2f, -0.05f);

glVertex2f(-0.14f, 0.05f);

glVertex2f(-0.2f, 0.05f);

glVertex2f(-0.14f, -0.05f);

glVertex2f(-0.2f, -0.05f);

glVertex2f(-0.17f, 0.05f);

glVertex2f(-0.17f, -0.05f);

glEnd();

glLineWidth(2);

glBegin(GL\_LINES);

glColor3ub(0, 0, 0);

glVertex2f(-0.5f, 0.2f);

glVertex2f(-0.1f, 0.2f);

glVertex2f(-0.49f, 0.2f);

glVertex2f(-0.49f, -0.2f);

glVertex2f(-0.13f, -0.2f);

glVertex2f(-0.49f, -0.2f);

glVertex2f(-0.13f, -0.2f);

glVertex2f(-0.13f, 0.2f);

glVertex2f(-0.5f, 0.2f);

glVertex2f(-0.58f, 0.5f);

glVertex2f(-0.58f, 0.5f);

glVertex2f(-0.63f, 0.2f);

glVertex2f(-0.57f, 0.445f);

glVertex2f(-0.61f, 0.2f);

glVertex2f(-0.61f, 0.2f);

glVertex2f(-0.61f, -0.15f);

glVertex2f(-0.61f, -0.15f);

glVertex2f(-0.49f, -0.2f);

glVertex2f(-0.1f, 0.2f);

glVertex2f(-0.18f, 0.5f);

glVertex2f(-0.18f, 0.5f);

glVertex2f(-0.58f, 0.5f);

glEnd();

glBegin(GL\_POLYGON);

glColor3ub(93, 67, 44);

glVertex2f(-0.5f, 0.2f);

glVertex2f(-0.1f, 0.2f);

glVertex2f(-0.18f, 0.5f);

glVertex2f(-0.58f, 0.5f);

glEnd();

glBegin(GL\_POLYGON);

glColor3ub(166,128,100);

glVertex2f(-0.49f, 0.2f);

glVertex2f(-0.13f, 0.2f);

glVertex2f(-0.13f, -0.2f);

glVertex2f(-0.49f, -0.2f);

glEnd();

glBegin(GL\_POLYGON);

glColor3ub(166, 128, 100);

glVertex2f(-0.58f, 0.5f);

glVertex2f(-0.63f, 0.2f);

glVertex2f(-0.61f, 0.2f);

glVertex2f(-0.61f, -0.15f);

glVertex2f(-0.49f, -0.2f);

glVertex2f(-0.17f, -0.2f);

glEnd();

}

void tree01()

{

glBegin(GL\_POLYGON);

glColor3ub(132, 55, 42);

glVertex2f(-0.72f, -0.15f);

glVertex2f(-0.65f, -0.2f);

glVertex2f(-0.735f, -0.17f);

glVertex2f(-0.74f, -0.25f);

glVertex2f(-0.775f, -0.17f);

glVertex2f(-0.85f, -0.2f);

glVertex2f(-0.78f, -0.15f);

//glVertex2f(-0.7f,-0.25f);

glEnd();

glBegin(GL\_QUADS);

glColor3ub(132, 55, 42);

glVertex2f(-0.78f, -0.15f);

glVertex2f(-0.78f, 0.23f);

glVertex2f(-0.72f, 0.23f);

glVertex2f(-0.72f, -0.15f);

glEnd();

int i;

GLfloat x = -.75f; GLfloat y = .33f; GLfloat radius = .06f;

int triangleAmount = 20;

GLfloat twicePi = 2.0f \* PI;

glEnd();

}

void tree0()

{

glBegin(GL\_POLYGON);

glColor3ub(255,255,255);

glVertex2f(-0.72f, -0.15f);

glVertex2f(-0.65f, -0.2f);

glVertex2f(-0.735f, -0.17f);

glVertex2f(-0.74f, -0.25f);

glVertex2f(-0.775f, -0.17f);

glVertex2f(-0.85f, -0.2f);

glVertex2f(-0.78f, -0.15f);

//glVertex2f(-0.7f,-0.25f);

glEnd();

glBegin(GL\_QUADS);

glColor3ub(255, 255, 255);

glVertex2f(-0.78f, -0.15f);

glVertex2f(-0.78f, 0.23f);

glVertex2f(-0.72f, 0.23f);

glVertex2f(-0.72f, -0.15f);

glEnd();

int i;

GLfloat x = -.75f; GLfloat y = .33f; GLfloat radius = .06f;

int triangleAmount = 20;

GLfloat twicePi = 2.0f \* PI;

glEnd();

}

void toileta()

{

glBegin(GL\_QUADS);

glColor3ub(98, 6, 4);

glVertex2f(0.40f, 0.1f);//a

glVertex2f(0.45f, 0.1f);//a

glVertex2f(0.45f, 0.1f);//a

glVertex2f(0.45f, 0.5f);//a//// right side

glVertex2f(0.45f, 0.5f);//a

glVertex2f(0.40f, 0.5f);//a

glVertex2f(0.40f, 0.5f);//a

glVertex2f(0.40f, 0.1f);//a

//LINE FOR SIDE

glBegin(GL\_LINES);

glColor3ub(0, 0, 0);

glVertex2f(0.40f, 0.1f);//a

glVertex2f(0.45f, 0.1f);//a

glVertex2f(0.45f, 0.1f);//a

glVertex2f(0.45f, 0.5f);//a//// right side

glVertex2f(0.45f, 0.5f);//a

glVertex2f(0.40f, 0.5f);//a

glVertex2f(0.40f, 0.5f);//a

glVertex2f(0.40f, 0.1f);//a

//below top

glBegin(GL\_QUADS);

glColor3ub(132,55,42);

glVertex2f(0.19f, 0.5f);//a

glVertex2f(0.41f, 0.5f);//a

glVertex2f(0.41f, 0.5f);//b r

glVertex2f(0.45f, 0.5f);//b

glVertex2f(0.45f, 0.5f);//b r

glVertex2f(0.25f, 0.5f);//b

glVertex2f(0.30f, 0.6f);//b r

glVertex2f(0.19f, 0.5f);//b

//LINE FOR TOP VIEW

glBegin(GL\_LINES);

glColor3ub(0,0,0);

glVertex2f(0.19f, 0.5f);//a

glVertex2f(0.41f, 0.5f);//a

glVertex2f(0.41f, 0.5f);//b r

glVertex2f(0.45f, 0.5f);//b

glVertex2f(0.45f, 0.5f);//b r

glVertex2f(0.25f, 0.5f);//b

glVertex2f(0.30f, 0.6f);//b r

glVertex2f(0.19f, 0.5f);//b

glEnd();

//door

glBegin(GL\_QUADS);

glColor3ub(255,255,255);

glVertex2f(0.24f, 0.1f);//

glVertex2f(0.32f, 0.1f);//a bot

glVertex2f(0.32f, 0.1f);//

glVertex2f(0.32f, 0.4f);//b r

glVertex2f(0.32f, 0.4f);//

glVertex2f(0.24f, 0.4f);//c t

glVertex2f(0.24f, 0.4f);// dl

glVertex2f(0.24f, 0.1f);// fornt pat

glEnd();

//line for door

glBegin(GL\_LINES);

glColor3ub(0, 0, 0);

glVertex2f(0.24f, 0.1f);//

glVertex2f(0.32f, 0.1f);//a bot

glVertex2f(0.32f, 0.1f);//

glVertex2f(0.32f, 0.4f);//b r

glVertex2f(0.32f, 0.4f);//

glVertex2f(0.24f, 0.4f);//c t

glVertex2f(0.28f, 0.4f);//

glVertex2f(0.28f, 0.1f);//c t

glVertex2f(0.24f, 0.4f);// dl

glVertex2f(0.24f, 0.1f);// fornt pat

glEnd();

//beow front

glBegin(GL\_QUADS);

glColor3ub(93, 67, 44);

glVertex2f(0.20f, 0.1f);//

glVertex2f(0.40f, 0.1f);//a bot

glVertex2f(0.40f, 0.1f);//

glVertex2f(0.40f, 0.5f);//b r

glVertex2f(0.40f, 0.5f);//

glVertex2f(0.20f, 0.5f);//c t

glVertex2f(0.20f, 0.5f);// dl

glVertex2f(0.20f, 0.1f);// fornt pat

glEnd();

// LINE FOR FRONT

glBegin(GL\_LINES);

glColor3ub(0,0,0);

glVertex2f(0.20f, 0.1f);//

glVertex2f(0.40f, 0.1f);//a bot

glVertex2f(0.40f, 0.1f);//

glVertex2f(0.40f, 0.5f);//b r

glVertex2f(0.40f, 0.5f);//

glVertex2f(0.20f, 0.5f);//c t

glVertex2f(0.20f, 0.5f);// dl

glVertex2f(0.20f, 0.1f);// fornt pat

glEnd();

//WAY TO TOILET

glBegin(GL\_QUADS);

glColor3ub(118, 122, 121);

glVertex2f(0.24f, 0.1f);//

glVertex2f(0.32f, 0.1f);//a bot

glVertex2f(0.32f, 0.1f);//

glVertex2f(0.18f, -0.30f);//b r

glVertex2f(0.18f, -0.30f);//

glVertex2f(0.10f, -0.30f);//c t

glVertex2f(0.10f, -0.30f);// dl

glVertex2f(0.24f, 0.1f);// fornt pat

glEnd();

glBegin(GL\_QUADS);

glColor3ub(118, 122, 121);

glVertex2f(0.10f, -0.30f);//

glVertex2f(0.18f, -0.30f);//a bot

glVertex2f(0.18f, -0.30f);//

glVertex2f(-0.4f, -0.43f);//b r

glVertex2f(-0.4f, -0.43f);//

glVertex2f(-0.3f, -0.35f);//c t

glVertex2f(-0.3f, -0.35f);// dl

glVertex2f(0.10f, -0.30f);// fornt pat

glEnd();

}

void toiletb()

{

glBegin(GL\_QUADS);

glColor3ub(98, 6, 4);

glVertex2f(0.40f, 0.1f);//a

glVertex2f(0.45f, 0.1f);//a

glVertex2f(0.45f, 0.1f);//a

glVertex2f(0.45f, 0.5f);//a//// right side

glVertex2f(0.45f, 0.5f);//a

glVertex2f(0.40f, 0.5f);//a

glVertex2f(0.40f, 0.5f);//a

glVertex2f(0.40f, 0.1f);//a

//LINE FOR SIDE

glBegin(GL\_LINES);

glColor3ub(0, 0, 0);

glVertex2f(0.40f, 0.1f);//a

glVertex2f(0.45f, 0.1f);//a

glVertex2f(0.45f, 0.1f);//a

glVertex2f(0.45f, 0.5f);//a//// right side

glVertex2f(0.45f, 0.5f);//a

glVertex2f(0.40f, 0.5f);//a

glVertex2f(0.40f, 0.5f);//a

glVertex2f(0.40f, 0.1f);//a

//below top

glBegin(GL\_QUADS);

glColor3ub(132, 55, 42);

glVertex2f(0.19f, 0.5f);//a

glVertex2f(0.41f, 0.5f);//a

glVertex2f(0.41f, 0.5f);//b r

glVertex2f(0.45f, 0.5f);//b

glVertex2f(0.45f, 0.5f);//b r

glVertex2f(0.25f, 0.5f);//b

glVertex2f(0.30f, 0.6f);//b r

glVertex2f(0.19f, 0.5f);//b

//LINE FOR TOP VIEW

glBegin(GL\_LINES);

glColor3ub(0, 0, 0);

glVertex2f(0.19f, 0.5f);//a

glVertex2f(0.41f, 0.5f);//a

glVertex2f(0.41f, 0.5f);//b r

glVertex2f(0.45f, 0.5f);//b

glVertex2f(0.45f, 0.5f);//b r

glVertex2f(0.25f, 0.5f);//b

glVertex2f(0.30f, 0.6f);//b r

glVertex2f(0.19f, 0.5f);//b

glEnd();

//beow front

glBegin(GL\_QUADS);

glColor3ub(93, 67, 44);

glVertex2f(0.20f, 0.1f);//

glVertex2f(0.40f, 0.1f);//a bot

glVertex2f(0.40f, 0.1f);//

glVertex2f(0.40f, 0.5f);//b r

glVertex2f(0.40f, 0.5f);//

glVertex2f(0.20f, 0.5f);//c t

glVertex2f(0.20f, 0.5f);// dl

glVertex2f(0.20f, 0.1f);// fornt pat

glEnd();

// LINE FOR FRONT

glBegin(GL\_LINES);

glColor3ub(0, 0, 0);

glVertex2f(0.20f, 0.1f);//

glVertex2f(0.40f, 0.1f);//a bot

glVertex2f(0.40f, 0.1f);//

glVertex2f(0.40f, 0.5f);//b r

glVertex2f(0.40f, 0.5f);//

glVertex2f(0.20f, 0.5f);//c t

glVertex2f(0.20f, 0.5f);// dl

glVertex2f(0.20f, 0.1f);// fornt pat

glEnd();

//door

glBegin(GL\_QUADS);

glColor3ub(150, 67, 44);

glVertex2f(0.24f, 0.1f);//

glVertex2f(0.32f, 0.1f);//a bot

glVertex2f(0.32f, 0.1f);//

glVertex2f(0.32f, 0.4f);//b r

glVertex2f(0.32f, 0.4f);//

glVertex2f(0.24f, 0.4f);//c t

glVertex2f(0.24f, 0.4f);// dl

glVertex2f(0.24f, 0.1f);// fornt pat

glEnd();

//line for door

glBegin(GL\_LINES);

glColor3ub(0, 0, 0);

glVertex2f(0.24f, 0.1f);//

glVertex2f(0.32f, 0.1f);//a bot

glVertex2f(0.32f, 0.1f);//

glVertex2f(0.32f, 0.4f);//b r

glVertex2f(0.32f, 0.4f);//

glVertex2f(0.24f, 0.4f);//c t

glVertex2f(0.28f, 0.4f);//

glVertex2f(0.28f, 0.1f);//c t

glVertex2f(0.24f, 0.4f);// dl

glVertex2f(0.24f, 0.1f);// fornt pat

glEnd();

//WAY TO TOILET

glBegin(GL\_QUADS);

glColor3ub(118, 122, 121);

glVertex2f(0.24f, 0.1f);//

glVertex2f(0.32f, 0.1f);//a bot

glVertex2f(0.32f, 0.1f);//

glVertex2f(0.18f, -0.30f);//b r

glVertex2f(0.18f, -0.30f);//

glVertex2f(0.10f, -0.30f);//c t

glVertex2f(0.10f, -0.30f);// dl

glVertex2f(0.24f, 0.1f);// fornt pat

glEnd();

glBegin(GL\_QUADS);

glColor3ub(118, 122, 121);

glVertex2f(0.10f, -0.30f);//

glVertex2f(0.18f, -0.30f);//a bot

glVertex2f(0.18f, -0.30f);//

glVertex2f(-0.4f, -0.43f);//b r

glVertex2f(-0.4f, -0.43f);//

glVertex2f(-0.3f, -0.35f);//c t

glVertex2f(-0.3f, -0.35f);// dl

glVertex2f(0.10f, -0.30f);// fornt pat

glEnd();

}

void way()

{

glBegin(GL\_QUADS);

glColor3ub(118, 122, 121);

glVertex2f(-0.35f, -0.2f);

glVertex2f(-0.22f, -0.2f);

glVertex2f(-0.28f, -0.5f);

glVertex2f(-0.43f, -0.5f);

glEnd();

glBegin(GL\_QUADS);

glColor3ub(118, 122, 121);

glVertex2f(-0.43f, -0.5f);

glVertex2f(-0.75f, -1.0f);

glVertex2f(-0.56f, -1.0f);

glVertex2f(-0.28f, -0.5f);

glEnd();

}

void fence()

{

glLineWidth(4);

glBegin(GL\_LINES);

glColor3ub(166, 128, 100);

glVertex2f(-1.0f, -0.1f);

glVertex2f(-0.6f, -0.1f);

glColor3ub(93, 67, 44);

glVertex2f(-1.0f, -0.05f);

glVertex2f(-0.6f, -0.05f);

glColor3ub(166, 128, 100);

glVertex2f(-1.0f, 0.0f);

glVertex2f(-0.6f, 0.0f);

glColor3ub(93, 67, 44);

glVertex2f(-1.0f, 0.05f);

glVertex2f(-0.6f, 0.05f);

glColor3ub(166, 128, 100);

glVertex2f(-1.0f, 0.1f);

glVertex2f(-0.6f, 0.1f);

glColor3ub(93, 67, 44);

glVertex2f(-0.95f, 0.13f);

glVertex2f(-0.95f, -0.12f);

glColor3ub(166, 128, 100);

glVertex2f(-0.9f, 0.13f);

glVertex2f(-0.9f, -0.12f);

glColor3ub(93, 67, 44);

glVertex2f(-0.85f, 0.13f);

glVertex2f(-0.85f, -0.12f);

glColor3ub(166, 128, 100);

glVertex2f(-0.8f, 0.13f);

glVertex2f(-0.8f, -0.12f);

glColor3ub(166, 128, 100);

glVertex2f(-0.75f, 0.13f);

glVertex2f(-0.75f, -0.12f);

glColor3ub(93, 67, 44);

glVertex2f(-0.7f, 0.13f);

glVertex2f(-0.7f, -0.12f);

glColor3ub(93, 67, 44);

glVertex2f(-0.65f, 0.13f);

glVertex2f(-0.65f, -0.12f);

glEnd();

}

void grass4()

{

int i;

int triangleAmount = 20;

GLfloat twicePi = 2.0f \* PI;

GLfloat e = -.05f; GLfloat f = -.35f; GLfloat radius11 = .02f;

glBegin(GL\_TRIANGLE\_FAN);

glColor3ub(255, 255, 255);

glVertex2f(e, f); // center of circle

for (i = 0; i <= triangleAmount; i++) {

glVertex2f(

e + (radius11 \* cos(i \* twicePi / triangleAmount)),

f + (radius11 \* sin(i \* twicePi / triangleAmount))

);

}

glEnd();

GLfloat g = 0.05f; GLfloat h = -0.35f;

glBegin(GL\_TRIANGLE\_FAN);

glColor3ub(255, 255, 255);

glVertex2f(g, h); // center of circle

for (i = 0; i <= triangleAmount; i++) {

glVertex2f(

g + (radius11 \* cos(i \* twicePi / triangleAmount)),

h + (radius11 \* sin(i \* twicePi / triangleAmount))

);

}

glEnd();

GLfloat a1 = 0.0f; GLfloat b1 = -0.3f;

glBegin(GL\_TRIANGLE\_FAN);

glColor3ub(255, 255, 255);

glVertex2f(a1, b1); // center of circle

for (i = 0; i <= triangleAmount; i++) {

glVertex2f(

a1 + (radius11 \* cos(i \* twicePi / triangleAmount)),

b1 + (radius11 \* sin(i \* twicePi / triangleAmount))

);

}

glEnd();

glLineWidth(4);

glBegin(GL\_LINES);

glColor3ub(255, 255, 255);

glVertex2f(-.05f, -0.35f);

glVertex2f(-0.0f, -0.4f);//

glVertex2f(0.05f, -0.35f);

glVertex2f(0.0f, -0.4f);//

glVertex2f(0.027f, -0.33f);

glVertex2f(0.0f, -0.4f);//

glVertex2f(-0.027f, -0.33f);

glVertex2f(0.0f, -0.4f);//

glVertex2f(0.0f, -0.3f);

glVertex2f(0.0f, -0.4f);//

glVertex2f(-0.075f, -0.37f);

glVertex2f(-0.0f, -0.4f);//

glVertex2f(0.0745f, -0.37f);

glVertex2f(-0.0f, -0.4f);//

glEnd();

}

void grass5()

{

int i;

int triangleAmount = 20;

GLfloat twicePi = 2.0f \* PI;

GLfloat e = -.05f; GLfloat f = -.65f; GLfloat radius11 = .02f;

glBegin(GL\_TRIANGLE\_FAN);

glColor3ub(255, 255, 255);

glVertex2f(e, f); // center of circle

for (i = 0; i <= triangleAmount; i++) {

glVertex2f(

e + (radius11 \* cos(i \* twicePi / triangleAmount)),

f + (radius11 \* sin(i \* twicePi / triangleAmount))

);

}

glEnd();

GLfloat g = 0.05f; GLfloat h = -0.65f;

glBegin(GL\_TRIANGLE\_FAN);

glColor3ub(255, 255, 255);

glVertex2f(g, h); // center of circle

for (i = 0; i <= triangleAmount; i++) {

glVertex2f(

g + (radius11 \* cos(i \* twicePi / triangleAmount)),

h + (radius11 \* sin(i \* twicePi / triangleAmount))

);

}

glEnd();

GLfloat a1 = 0.0f; GLfloat b1 = -0.6f;

glBegin(GL\_TRIANGLE\_FAN);

glColor3ub(255, 255, 255);

glVertex2f(a1, b1); // center of circle

for (i = 0; i <= triangleAmount; i++) {

glVertex2f(

a1 + (radius11 \* cos(i \* twicePi / triangleAmount)),

b1 + (radius11 \* sin(i \* twicePi / triangleAmount))

);

}

glEnd();

glLineWidth(4);

glBegin(GL\_LINES);

glColor3ub(255, 255, 255);

glVertex2f(-.05f, -0.65f);

glVertex2f(-0.0f, -0.7f);

glVertex2f(0.05f, -0.65f);

glVertex2f(0.0f, -0.7f);

glVertex2f(0.027f, -0.62f);

glVertex2f(0.0f, -0.7f);

glVertex2f(-0.027f, -0.62f);

glVertex2f(0.0f, -0.7f);

glVertex2f(-0.0f, -0.6f);

glVertex2f(0.0f, -0.7f);

glVertex2f(-0.075f, -0.67f);

glVertex2f(-0.0f, -0.7f);

glVertex2f(0.075f, -0.67f);

glVertex2f(-0.0f, -0.7f);

glEnd();

}

void grass6()

{

int i;

int triangleAmount = 20;

GLfloat twicePi = 2.0f \* PI;

GLfloat e = -.85f; GLfloat f = -.75f; GLfloat radius11 = .02f;

glBegin(GL\_TRIANGLE\_FAN);

glColor3ub(255, 255, 255);

glVertex2f(e, f); // center of circle

for (i = 0; i <= triangleAmount; i++) {

glVertex2f(

e + (radius11 \* cos(i \* twicePi / triangleAmount)),

f + (radius11 \* sin(i \* twicePi / triangleAmount))

);

}

glEnd();

GLfloat g = -0.75f; GLfloat h = -0.75f;

glBegin(GL\_TRIANGLE\_FAN);

glColor3ub(255, 255, 255);

glVertex2f(g, h); // center of circle

for (i = 0; i <= triangleAmount; i++) {

glVertex2f(

g + (radius11 \* cos(i \* twicePi / triangleAmount)),

h + (radius11 \* sin(i \* twicePi / triangleAmount))

);

}

glEnd();

GLfloat a1 = -0.8f; GLfloat b1 = -0.7f;

glBegin(GL\_TRIANGLE\_FAN);

glColor3ub(255, 255, 255);

glVertex2f(a1, b1); // center of circle

for (i = 0; i <= triangleAmount; i++) {

glVertex2f(

a1 + (radius11 \* cos(i \* twicePi / triangleAmount)),

b1 + (radius11 \* sin(i \* twicePi / triangleAmount))

);

}

glEnd();

glLineWidth(4);

glBegin(GL\_LINES);

glColor3ub(255, 255, 255);

glVertex2f(-.85f, -0.75f);

glVertex2f(-0.8f, -0.8f);

glVertex2f(-0.75f, -0.75f);

glVertex2f(-0.8f, -0.8f);

glVertex2f(-0.827f, -0.72f);

glVertex2f(-0.8f, -0.8f);

glVertex2f(-0.773f, -0.72f);

glVertex2f(-0.8f, -0.8f);

glVertex2f(-0.8f, -0.7f);

glVertex2f(-0.8f, -0.8f);

glVertex2f(-0.725f, -0.77f);

glVertex2f(-0.8f, -0.8f);

glVertex2f(-0.875f, -0.77f);

glVertex2f(-0.8f, -0.8f);

glEnd();

}

void well()

{

glBegin(GL\_POLYGON);

glColor3ub(68, 255, 255);

glVertex2f(-0.9f, -0.35f);

glVertex2f(-0.85f, -0.375f);

glVertex2f(-0.8f, -0.38f);

glVertex2f(-0.7f, -0.38f);

glVertex2f(-0.65f, -0.375f);

glVertex2f(-0.6f, -0.35f);

glVertex2f(-0.65f, -0.33f);

glVertex2f(-0.7f, -0.325f);

glVertex2f(-0.8f, -0.325f);

glVertex2f(-0.85f, -0.33f);

glEnd();

glLineWidth(5);

glBegin(GL\_LINES);

glColor3ub(68, 255, 255);

glVertex2f(-0.9f, -0.35f);

glVertex2f(-0.85f, -0.33f);//

glVertex2f(-0.85f, -0.33f);

glVertex2f(-0.8f, -0.325f);//

glVertex2f(-0.8f, -0.325f);

glVertex2f(-0.7f, -0.325f);//

glVertex2f(-0.7f, -0.325f);

glVertex2f(-0.65f, -0.33f);//

glVertex2f(-0.65f, -0.33f);

glVertex2f(-0.6f, -0.35f);

glEnd();

}

void StartingText()

{

char text[120];

//sprintf(text, "BEAUTY OF NATURE BEAUTIFUL VILLAGE SCENARIO", 5.00, 8.00);

glColor3f(0, 0, 0);

glRasterPos2f(-20, 12);

for (int i = 0; text[i] != '\0'; i++)

{

if (text[i] == ' ' && text[i + 1] == ' ')

{

glutBitmapCharacter(GLUT\_BITMAP\_TIMES\_ROMAN\_24, text[i]);

glRasterPos2f(-32, 02);

}

else glutBitmapCharacter(GLUT\_BITMAP\_TIMES\_ROMAN\_24, text[i]);

}

//sprintf(text, " PRESS D FOR DAY VIEW,PRESS N FOR NIGHT VIEW,PRESS R FOR RAIN VIEW", 5.00, 8.00);

glColor3f(0, 0, 0);

glRasterPos2f(-100, 0);

for (int i = 0; text[i] != '\0'; i++)

{

if (text[i] == ' ' && text[i + 1] == ' ')

{

glutBitmapCharacter(GLUT\_BITMAP\_TIMES\_ROMAN\_10, text[i]);

glRasterPos2f(-30.5, -10);

}

else glutBitmapCharacter(GLUT\_BITMAP\_TIMES\_ROMAN\_10, text[i]);

}

//sprintf(text, " CREATED BY-", 5.00, 8.00);

glColor3f(0, 0, 0);

glRasterPos2f(-100, 0);

for (int i = 0; text[i] != '\0'; i++)

{

if (text[i] == ' ' && text[i + 1] == ' ')

{

glutBitmapCharacter(GLUT\_BITMAP\_TIMES\_ROMAN\_10, text[i]);

glRasterPos2f(18, -25);

}

else glutBitmapCharacter(GLUT\_BITMAP\_TIMES\_ROMAN\_10, text[i]);

}

//sprintf(text, " ISLAM,MD,MAHIDUL", 5.00, 8.00);

glColor3f(0, 0, 0);

glRasterPos2f(-100, 0);

for (int i = 0; text[i] != '\0'; i++)

{

if (text[i] == ' ' && text[i + 1] == ' ')

{

glutBitmapCharacter(GLUT\_BITMAP\_TIMES\_ROMAN\_10, text[i]);

glRasterPos2f(18, -30);

}

else glutBitmapCharacter(GLUT\_BITMAP\_TIMES\_ROMAN\_10, text[i]);

}

}

void DrawSphere()

{

glColorMaterial(GL\_FRONT\_AND\_BACK, GL\_AMBIENT\_AND\_DIFFUSE);

glEnable(GL\_COLOR\_MATERIAL);

glColor4f(93, 67, 44, 0.0f);

glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT);

glEnable(GL\_DEPTH\_TEST);

glLoadIdentity();

glPushMatrix();

glTranslatef(position2, 0.0f, 0.0f);

//cloud3();

glPopMatrix();

moon();

//sun();

//hut();

hut1();

tree0();

tree01();

//treetilak();

backgroundtreeA();

backgroundtreeB();

fence();

grass4();

grass5();

//grass6();

way();

toileta();

toiletb();

well();

//Straw();

ground();

river();

glPushMatrix();

glTranslatef(0.0f, position4, 0.0f);

sun();

glPopMatrix();

stars();

sky2();

glFlush();

}

void display2()

{

glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT);

glEnable(GL\_DEPTH\_TEST);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

PointLight(0, 0, 1, 0, 1, 1);

DrawSphere();

glLoadIdentity();

glutSwapBuffers();

}

struct Point

{

float x, y;

unsigned char r, g, b, a;

};

std::vector< Point > points;

void display1(void)

{

glClearColor(0.0f, 0.0f, 0.0f, 1.0f);

glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

glLineWidth(2);

sky();

cloud1();

cloud2();

//cloud3();

backgroundtreeA();

backgroundtreeB();

river();

ground();

//grass1();

//grass2();

//grass3();

way();

//boat2();

fence();

tree0();

tree01();

well();

hut();

toileta();

toiletb();

//huta();

//hutb();

glOrtho(-50, 50, -50, 50, -1, 1);

glMatrixMode(GL\_MODELVIEW);

glLoadIdentity();

// draw

glPushMatrix();

glTranslatef(0.0f, position3, 0.0f);

glColor3ub(255, 255, 255);

glEnableClientState(GL\_VERTEX\_ARRAY);

glEnableClientState(GL\_COLOR\_ARRAY);

glVertexPointer(2, GL\_FLOAT, sizeof(Point), &points[0].x);

glColorPointer(4, GL\_UNSIGNED\_BYTE, sizeof(Point), &points[0].r);

glPointSize(2.5);

glDrawArrays(GL\_POINTS, 0, points.size());

glDisableClientState(GL\_VERTEX\_ARRAY);

glDisableClientState(GL\_COLOR\_ARRAY);

glPopMatrix();

glFlush();

glutSwapBuffers();

}

void display() {

glClearColor(0.0f, 0.0f, 0.0f, 1.0f);

glClear(GL\_COLOR\_BUFFER\_BIT);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

glLineWidth(2);

sky();

sun();

glPushMatrix();

glTranslatef(position2, 0.0f, 0.0f);

cloud1();

cloud2();

glPopMatrix();

glPushMatrix();

glTranslatef(position22, 0.0f, 0.0f);

//bird();

glPopMatrix();

backgroundtreeA();

backgroundtreeB();

//hometilak();

river();

glPushMatrix();

glTranslatef(position1, 0.0f, 0.0f);

glPopMatrix();

ground();

//grass1();

//grass2();

//grass3();

way();

toileta();

toiletb();

fence();

tree0();

tree01();

//treetilak();

//treea();

well();

//Straw();

hut();

//hutb();

glFlush();

glutSwapBuffers();

}

void reshape(int w, int h)

{

// std::cout<<"Reshape is called"<<std::endl;

float aspectRatio = (float)w / (float)h;

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluPerspective(145, aspectRatio, 1.0, 100.0);

glMatrixMode(GL\_MODELVIEW);

}

void Display(void)

{

//std::cout<<"Display 1 called"<<std::endl;

glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT);

glLoadIdentity();

glTranslatef(0, 0, -20);

//StartingText();

glFlush();

glutSwapBuffers();

}

void init(void)

{

glClearColor(1.0f, 1.0f, 1.0f, 1.0f);

glClearDepth(1.0);

glEnable(GL\_DEPTH\_TEST);

glEnable(GL\_LIGHTING);

glShadeModel(GL\_SMOOTH);

glEnable(GL\_COLOR\_MATERIAL);

glColorMaterial(GL\_FRONT, GL\_AMBIENT\_AND\_DIFFUSE);

glEnable(GL\_LIGHT0);

//std::cout<<"Init is called"<<std::endl;

}

void handleKeypress(unsigned char key, int x, int y) {

switch (key) {

case 'd':

glutDestroyWindow(1);

glutInitWindowSize(1240, 750);

glutInitWindowPosition((glutGet(GLUT\_SCREEN\_WIDTH) - 1240) / 2, (glutGet(GLUT\_SCREEN\_HEIGHT) - 750) / 2);

glutCreateWindow("RASAILI TILAK");

glutKeyboardFunc(handleKeypress);

glutDisplayFunc(display);

PlaySound(TEXT("bird-2.wav"), NULL, SND\_ASYNC);

break;

break;

case 'r':

glutDestroyWindow(1);

glutInitWindowSize(1240, 750);

glutInitWindowPosition((glutGet(GLUT\_SCREEN\_WIDTH) - 1240) / 2, (glutGet(GLUT\_SCREEN\_HEIGHT) - 750) / 2);

glutCreateWindow("RASAILI TILAK");

glutKeyboardFunc(handleKeypress);

glutDisplayFunc(display1);

PlaySound(TEXT("rain-07.wav"), NULL, SND\_ASYNC);

break;

case 'n':

glutDestroyWindow(1);

glutInitWindowSize(1240, 750);

glutInitWindowPosition((glutGet(GLUT\_SCREEN\_WIDTH) - 1240) / 2, (glutGet(GLUT\_SCREEN\_HEIGHT) - 750) / 2);

glutCreateWindow("RASAILI TILAK");

glutKeyboardFunc(handleKeypress);

glutDisplayFunc(display2);

PlaySound(TEXT("cricket-2.wav"), NULL, SND\_ASYNC);

glutPostRedisplay();

break;

case 'D':

glutDestroyWindow(1);

glutInitWindowSize(1240, 750);

glutInitWindowPosition((glutGet(GLUT\_SCREEN\_WIDTH) - 1240) / 2, (glutGet(GLUT\_SCREEN\_HEIGHT) - 750) / 2);

glutCreateWindow("RASAILI TILAK");

glutKeyboardFunc(handleKeypress);

glutDisplayFunc(display);

PlaySound(TEXT("bird-2.wav"), NULL, SND\_ASYNC);

break;

break;

case 'R':

glutDestroyWindow(1);

glutInitWindowSize(1240, 750);

glutInitWindowPosition((glutGet(GLUT\_SCREEN\_WIDTH) - 1240) / 2, (glutGet(GLUT\_SCREEN\_HEIGHT) - 750) / 2);

glutCreateWindow("RASAILI TILAK");

glutKeyboardFunc(handleKeypress);

glutDisplayFunc(display1);

PlaySound(TEXT("rain-07.wav"), NULL, SND\_ASYNC);

break;

case 'N':

glutDestroyWindow(1);

glutInitWindowSize(1240, 750);

glutInitWindowPosition((glutGet(GLUT\_SCREEN\_WIDTH) - 1240) / 2, (glutGet(GLUT\_SCREEN\_HEIGHT) - 750) / 2);

glutCreateWindow("RASAILI TILAK");

glutKeyboardFunc(handleKeypress);

glutDisplayFunc(display2);

PlaySound(TEXT("cricket-2.wav"), NULL, SND\_ASYNC);

glutPostRedisplay();

}

}

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

{

glutInit(&argc, argv);

glutInitWindowSize(1240, 750);

glutInitWindowPosition((glutGet(GLUT\_SCREEN\_WIDTH) - 1240) / 2, (glutGet(GLUT\_SCREEN\_HEIGHT) - 750) / 2);

glutCreateWindow("RASAILI TILAK");

init();

glutReshapeFunc(reshape);

glutDisplayFunc(Display);

for (size\_t i = 0; i < 1000; ++i)

{

Point pt;

pt.x = -50 + (rand() % 100);

pt.y = -50 + (rand() % 100);

pt.r = 255;

pt.g = 255;

pt.b = 255;

pt.a = 255;

points.push\_back(pt);

}

glutTimerFunc(100, cloud, 0);

glutTimerFunc(100, sunn, 0);

// glutTimerFunc(100, boat, 0);

glutTimerFunc(100, rain, 0);

// glutTimerFunc(100, birdd, 0);

glutKeyboardFunc(handleKeypress);

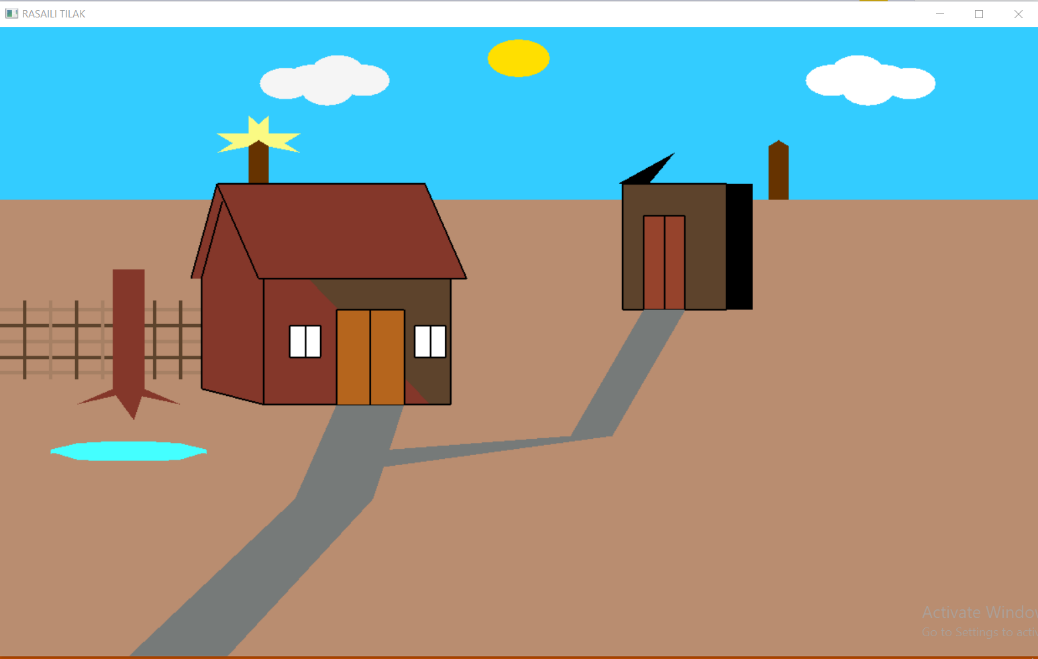
init();

glutMainLoop();

return 0;

}

# OUTPUT OF THE PROGRAM





Day view

Fig (1)

Night view

Fig (2)

# Conclusions

The project “House in Desert” clearly demonstrates the display desert environment. In day and Night time.

Finally, we conclude that this program clearly illustrates the House in Desert using openGL and has been completed successfully and is ready to be demonstrated.

# Bibliography

WE HAVE OBTAINED INFORMATION FROM MANY RESOURCES TO DESIGN AND IMPLEMENT OUR PROJECT SUCCESSIVELY. WE HAVE ACQUIRED MOST OF THE KNOWLEDGE FROM RELATED WEBSITES. THE FOLLOWING ARE SOME OF THE RESOURCES:

* TEXT BOOKS :

INTERACTIVE COMPUTER GRAPHICS A TOP-DOWN APPROACH

-By Edward Angel.

* COMPUTER GRAPHICS,PRINCIPLES & PRACTICES
  + Foley van dam
  + Feiner hughes
* WEB REFERENCES: <http://jerome.jouvie.free.fr/OpenGl/Lessons/Lesson3.php>[http://google.com](http://google.com/) [http://opengl.org](http://opengl.org/)