Project Report on Front View of Ahsan Manzil

Computer Graphics

Group: H

8/18/21

Group Members

Name	ID
SAHA JUNIOR SAMAY	18-37168-1
ISMAIL HOSSAIN MUZUMDER	18-37052-1
FATEMATUJ JOHURA	18-37846-2
MD HANJALAYEAMIN	18-36166-1
TASNUBA KAMAL	19-39741-1

1. Introduction

Graphics provides one of the most natural means of communicating with a computer since our highly developed 2D and 3D pattern-recognition abilities allow us to perceive and process pictorial data rapidly and efficiently. Interactive computer graphics is the most important means of producing pictures since the invention of photography and television. It has the added advantage that, with the computer, we can take photos of concrete real-world objects and abstract, synthetic objects, such as mathematical surfaces and data that have no inherent geometry, such as survey results. Using this editor, you can draw and paint using the mouse. It can also perform a host of other functions like drawing lines, circles, polygons, etc. Interactive picture construction techniques such as essential positioning, rubber-band, dragging, and drawing are used. Block operations like cut, copy, and paste are supported to simultaneously edit large areas of the workspace. It is user-friendly and intuitive to use.

2. Problem Statement

Here, We have decided to build a realistic view of Ahsan Manzil. The main problem occurred when we tried to create our separate header files. We've build 2 different header module name, 'components.h' & 'shapes.h'. We've designed the whole code on a GLUT project. Here, we need to make a scene in a 2D format. We had to use various kinds of built-in functions, and in most cases, we need to build our own framework for reusability.

3. Objective

Each of our project members wanted to design an entirely different creative project where graphics of historic structure will be painted on. So we came up with an idea to design THE AHSAN MANZIL. Here our goal is to create a design of our historical structure which is AHSAN MANZIL. We also tried to implement some animated feature which is moving vehicles. We tried to design a road in front of the AHSAN MANZIL and some moving cars. We tried to implement some animated effects on this project. First, we create a mind map to relate words and images to the event. Then we Brainstorm the concepts and developed quick ideas in written form. After that, we tried to implement our visions into a graphical representation. Our main objective is to show photographic or illustrative imagery. We tried to measure and enforce perfect sizing and position. That's why we had to run various scathes first. Our project shows the general shape of a graphic element: the AHSAN MANZIL with a road in front of it and some moving vehicles. We tried to find resources to solve technical problems. We read some reference books, took help from online resources, communicated with our project members, and tried to figure out the solution. We discussed and choose the appropriate codes and software for implementing this project. Finally, we achieved a design that is well crafted and presentable.

4. System Implementation

Source code of main.cpp

```
#include <windows.h>
#include<mmsystem.h>
#include "components.h"

/*  Variable Declaration    */
GLfloat rotation_increaser = 0.0f;
GLfloat position = 0.0f;
GLfloat speed = 0.1f;
GLfloat car_pos = 0.0f;
GLfloat car_speed = 0.079f;
GLfloat car3_pos = 0.0f;
GLfloat car4_pos = 0.78f;

GLfloat cloud_position = -0.20f;
GLfloat cloud_speed = 0.008f;
GLfloat rainPos= -1.0;

GLfloat ship_pos = 0.78f;
GLfloat ship_pos_1 = -0.78f;
GLfloat ship_speed = 0.005f;

GLfloat autoNightTimer = 0.0f;
GLfloat autoNightSpeed = 0.05f;

GLfloat autoRainTimer = 0.0f;
```

```
 GLfloat autoRainSpeed = 0.05f;

 bool isDay = true;
 bool isRainy = false;
 bool isHide = false;
 bool isFullScreen = false;
 bool isAuto = true;


 void autoView()
 {
     if(autoNightTimer > 25)
     {
         isDay = !isDay;
         autoNightTimer = 0.0f;
     }
     autoNightTimer += autoNightSpeed;

     if(autoRainTimer > 10)
     {
         isRainy = !isRainy;
         (isRainy) ? PlaySound("assets/rain.wav", NULL,
SND_FILENAME|SND_ASYNC|SND_LOOP) : PlaySound(NULL, 0, 0);
         autoRainTimer = 0.0f;
     }
     autoRainTimer +=autoRainSpeed;
 }



 void update(int value)
 {

     if(position <-1.0)
         position = 1.0f;
     position -= speed;

     if(car_pos > 1.0)
         car_pos = -1.2f;
     car_pos += car_speed;


     if(car3_pos < -1.0)
         car3_pos = 1.2f;
     car3_pos -= car_speed;
```

```
    if(car4_pos < -1.0)
        car4_pos = 1.2f;
    car4_pos -= car_speed;

    if(cloud_position < -1.2)
        cloud_position = 1.2f;
    cloud_position -=cloud_speed;

    if(rainPos<-1.0)
        rainPos = -1.0;
    rainPos -=1.47;

    if(ship_pos < -1.8)
        ship_pos = 1.2f;
    ship_pos -= ship_speed;

    if(ship_pos_1 > 1.3)
        ship_pos_1 = -1.4f;
    ship_pos_1 += ship_speed;

    (isAuto) ? autoView() : blank();

    glutPostRedisplay();
    glutTimerFunc(100, update, 0);
}


void Idle()
{
    glutPostRedisplay();
}

void initState()
{
    glClearColor(0.5f, 1.0f, 1.0f, 0.0f);
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    glEnable(GL_COLOR_MATERIAL | GL_LIGHTING);
    glLineWidth(4);
    glPointSize(7);
    glClear(GL_COLOR_BUFFER_BIT );
    GLfloat global_ambient[] = {1.9,0.0,0.0, 0.1};
    glLightModelfv(GL_LIGHT_MODEL_AMBIENT, global_ambient);
}
```

```
void axisDraw()
{
    glBegin(GL_LINES);
    glColor3ub(0, 0, 0);
    glVertex2f(1.0f, 0.0f);
    glVertex2f(-1.0, 0.0f);
    glVertex2f(0.0f, 1.0f);
    glVertex2f(0.0, -1.0f);
    glEnd();
}

/*  Main Building's Code Starts From Here*/
void ahsanMonjilView()
{
    anyQuad(0.8f, 0.5f, 0.8f, 0.1f, -0.8f, 0.1f, -0.8f, 0.5f, 255, 99, 71);;

    glBegin(GL_LINES);
    glColor3ub(255, 99, 71);
    glVertex2f(-0.81f, 0.51f);
    glVertex2f(0.86f, 0.51f);
    glEnd();


    glLineWidth(10);
    glBegin(GL_LINES);
    glColor3ub(255, 99, 71);
    glVertex2f(-0.55f, 0.53f);
    glVertex2f(0.6f, 0.53f);
    glEnd();
    glLineWidth(4);

    //  MIDDLE BODY
    glLoadIdentity();
    glColor3ub(255, 99, 71);
    anyQuad(-0.06, 0.54, -0.06, 0.3, 0.11, 0.3, 0.11, 0.54, 255, 99, 71);
    glLoadIdentity();

    //  MAIN DOOR
    glLoadIdentity();
    glTranslatef(0.736, -0.1245, 0);
    glScalef(1, 1.5, 0);
    ovalWindowComponent(178, 34, 34);
    glLoadIdentity();

    //  LEFT DOOR
    glLoadIdentity();
    glTranslatef(0.32, -0.066, 0);
    glScalef(0.5, 1.3, 0);
    ovalWindowComponent(178, 34, 34);
    glLoadIdentity();

    //  RIGHT DOOR
    glLoadIdentity();
    glTranslatef(0.44, -0.066, 0);
    glScalef(0.5, 1.3, 0);
    ovalWindowComponent(178, 34, 34);
    glLoadIdentity();

    //  MAIN DOME => RIGHT
    glLoadIdentity();
    glColor3ub(255, 99, 71);
    glBegin(GL_POLYGON);
    glVertex2f(-0.06, 0.655);
    glVertex2f(0.025, 0.655);
    glVertex2f(0.025, 0.8);
    glVertex2f(0.0, 0.79);
    glVertex2f(-0.025, 0.76);
    glVertex2f(-0.04, 0.735);
    glVertex2f(-0.05, 0.71);
    glEnd();

    glLoadIdentity();
    //  Top Point
    anyQuad(0.02, 0.8, 0.03, 0.8, 0.027, 0.815, 0.023, 0.815, 255, 99, 71);
    glLoadIdentity();

    glLoadIdentity();
    glColor3ub(0, 0, 0);
    glLineWidth(2);
    lineComponent(0.025, 0.815, 0.025, 0.83);
    glLoadIdentity();

    //  MAIN DOME => LEFT
    glLoadIdentity();
    glColor3ub(255, 99, 71);
    glBegin(GL_POLYGON);
    glVertex2f(0.11, 0.655);
    glVertex2f(0.025, 0.655);
    glVertex2f(0.025, 0.8);
    glVertex2f(0.05, 0.79);
    glVertex2f(0.075, 0.76);
    glVertex2f(0.09, 0.735);
```

```
     glVertex2f(0.10, 0.71);
     glEnd();

     glLoadIdentity();
     //  DOME => MIDDLE BORDER
     glColor3ub(0, 0, 0);
     glTranslatef(0, 0.02, 0);
     glBegin(GL_POLYGON);
     glVertex2f(-0.065, 0.63);
     glVertex2f(-0.06, 0.54);
     glVertex2f(0.11, 0.54);
     glVertex2f(0.115, 0.63);
     glVertex2f(0.085, 0.65);
     glVertex2f(0.06, 0.66);
     glVertex2f(0.025, 0.665);
     glVertex2f(-0.005, 0.66);
     glVertex2f(-0.03, 0.65);
     glEnd();
     glLoadIdentity();

     glLoadIdentity();
     //  DOME => LOWER PART
     glColor3ub(255, 99, 71);
     glBegin(GL_POLYGON);
     glVertex2f(-0.06, 0.63);
     glVertex2f(-0.06, 0.54);
     glVertex2f(0.11, 0.54);
     glVertex2f(0.11, 0.63);
     glVertex2f(0.085, 0.65);
     glVertex2f(0.06, 0.66);
     glVertex2f(0.025, 0.665);
     glVertex2f(-0.005, 0.66);
     glVertex2f(-0.03, 0.65);
     glEnd();
     glLoadIdentity();



     //  => MIDDLE LEFT PILLAR
     glLoadIdentity();
     glColor3ub(255,140,0);
     anyQuad(-0.09, 0.59, -0.084, 0.564, -0.062, 0.564, -0.057, 0.59,
255,140,0);
     glLoadIdentity();
     glTranslatef(0.237, 0.24, 0);
     glScalef(0.4, 0.64, 0);
```

```
    pillarComponent();
    glLoadIdentity();

    //DecorLines
    glLoadIdentity();
    glColor3ub(255,140,0);
    glLineWidth(5);
    lineComponent(-0.015, 0.575, 0.062, 0.575);
    glLoadIdentity();

    glLoadIdentity();
    glColor3ub(0,0,0);
    glLineWidth(3);
    lineComponent(-0.011, 0.57, -0.011, 0.48);
    lineComponent(0.059, 0.57, 0.059, 0.48);
    glLoadIdentity();

    glLoadIdentity();
    glColor3ub(0,0,0);
    glLineWidth(2);
    lineComponent(0.05, 0.57, 0.05, 0.5);
    lineComponent(0.038, 0.57, 0.038, 0.52);
    lineComponent(0.025, 0.57, 0.025, 0.54);
    lineComponent(0.013, 0.57, 0.013, 0.52);
    lineComponent(0.0, 0.57, 0.0, 0.5);
    glLoadIdentity();

    glLoadIdentity();
    glColor3ub(255,140,0);
    glLineWidth(3);
    lineComponent(-0.06, 0.545, -0.013, 0.545);
    lineComponent(0.064, 0.545, 0.11, 0.545);
    glLoadIdentity();

    glLoadIdentity();
    glColor3ub(0,0,0);
    glLineWidth(1);
    lineComponent(-0.0525, 0.54, -0.0525, 0.52);
    lineComponent(-0.04, 0.54, -0.04, 0.52);
    lineComponent(-0.025, 0.54, -0.025, 0.52);
    lineComponent(0.0725, 0.54, 0.0725, 0.52);
    lineComponent(0.085, 0.54, 0.085, 0.52);
    lineComponent(0.1, 0.54, 0.1, 0.52);
    glLoadIdentity();

    glLoadIdentity();
```

```
     glColor3ub(0,0,0);
     glLineWidth(3);
     lineComponent(-0.05, 0.65, -0.05, 0.546);
     lineComponent(-0.013, 0.67, -0.013, 0.58);
     lineComponent(0.061, 0.67, 0.061, 0.58);
     lineComponent(0.1, 0.65, 0.1, 0.543);
     glLoadIdentity();

     glLoadIdentity();
     //  => TOP DOOR
     glTranslatef(0.308, 0.49, 0);
     glScalef(0.4, 0.3, 0);
     ovalWindowComponent(178, 34, 34);
     glLoadIdentity();

     //  => Middle Right Pillar
     glLoadIdentity();
     glColor3ub(255,140,0);
     glTranslatef(0.197, 0, 0);
     anyQuad(-0.09, 0.59, -0.084, 0.564, -0.062, 0.564, -0.057, 0.59,
255,140,0);
     glLoadIdentity();
     glTranslatef(0.434, 0.24, 0);
     glScalef(0.4, 0.64, 0);
     pillarComponent();
     glLoadIdentity();

     glLoadIdentity();
     glTranslatef(0.15, -0.09, 0);
     threeDotComponent();
     glLoadIdentity();

     glLoadIdentity();
     glTranslatef(0.25, -0.09, 0);
     threeDotComponent();
     glLoadIdentity();

     glLoadIdentity();
     glTranslatef(0.35, -0.09, 0);
     threeDotComponent();
     glLoadIdentity();

     glLoadIdentity();
     glTranslatef(0.45, -0.09, 0);
     threeDotComponent();
     glLoadIdentity();
```

```
    glLoadIdentity();
    glTranslatef(0.55, -0.09, 0);
    threeDotComponent();
    glLoadIdentity();


    glLoadIdentity();
    glTranslatef(0.82, -0.09, 0);
    threeDotComponent();
    glLoadIdentity();

    glLoadIdentity();
    glTranslatef(0.92, -0.09, 0);
    threeDotComponent();
    glLoadIdentity();

    glLoadIdentity();
    glTranslatef(1.02, -0.09, 0);
    threeDotComponent();
    glLoadIdentity();

    glLoadIdentity();
    glTranslatef(1.12, -0.09, 0);
    threeDotComponent();
    glLoadIdentity();

    glLoadIdentity();
    glTranslatef(1.22, -0.09, 0);
    threeDotComponent();
    glLoadIdentity();

    glLoadIdentity();
    pillarComponent();
    glLoadIdentity();

    glLoadIdentity();
    glTranslatef(0.2, 0, 0);
    pillarComponent();
    glLoadIdentity();

    glLoadIdentity();
    glTranslatef(1.4, 0, 0);
    pillarComponent();
    glLoadIdentity();
    glLoadIdentity();
    glTranslatef(1.6, 0, 0);
    pillarComponent();
    glLoadIdentity();

    glLoadIdentity();
    gombujhComponent();
    glLoadIdentity();

    glLoadIdentity();
    glTranslatef(1.4, 0, 0);
    gombujhComponent();
    glLoadIdentity();

    glLoadIdentity();
    squareWindowComponent(178, 34, 34);
    glLoadIdentity();

    // LEFT WINDOW
    glLoadIdentity();
    squareWindowComponent(178, 34, 34);
    glLoadIdentity();

    glLoadIdentity();
    glTranslatef(0.07, 0, 0);
    squareWindowComponent(178, 34, 34);
    glLoadIdentity();

    glLoadIdentity();
    glTranslatef(0, -0.15, 0);
    ovalWindowComponent(178, 34, 34);
    glLoadIdentity();

    glLoadIdentity();
    glTranslatef(0.07, -0.15, 0);
    ovalWindowComponent(178, 34, 34);
    glLoadIdentity();

    // => RIGHT WINDOW
    glLoadIdentity();
    glTranslatef(1.4, 0, 0);
    squareWindowComponent(178, 34, 34);
    glLoadIdentity();

    glLoadIdentity();
    glTranslatef(1.47, 0, 0);
    squareWindowComponent(178, 34, 34);
    glLoadIdentity();

    glLoadIdentity();
    glTranslatef(1.4, -0.15, 0);
    ovalWindowComponent(178, 34, 34);
    glLoadIdentity();

    glLoadIdentity();
    glTranslatef(1.47, -0.15, 0);
    ovalWindowComponent(178, 34, 34);
    glLoadIdentity();

    //  Bottom Row of Left Window
    glLoadIdentity();
    glTranslatef(0.2, -0.2, 0);
    ovalWindowComponent(0, 0, 0);
    glLoadIdentity();

    glLoadIdentity();
    glTranslatef(0.3, -0.2, 0);
    ovalWindowComponent(0, 0, 0);
    glLoadIdentity();

    glLoadIdentity();
    glTranslatef(0.4, -0.2, 0);
    ovalWindowComponent(0, 0, 0);
    glLoadIdentity();

    glLoadIdentity();
    glTranslatef(0.5, -0.2, 0);
    ovalWindowComponent(0, 0, 0);
    glLoadIdentity();

    //  Upper Row of Left Window
    glLoadIdentity();
    glTranslatef(0.2, 0, 0);
    ovalWindowComponent(0, 0, 0);
    glLoadIdentity();

    glLoadIdentity();
    glTranslatef(0.3, 0, 0);
    ovalWindowComponent(0, 0, 0);
    glLoadIdentity();

    glLoadIdentity();
    glTranslatef(0.4, 0, 0);
    ovalWindowComponent(0, 0, 0);
    glLoadIdentity();

    glLoadIdentity();
    glTranslatef(0.5, 0, 0);
    ovalWindowComponent(0, 0, 0);
    glLoadIdentity();

    glLoadIdentity();
    glTranslatef(0.6, 0, 0);
    ovalWindowComponent(0, 0, 0);
    glLoadIdentity();


    //  Bottom Row of Right Window
    glLoadIdentity();
    glTranslatef(1.27, -0.2, 0);
    ovalWindowComponent(0, 0, 0);
    glLoadIdentity();

    glLoadIdentity();
    glTranslatef(1.17, -0.2, 0);
    ovalWindowComponent(0, 0, 0);
    glLoadIdentity();

    glLoadIdentity();
    glTranslatef(1.07, -0.2, 0);
    ovalWindowComponent(0, 0, 0);
    glLoadIdentity();

    glLoadIdentity();
    glTranslatef(0.97, -0.2, 0);
    ovalWindowComponent(0, 0, 0);
    glLoadIdentity();


    //  Upper Row of Right Window
    glLoadIdentity();
    glTranslatef(1.27, 0, 0);
    ovalWindowComponent(0, 0, 0);
    glLoadIdentity();

    glLoadIdentity();
    glTranslatef(1.17, 0, 0);
    ovalWindowComponent(0, 0, 0);
    glLoadIdentity();

    glLoadIdentity();
    glTranslatef(1.07, 0, 0);
    ovalWindowComponent(0, 0, 0);
    glLoadIdentity();

    glLoadIdentity();
    glTranslatef(0.97, 0, 0);
    ovalWindowComponent(0, 0, 0);
    glLoadIdentity();

    glLoadIdentity();
    glTranslatef(0.87, 0, 0);
    ovalWindowComponent(0, 0, 0);
    glLoadIdentity();

    //  Stairs
    anyQuad(-0.08f, 0.3f, 0.13f, 0.3f, 0.29f, 0.0f, -0.25f, 0.0f, 36, 33, 36);

    anyQuad(-0.17f, 0.15f, 0.21f, 0.15f, 0.29f, 0.0f, -0.25f, 0.0f, 53, 56,
57);

    glLineWidth(7);
    glBegin(GL_LINES);
    glColor3ub(35,43,43);
    glVertex2f(0.13f, 0.3f);
    glVertex2f(-0.08f, 0.3f);

    glVertex2f(0.16f, 0.25f);
    glVertex2f(-0.11f, 0.25f);

    glVertex2f(0.19f, 0.20f);
    glVertex2f(-0.14f, 0.20f);

    glVertex2f(0.21f, 0.15f);
    glVertex2f(-0.17f, 0.15f);

    glVertex2f(0.24f, 0.10f);
    glVertex2f(-0.20f, 0.10f);

    glVertex2f(0.27f, 0.05f);
    glVertex2f(-0.22f, 0.05f);

    glVertex2f(0.29f, 0.0f);
    glVertex2f(-0.25f, 0.0f);
    glEnd();
    glLineWidth(4);


}

// The River
void WaterView(int R, int G, int B)
{

    anyQuad(-1.0f, -0.22f, -1.0f, -1.0f, 1.0f, -1.0f,1.0f, -0.22f, R, G, B);

    glBegin(GL_POLYGON);

    glColor3ub(R + 2, G + 3, B + 10);
    glVertex2f(1.0f, -0.8f);
    glVertex2f(-1.0f, -0.8f);
    glVertex2f(-1.0f, -0.8f);
    glVertex2f(-0.8f, -0.8f);
    glVertex2f(-0.6f, -0.7f);
    glVertex2f(-0.4f, -0.7f);
    glVertex2f(-0.2f, -0.8f);
    glVertex2f(-0.0f, -0.7f);
    glVertex2f(0.2f, -0.7f);
    glVertex2f(0.4f, -0.8f);
    glVertex2f(0.6f, -0.8f);
    glVertex2f(0.8f, -0.7f);
    glVertex2f(1.0f, -0.7f);
    glEnd();

    //  Waves
    glBegin(GL_LINE_STRIP);
    glColor3ub(R, G, B);
    glVertex2f(1.0f, -0.8f);
    glVertex2f(-1.0f, -0.8f);
    glVertex2f(-1.0f, -0.7f);
    glVertex2f(-0.8f, -0.8f);
    glVertex2f(-0.6f, -0.7f);
    glVertex2f(-0.4f, -0.7f);
    glVertex2f(-0.2f, -0.8f);
    glVertex2f(-0.0f, -0.7f);
    glVertex2f(0.2f, -0.7f);
    glVertex2f(0.4f, -0.8f);
    glVertex2f(0.6f, -0.8f);
    glVertex2f(0.8f, -0.7f);
    glVertex2f(1.0f, -0.7f);
    glEnd();

    glLoadIdentity();
    glPushMatrix();
    glTranslatef(ship_pos_1,0,0);
    glScalef(0.6, 0.6, 0);
    shipComponent();
    glPopMatrix();
    glLoadIdentity();

    glLoadIdentity();
    glPushMatrix();
    glTranslatef(ship_pos,0,0);
    glScalef(0.9, 0.9, 0);
    shipComponent();
    glPopMatrix();
    glLoadIdentity();
}

// Road Between Main Building & River
void roadView(int R, int G, int B)
{
    glLoadIdentity();
    glTranslatef(0.025, 0.0, 0);
    lampComponent(R, G, B);
    glLoadIdentity();
    //road-base

    glLoadIdentity();
    glTranslatef(-0.2, 0.0, 0);
    lampComponent(R, G, B);
    glLoadIdentity();

    glLoadIdentity();
    glTranslatef(0.3, 0.0, 0);
    lampComponent(R, G, B);
    glLoadIdentity();


    glLoadIdentity();
    glTranslatef(1.15, 0.0, 0);
    lampComponent(R, G, B);
    glLoadIdentity();


    glLoadIdentity();
```

```
     glTranslatef(1.4, 0.0, 0);
     lampComponent(R, G, B);
     glLoadIdentity();

     glLoadIdentity();
     glTranslatef(1.6, 0.0, 0);
     lampComponent(R, G, B);
     glLoadIdentity();



     anyQuad(-1.0f, 0.0f, -1.0f, -0.22f, 1.0f, -0.22f, 1.0f, 0.0f, 145, 145,
145);

     //  Border of the Road
     glLineWidth(6.0);
     glBegin(GL_LINES);
     glColor3ub(255,255,255);
     glVertex2f(-1.0f, -0.005f);//building front road border
     glVertex2f(1.0f, -0.005f);
     glEnd();

     //  Lines of the Road
     float x1, x2, y;
     x1=-1.0;
     x2=-0.92;
     y=-0.0999;
     glLineWidth(6.0);

     glBegin(GL_LINES);

     for(int i =0; i<17; i++)
     {
         glVertex2f(x1,y );
         glVertex2f(x2,y);

         x1=x1+.12;
         x2=x2+.12;
     }

     glEnd();

     //  Import from "components.h"
     credit();

     glLoadIdentity();
```

```
    glPushMatrix();
    glTranslatef(car_pos,0,0);
    carComponent();
    glPopMatrix();
    glLoadIdentity();

    glLoadIdentity();
    glPushMatrix();
    glTranslatef(car_pos,0,0);
    car_2Component();
    glPopMatrix();
    glLoadIdentity();

    glLoadIdentity();
    glPushMatrix();
    glTranslatef(car3_pos,0,0);
    car_3Component();
    glPopMatrix();
    glLoadIdentity();

    glLoadIdentity();
    glPushMatrix();
    glTranslatef(car4_pos,0,0);
    car_4Component();
    glPopMatrix();
    glLoadIdentity();

    glLineWidth(6.0);
    glBegin(GL_LINES);
    glColor3ub(255,255,255);
    glVertex2f(-1.0f, -0.21f);//building front road border
    glVertex2f(1.0f, -0.21f);
    glEnd();
    glLineWidth(4.0);
}

    // Background Mountain View
void mountainView(int R, int G, int B)
{

    glLoadIdentity();
    //  Mountain
    glColor3ub(R, G, B);
    glBegin(GL_POLYGON);
    glVertex2f(-1, 0.5);
    glVertex2f(1, 0.5);
    glVertex2f(0.9, 0.7);
    glVertex2f(0.8, 0.55);
    glVertex2f(0.5, 0.75);
    glVertex2f(0.3, 0.55);
    glVertex2f(0.1, 0.88);
    glVertex2f(-0.1, 0.66);
    glVertex2f(-0.3, 0.79);
    glVertex2f(-0.6, 0.63);
    glVertex2f(-0.7, 0.71);
    glVertex2f(-0.9, 0.59);
    glEnd();//endOfMountain
    glLoadIdentity();

    grassComponent();
}

    // Side Buildings
void buildingsView(int R, int G, int B )
{
    buildingComponent();
    glTranslatef(0.03f, 0.1f, 0);
    //  Tall Building One
    glScalef(0.5, 1.1, 0);
    anyQuad(-0.7f, 0.7f, -0.7f, 0.3f, -0.9f, 0.3f, -0.9f, 0.7f, 163, 163, 117);
    glLoadIdentity();

    float wp1 = -0.195, wp2 = 0.755;
    for(int w = 0; w < 5; w++)
    {
        for(int wj = 0; wj < 8; wj++)
        {
            //  Small Window of Tall Building One
            glLoadIdentity();
            glTranslatef(wp1, wp2, 0);
            glScalef(0.3, 0.25, 0);
            squareWindowComponent(R, G, B);
            glLoadIdentity();
            wp2 -= 0.045;
        }
        wp2 = 0.755;
        wp1 += 0.018;
    }

    glLoadIdentity();
    //  Tall Building Two
    glTranslatef(0.13f, 0.1f, 0);
    glScalef(0.4, 1.05, 0);
    anyQuad(-0.7f, 0.7f, -0.7f, 0.3f, -0.9f, 0.3f, -0.9f, 0.7f, 194, 194, 163);
    glLoadIdentity();

    wp1 = -0.04, wp2 = 0.745;
    for(int w = 0; w < 4; w++)
    {
        for(int wj = 0; wj < 7; wj++)
        {
            //  Small Window of Tall Building Two
            glLoadIdentity();
            glTranslatef(wp1, wp2, 0);
            glScalef(0.25, 0.2, 0);
            squareWindowComponent(R, G, B);
            glLoadIdentity();
            wp2 -= 0.045;
        }
        wp2 = 0.745;
        wp1 += 0.018;
    }

    glLoadIdentity();
    // Tall Building Three
    anyTriangle(0.228, 0.85, 0.2905, 0.85, 0.26, 0.9, 179, 119, 0);
    glTranslatef(0.5f, 0.1f, 0);
    glScalef(0.3, 1.08, 0);
    anyQuad(-0.7f, 0.7f, -0.7f, 0.3f, -0.9f, 0.3f, -0.9f, 0.7f, 179, 119, 0);
    glLoadIdentity();

    wp1 = 0.405, wp2 = 0.75;
    for(int w = 0; w < 3; w++)
    {
        for(int wj = 0; wj < 7; wj++)
        {
            //  Small Window of Tall Building Three
            glLoadIdentity();
            glTranslatef(wp1, wp2, 0);
            glScalef(0.23, 0.18, 0);
            squareWindowComponent(R, G, B);
            glLoadIdentity();
            wp2 -= 0.045;
        }
        wp2 = 0.75;
        wp1 += 0.018;
    }
    glLoadIdentity();

    //  Small Building Two
    glTranslatef(1.4f, 0.1f, 0);
    glScalef(0.6, 0.8, 0);
    anyQuad(-0.7f, 0.7f, -0.7f, 0.3f, -0.9f, 0.3f, -0.9f, 0.7f, 178, 190, 181);
    glLoadIdentity();

    wp1 = 1.275, wp2 = 0.4;

    //  => Small Buildings Small Window 2
    for(int wj = 0; wj < 3; wj++)
    {
        glLoadIdentity();
        glTranslatef(wp1, wp2, 0);
        glScalef(0.5, 0.5, 0);
        ovalWindowComponent(R, G, B);
        glLoadIdentity();
        wp2 -= 0.1;
    }
    wp1 = 1.382, wp2 = 0.53;
    for(int w = 0; w < 2; w++)
    {
        //  =>  Tall Building's Small Window 3
        for(int wj = 0; wj < 3; wj++)
        {
            glLoadIdentity();
            glTranslatef(wp1, wp2, 0);
            glScalef(0.7, 0.2, 0);
            squareWindowComponent(R, G, B);
            glLoadIdentity();
            wp2 -= 0.1;
        }
        wp2 = 0.53;
        wp1 += 0.068;
    }
}

    //  Side Trees
void treeView()
{
    glLoadIdentity();
    glTranslatef(0.2f, 0.2f, 0);
    treeComponent();
    glLoadIdentity();
    glLoadIdentity();
    glTranslatef(0.25f, 0.1f, 0);
    treeComponent();
    glLoadIdentity();

    glLoadIdentity();
    glTranslatef(0.15f, 0.1f, 0);
    treeComponent();
    glLoadIdentity();

    glLoadIdentity();
    glTranslatef(-1.65f, 0.3f, 0);
    treeComponent();
    glLoadIdentity();

    glLoadIdentity();
    glTranslatef(-1.59f, 0.3f, 0);
    treeComponent();
    glLoadIdentity();

    glLoadIdentity();
    glTranslatef(-1.7f, 0.2f, 0);
    treeComponent();
    glLoadIdentity();

    glLoadIdentity();
    glTranslatef(-1.62f, 0.2f, 0);
    treeComponent();
    glLoadIdentity();

    glLoadIdentity();
    glTranslatef(-1.65f, 0.1f, 0);
    treeComponent();
    glLoadIdentity();

    glLoadIdentity();
    glTranslatef(-1.55f, 0.1f, 0);
    treeComponent();
    glLoadIdentity();
}

    // Desert
void desertView()
{
    anyQuad(1.0f, 0.5f, 1.0f, 0.1f, -1.0f, 0.1f, -1.0f, 0.5f, 76, 70, 50);
}
```

```
    // Components for Night View
void NightSkyJoinedComponent()
{
    anyQuad(1.0f, 1.0f, 1.0f, -1.0f, -1.0f, -1.0f, -1.0f, 1.0f, 0, 0, 0);
    moonComponent();
    starComponent();
}

    // Sky Change Function
void skyView()
{
    (isDay) ? sunComponent() : NightSkyJoinedComponent();
}
    // Cloud View Function
void cloudView()
{
    glPushMatrix();
    glTranslatef(cloud_position,0,0);
    cloudComponent();

    glTranslatef(cloud_position + 0.8f,0,0);
    cloudComponent();

    glTranslatef(cloud_position - 0.4f,0,0);
    cloudComponent();

    glTranslatef(cloud_position + 1.5f,0,0);
    cloudComponent();

    glTranslatef(cloud_position - 1.2f,0,0);
    cloudComponent();

    glTranslatef(cloud_position - 0.9f,0,0);
    cloudComponent();
    glPopMatrix();
}

    //  Rain View Function
void rainView()
{
    float x=0.0, y = 1.5, x1=-0.099;
    glColor3ub(255,255,255);
    glPushMatrix();
    glTranslatef(0,rainPos,0);
    glBegin(GL_LINES);
```

```
     for(int i=600; i>=0; i--)
     {
         for(int j=0; j<=i; j++)
         {
             glVertex3f(x,y,0);
             glVertex3f(x+0.05,y+0.09,0);
             x+=float(rand()%5)/10;
         }
         for(int j=0; j<=i; j++)
         {
             glVertex3f(x1,y,0);
             glVertex3f(x1+0.05,y+0.09,0);
             x1-=float(rand()%5)/10;
         }
         y+=float(rand()%10)/10;
         x=0.0;
         x1=-0.099;
     }
     glEnd();
     glPopMatrix();
 }

     //  Instructions
 void instructions()
 {
     anyText("P = PAUSE , S = START , KEY UP = SPEED INCREASE , KEY DOWN = SPEED
DECREASE , KEY LEFT = CAR HORN , KEY RIGHT = HORN FOR SHIP", -0.95f, -0.90f,
255, 255, 255);
     anyText("D = CHANGE VIEW , A = AUTO ON \\ OFF , R = RAIN ON \\ OFF , I =
INFO HIDE \\ SHOW , F = FULL SCREEN \\ MINIMIZED , E = EXIT", -0.95f, -0.95f,
255, 255, 255);
 }
     // Full Screen Compatibility
 void fullView()
 {
     initState();
     skyView();
     (isDay) ? mountainView(15, 114, 22) : mountainView(0, 51, 17);
     (isRainy) ? cloudView() : blank();
     desertView();
     (isDay) ? buildingsView(0, 0, 0) : buildingsView(255,255,224);
     ahsanMonjilView();
     treeView();
     (isDay) ? roadView(255, 255, 255): roadView(255, 255, 0);
     (isDay) ? WaterView(2, 120, 191) : WaterView(0, 19, 77);
```

```
    (isHide) ? blank() : instructions();
    (isRainy) ? rainView() : blank();
    glDisable(GL_COLOR_MATERIAL | GL_LIGHTING);
    glFlush();
}

    //  Instruction Triggers
void handleKeypress(unsigned char key, int x, int y)
{
    switch (key)
    {
    case 'p':
        car_speed = 0.0f;
        ship_speed = 0.0f;
        break;
    case 's':
        car_speed = 0.079f;
        ship_speed = 0.005f;
        break;
    case 'd':
        isAuto = false;
        isDay = !isDay;
        break;
    case 'r':
        isAuto = false;
        isRainy = !isRainy;
        (isRainy) ? PlaySound("assets/rain.wav", NULL,
SND_FILENAME|SND_ASYNC|SND_LOOP) : PlaySound(NULL, 0, 0);
        break;
    case 'i':
        isHide = !isHide;
        break;
    case 'f':
        isFullScreen = !isFullScreen;
        (isFullScreen) ? glutFullScreen() : glutReshapeWindow(1400, 800);
        break;
    case 'a':
        isAuto = !isAuto;
        break;
    case 'e':
        exit(0);
        break;
        glutPostRedisplay();
    }
}
```

```
    // Mouse Compatibility
void handleMouse(int button, int state, int x, int y)
{
    if (button == GLUT_LEFT_BUTTON)
    {
    }
    if (button == GLUT_RIGHT_BUTTON)
    {
    }
    glutPostRedisplay();
}
    // Speed Up & Down
void SpecialInput(int key, int x, int y)
{
    switch(key)
    {
    case GLUT_KEY_UP:
        car_speed += 0.05;
        ship_speed += 0.005f;
        break;
    case GLUT_KEY_DOWN:
        if(car_speed > 0.05 || ship_speed > 0.005)
        {
            car_speed -= 0.05;
            ship_speed -= 0.005f;
        }
        else
        {
            car_speed = 0.0f;
            ship_speed = 0.0f;
        }
        break;
    case GLUT_KEY_LEFT:
        PlaySound("assets/car_double_horn.wav", NULL, SND_FILENAME|SND_ASYNC);
        break;
    case GLUT_KEY_RIGHT:
        PlaySound("assets/ship.wav", NULL, SND_FILENAME|SND_ASYNC);
        break;
    }
    glutPostRedisplay();
}


int main(int argc, char** argv)
{
    FreeConsole();
    glutInit(&argc, argv);
    glutInitWindowSize(1400, 800);
    glutCreateWindow("AHSAN MANZIL FRONT VIEW");
    glutDisplayFunc(fullView);
    glutIdleFunc(Idle);
    glutKeyboardFunc(handleKeypress);
    glutMouseFunc(handleMouse);
    glutSpecialFunc(SpecialInput);
    glutTimerFunc(1500, update, 0);
    glutMainLoop();
    return 0;
}
```

<u>Source code of component.h</u>

```
#include "shapes.h"

void blank()
{
}

void pillarComponent()
{

    anyQuad(-0.75f, 0.5f, -0.75f, 0.1f, -0.8f, 0.1f, -0.8f, 0.5f, 255, 140, 0);

    anyQuad(-0.755f, 0.55f, -0.755f, 0.5f, -0.795f, 0.5f, -0.795f, 0.55f, 255,
140, 0);

    Circle(-.775f, .55f, .02f, 255, 140, 0);

    glBegin(GL_LINES);
    glColor3ub(255, 140, 0);
    glVertex2f(-0.776f, 0.59f);
    glVertex2f(-0.776f, 0.56f);
    glVertex2f(-0.785f, 0.58f);
    glVertex2f(-0.77f, 0.58f);
    glEnd();
}

void gombujhComponent()
{

    anyQuad(-0.6f, 0.57f, -0.6f, 0.52f, -0.75f, 0.52f, -0.75f, 0.57f, 255, 140,
0);
```

```
    Circle(-.68f, .56f, .045f, 255, 140, 0);

    glBegin(GL_POINTS);
    glColor3ub(255, 140, 0);
    glVertex2f(-0.68f, 0.62f);
    glEnd();
}

void squareWindowComponent(int R, int G, int B)
{
    anyQuad(-0.69f, 0.4f, -0.69f, 0.3f, -0.73f, 0.3f, -0.73f, 0.4f, R, G, B);
}

void ovalWindowComponent(int R, int G, int B)
{
    anyQuad(-0.69f, 0.4f, -0.69f, 0.3f, -0.73f, 0.3f, -0.73f, 0.4f, R, G, B);
    Circle(-.71f, .4f, .021f, R, G, B);
}

void threeDotComponent()
{
    glBegin(GL_POINTS);
    glColor3ub(0, 0, 0);
    glVertex2f(-0.68f, 0.62f);
    glVertex2f(-0.66f, 0.62f);
    glVertex2f(-0.64f, 0.62f);
    glEnd();
}

void grassComponent()
{
    anyQuad(-1, 0.1, -1, -0.2, 1, -0.2, 1, 0.1, 10, 255, 100);
}

void carComponent()
{
    glBegin(GL_POLYGON);
    glColor3ub(255, 255, 51);
    glVertex3f(-0.3, -0.07, 0);
    glVertex3f(-0.15, -0.07, 0);
    glVertex3f(-0.15, -0.035, 0);
    glVertex3f(-0.18, -0.035, 0);
    glVertex3f(-0.20, -0.02, 0);
    glVertex3f(-0.25, -0.02, 0);
    glVertex3f(-0.27, -0.035, 0);
    glVertex3f(-0.3, -0.035, 0);
    glVertex3f(-0.27, -0.035, 0);
    glVertex3f(-0.3, -0.07, 0);
    glEnd();

    glBegin(GL_POLYGON);
    glColor3ub(0, 0, 0);
    glVertex3f(-0.255, -0.038, 0);
    glVertex3f(-0.235, -0.038, 0);
    glVertex3f(-0.241, -0.030, 0);
    glVertex3f(-0.248, -0.030, 0);
    glVertex3f(-0.255, -0.035, 0);
    glEnd();

    glBegin(GL_POLYGON);
    glColor3ub(0, 0, 0);
    glVertex3f(-0.221, -0.038, 0);
    glVertex3f(-0.20, -0.038, 0);
    glVertex3f(-0.209, -0.030, 0);
    glVertex3f(-0.216, -0.030, 0);
    glVertex3f(-0.221, -0.035, 0);
    glEnd();

    glLineWidth(1.0);
    glBegin(GL_LINES);
    glColor3ub(185, 0, 0);
    glVertex2f(-0.27, -.04);
    glVertex2f(-0.18, -.04);
    glEnd();

    Circle(-0.26f, -0.070f, 0.010f, 0, 0, 0);
    Circle(-0.199f, -0.07f, 0.01f, 0, 0, 0);
}

void car_2Component()
{

    anyQuad(0.025f, 0.001f, 0.025f, -0.045f, 0.222f, -0.045f, 0.222f, 0.001f,
255, 77, 255);

    glBegin(GL_POLYGON);
    glColor3ub(255, 77, 255);

    glVertex2f(0.065f, 0.031f);
    glVertex2f(0.055f, 0.001f);
    glVertex2f(0.20f, -0.001f);
```

```
     glVertex2f(0.185f, 0.031f);

     glEnd();
     glBegin(GL_POLYGON);
     glColor3ub(168, 204, 215);

     glVertex2f(0.065f, 0.027f);
     glVertex2f(0.058f, 0.001f);
     glVertex2f(0.198f, 0.001f);

     glVertex2f(0.185f, 0.027f);

     glEnd();

     glBegin(GL_POLYGON);
     glColor3ub(255, 77, 255);

     glVertex2f(0.222f, 0.001f);
     glVertex2f(0.222f, -0.045f);
     glVertex2f(0.252f, -0.045f);

     glVertex2f(0.252f, -0.033f);

     glEnd();

     glLineWidth(1.0);
     glBegin(GL_LINES);

     glColor3ub(255, 77, 255);
     glVertex2f(.12f, 0.031f);
     glVertex2f(.12f, -0.045f);
     glEnd();

     anyQuad(0.233f, -0.033f, 0.233f, -0.038f, 0.252f, -0.038f, 0.252f, -0.033f,
204, 204, 0);

     anyQuad(0.015f, -0.035f, 0.015f, -0.045f, 0.025f, -0.045f, 0.025f, -0.035f,
255, 77, 255);

     Circle(0.055f, -0.045f, 0.010f, 0, 0, 0);
     Circle(0.20f, -0.045f, 0.01f, 0, 0, 0);
 }

 void car_3Component()
 {
     anyQuad(0.04f, -0.14f, 0.04f, -0.19f, 0.19f, -0.19f, 0.19f, -0.14f, 0, 128,
0);

     glBegin(GL_POLYGON);
     glColor3ub(0, 128, 0);

     glVertex2f(0.078f, -0.11f);
     glVertex2f(0.055f, -0.14f);
     glVertex2f(0.16f, -0.14f);

     glVertex2f(0.13f, -0.11f);

     glEnd();

     glBegin(GL_POLYGON);
     glColor3ub(168, 204, 215);

     glVertex2f(0.078f, -0.115f);
     glVertex2f(0.059f, -0.14f);
     glVertex2f(0.11f, -0.14f);
     glVertex2f(0.11f, -0.115f);

     glEnd();

     glBegin(GL_POLYGON);
     glColor3ub(168, 204, 215);

     glVertex2f(0.113f, -0.115f);
     glVertex2f(0.113f, -0.14f);
     glVertex2f(0.155f, -0.14f);
     glVertex2f(0.13f, -0.115f);

     glEnd();

     glBegin(GL_POLYGON);
     glColor3ub(0, 128, 0);

     glVertex2f(0.04f, -0.14f);
     glVertex2f(0.015f, -0.17f);
     glVertex2f(0.015f, -0.19f);

     glVertex2f(0.04f, -0.19f);

     glEnd();
     anyQuad(0.015f, -0.17f, 0.015f, -0.18f, 0.029f, -0.18f, 0.029f, -0.17f,
204, 204, 0);
```

```
     anyQuad(0.19f, -0.18f, 0.19f, -0.19f, 0.196f, -0.19f, 0.196f, -0.18f, 0,
128, 0);
     Circle(0.068f, -0.19f, 0.010f, 0, 0, 0);
     Circle(0.15f, -0.19f, 0.01f, 0, 0, 0);
 }

 void car_4Component()
 {
     anyQuad(0.04f, -0.14f, 0.04f, -0.19f, 0.19f, -0.19f, 0.19f, -0.14f, 255, 0,
0);

     glBegin(GL_POLYGON);
     glColor3ub(255, 0, 0);
     glVertex2f(0.078f, -0.11f);
     glVertex2f(0.055f, -0.14f);
     glVertex2f(0.16f, -0.14f);
     glVertex2f(0.13f, -0.11f);
     glEnd();

     glBegin(GL_POLYGON);
     glColor3ub(168, 204, 215);
     glVertex2f(0.078f, -0.115f);
     glVertex2f(0.059f, -0.14f);
     glVertex2f(0.11f, -0.14f);
     glVertex2f(0.11f, -0.115f);
     glEnd();

     glBegin(GL_POLYGON);
     glColor3ub(168, 204, 215);
     glVertex2f(0.113f, -0.115f);
     glVertex2f(0.113f, -0.14f);
     glVertex2f(0.155f, -0.14f);
     glVertex2f(0.13f, -0.115f);
     glEnd();

     glBegin(GL_POLYGON);
     glColor3ub(255, 0, 0);
     glVertex2f(0.04f, -0.14f);
     glVertex2f(0.015f, -0.17f);
     glVertex2f(0.015f, -0.19f);
     glVertex2f(0.04f, -0.19f);
     glEnd();

     anyQuad(0.015f, -0.17f, 0.015f, -0.18f, 0.029f, -0.18f, 0.029f, -0.17f,
204, 204, 0);
```

```
     anyQuad(0.19f, -0.18f, 0.19f, -0.19f, 0.196f, -0.19f, 0.196f, -0.18f, 255,
0, 0);

     Circle(0.068f, -0.19f, 0.010f, 0, 0, 0);

     Circle(0.15f, -0.19f, 0.01f, 0, 0, 0);
 }

 void buildingComponent()
 {

     anyQuad(-0.7f, 0.7f, -0.7f, 0.3f, -0.9f, 0.3f, -0.9f, 0.7f, 255, 255, 255);

     glLoadIdentity();
     glTranslatef(-0.15f, 0.2f, 0);
     squareWindowComponent(202, 164, 114);
     glLoadIdentity();

     glLoadIdentity();
     glTranslatef(-0.09f, 0.2f, 0);
     ovalWindowComponent(202, 164, 114);
     glLoadIdentity();

     glLoadIdentity();
     glTranslatef(-0.03f, 0.2f, 0);
     squareWindowComponent(202, 164, 114);
     glLoadIdentity();

     glLoadIdentity();
     glTranslatef(-0.15f, 0.01f, 0);
     squareWindowComponent(202, 164, 114);
     glLoadIdentity();

     glLoadIdentity();
     glTranslatef(-0.09f, 0.01f, 0);
     ovalWindowComponent(202, 164, 114);
     glLoadIdentity();

     glLoadIdentity();
     glTranslatef(-0.03f, 0.01f, 0);
     squareWindowComponent(202, 164, 114);
     glLoadIdentity();
 }
 void shipComponent()
 {
     anyQuad(0.0f, -0.7f, 0.1f, -0.8f, 0.7f, -0.8f, 0.8f, -0.7f, 41, 34, 31);
     anyQuad(0.1f, -0.55f, 0.1f, -0.7f, 0.7f, -0.7f, 0.7f, -0.55f, 179, 179,
179);

     anyQuad(0.15f, -0.65f, 0.15f, -0.6f, 0.35f, -0.6f, 0.35f, -0.65f, 255, 255,
255);

     glBegin(GL_LINES);
     glColor3ub(31, 31, 31);
     glVertex2f(0.2f, -0.65f);
     glVertex2f(0.2f, -0.6f);
     glEnd();

     glBegin(GL_LINES);
     glColor3ub(31, 31, 31);
     glVertex2f(0.25f, -0.65f);
     glVertex2f(0.25, -0.6f);
     glEnd();

     glBegin(GL_LINES);
     glColor3ub(31, 31, 31);
     glVertex2f(0.3f, -0.65f);
     glVertex2f(0.3f, -0.6f);
     glEnd();

     anyQuad(0.4f, -0.65f, 0.4f, -0.6f, 0.55f, -0.6f, 0.55f, -0.65f, 255, 255,
255);

     glBegin(GL_LINES);
     glColor3ub(31, 31, 31);
     glVertex2f(0.45f, -0.65f);
     glVertex2f(0.45f, -0.6f);
     glEnd();

     glBegin(GL_LINES);
     glColor3ub(31, 31, 31);
     glVertex2f(0.5f, -0.65f);
     glVertex2f(0.5f, -0.6f);
     glEnd();

     anyQuad(0.6f, -0.65f, 0.6f, -0.6f, 0.65f, -0.6f, 0.65, -0.65f, 255, 255,
255);
     anyQuad(0.2f, -0.4f, 0.2f, -0.55f, 0.25f, -0.55f, 0.25f, -0.4f, 43, 40,
40);
     anyQuad(0.21f, -0.4f, 0.2f, -0.3f, 0.25f, -0.3f, 0.24f, -0.4f, 235, 70, 0);
     anyQuad(0.3f, -0.4f, 0.3f, -0.55f, 0.35f, -0.55f, 0.35f, -0.4f, 43, 40,
40);
     anyQuad(0.31f, -0.4f, 0.3f, -0.3f, 0.35f, -0.3f, 0.34f, -0.4f, 235, 70, 0);

     anyQuad(0.4f, -0.4f, 0.4f, -0.55f, 0.45f, -0.55f, 0.45f, -0.4f, 43, 40,
40);
     anyQuad(0.41f, -0.4f, 0.4f, -0.3f, 0.45f, -0.3f, 0.44f, -0.4f, 235, 70, 0);
     anyQuad(0.5f, -0.4f, 0.5f, -0.55f, 0.55f, -0.55f, 0.55f, -0.4f, 43, 40,
40);
     anyQuad(0.51f, -0.4f, 0.5f, -0.3f, 0.55f, -0.3f, 0.54f, -0.4f, 235, 70, 0);

     anyQuad(0.6f, -0.4f, 0.6f, -0.55f, 0.65f, -0.55f, 0.65f, -0.4f, 43, 40,
40);

     anyQuad(0.61f, -0.4f, 0.6f, -0.3f, 0.65f, -0.3f, 0.64f, -0.4f, 235, 70, 0);
 }

 void sunComponent()
 {
     Circle(0.8f, 0.8f, 0.09f, 255, 255, 0);
 }

 void moonComponent()
 {
     Circle(0.8f, 0.8f, 0.09f, 245, 245, 245);
 }
 void starComponent()
 {
     glPointSize(2);
     glBegin(GL_POINTS);
     glColor3ub(255, 255, 255);
     glVertex2f(-0.9, 0.9);
     glVertex2f(-0.8, 0.8);
     glVertex2f(-0.85, 0.95);
     glVertex2f(-0.75, 0.85);
     glVertex2f(-0.55, 0.85);
     glVertex2f(-0.25, 0.85);
     glVertex2f(-0.25, 0.75);
     glVertex2f(-0.0, 0.75);
     glVertex2f(0.25, 0.85);
     glVertex2f(0.275, 0.75);
     glVertex2f(0.0, 0.75);
     glVertex2f(0.9, 0.91);
     glVertex2f(0.875, 0.95);
     glVertex2f(0, 0.99);
     glVertex2f(-0.1, 0.95);
```

```
    glVertex2f(-0.2, 0.93);
    glVertex2f(-0.4, 0.95);
    glVertex2f(0.1, 0.95);
    glVertex2f(0.2, 0.93);
    glVertex2f(0.4, 0.95);
    glVertex2f(0.7, 0.95);
    glVertex2f(0.8, 0.95);
    glEnd();
    glPointSize(7);
}

void cloudComponent()
{
    Circle(-0.58f, 0.75f, 0.060f, 70, 75, 71); //left cloud
    Circle(-0.5f, 0.82f, 0.068f, 70, 75, 71);
    Circle(-0.42f, 0.75f, 0.068f, 70, 75, 71); //right cloud
    Circle(-0.5f, 0.72f, 0.07f, 70, 75, 71);
}
void treeComponent()
{
    anyQuad(0.69f, 0.13f, 0.67f, 0.10f, 0.78f, 0.10f, 0.76f, 0.13f, 0, 102,
51);
    anyQuad(0.70f, 0.16f, 0.68f, 0.13f, 0.77f, 0.13f, 0.75f, 0.16f, 0, 153, 0);
    anyQuad(0.71f, 0.19f, 0.69f, 0.16f, 0.76f, 0.16f, 0.74f, 0.19f, 0, 204, 0);
    anyQuad(0.72f, 0.22f, 0.70f, 0.19f, 0.75f, 0.19f, 0.73f, 0.22f, 128, 255,
0);
    anyQuad(0.71f, 0.10f, 0.71f, 0.0f, 0.73f, 0.0f, 0.73f, 0.10f, 102, 51, 0);
    anyQuad(0.73f, 0.10f, 0.73f, 0.0f, 0.74f, 0.0f, 0.74f, 0.10f, 153, 76, 0);
}

void lampComponent(int R, int G, int B)
{
    anyQuad(-0.7f, 0.3f, -0.7f, 0.0f, -0.72f, 0.0f, -0.72f, 0.3f, 0, 0, 0);
    anyQuad(-0.7f, 0.3f, -0.7f, 0.28f, -0.65f, 0.28f, -0.65f, 0.3f, 0, 0, 0);
    anyQuad(-0.68f, 0.28f, -0.68f, 0.25f, -0.65f, 0.25f, -0.65f, 0.28f, R, G,
B);
}
void credit()
{
    anyText("Credit: Section: Computer Graphics[I] | Group: H", 0.2f, -0.17f,
103, 72, 70);
}
```

Source code of “shapes.h”

```
 #include <iostream>
 #include <GL/glut.h>
 #include <cmath>
 #define PI  3.14159265358979323846
 using namespace std;

 int i;
 int triangleAmount = 500;
 GLfloat x = 0,y = 0,radius = 0;
 GLfloat twicePi = 2.0f * PI;

 void Circle(GLfloat cx, GLfloat cy, GLfloat radius, int r, int g, int b)
 {
     glColor3ub(r,g,b);
      glBegin(GL_TRIANGLE_FAN);
            glVertex2f(cx, cy);
            for(int i = 0; i <= triangleAmount;i++) {
                  glVertex2f(
                        cx + (radius * cos(i *  twicePi / triangleAmount)),
                      cy + (radius * sin(i * twicePi / triangleAmount))
                  );
            }
      glEnd();
 }
 void anyTriangle( float a, float b, float c, float d, float e, float f, int R,
int G, int B ){
     glBegin(GL_TRIANGLES);
     glColor3ub(R, G, B);
     glVertex2f(a, b);
     glVertex2f(c, d);
     glVertex2f(e, f);
     glEnd();
 }

 void anyQuad( float a, float b, float c, float d, float e, float f, float g,
float h, int R, int G, int B ){
     glBegin(GL_QUADS);
     glColor3ub(R, G, B);
     glVertex2f(a, b);
     glVertex2f(c, d);
     glVertex2f(e, f);
     glVertex2f(g, h);
     glEnd();
 }
```

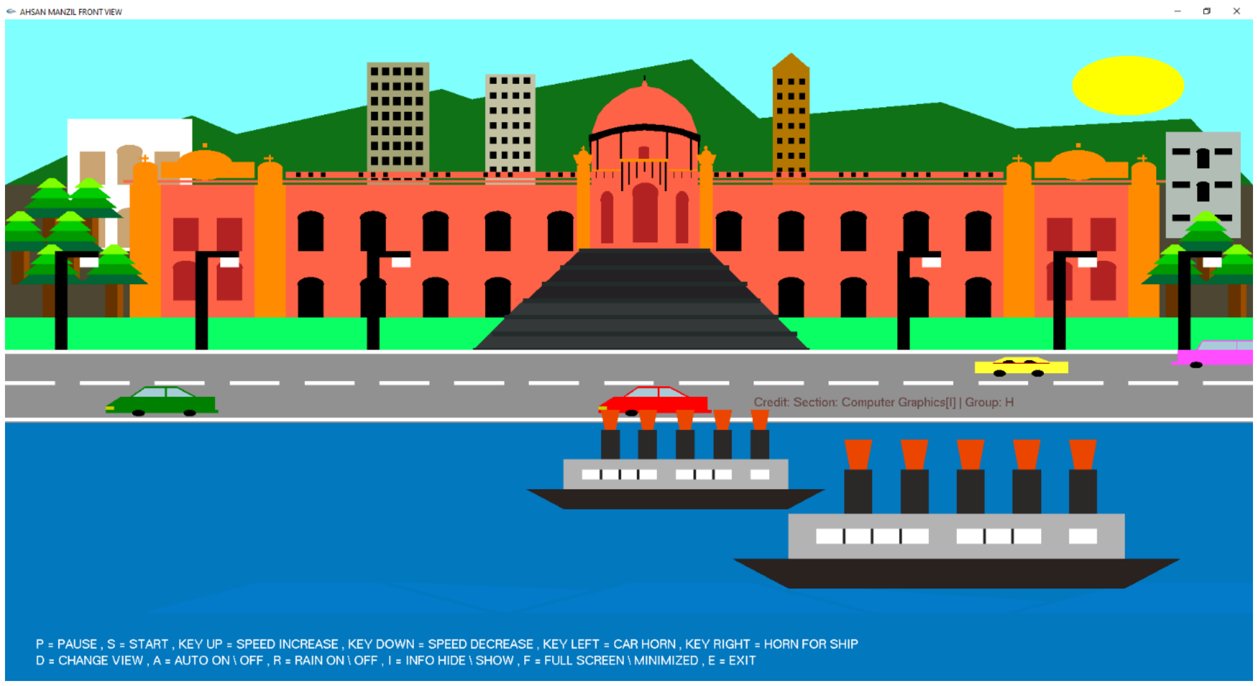
```
void lineComponent(float a, float b, float c, float d){
   glBegin(GL_LINES);
   glVertex2f(a, b);
   glVertex2f(c, d);
   glEnd();
}

void anyText(string text, float x, float y, int R, int G, int B){
    glColor3ub(R, G, B);
    glRasterPos2f(x, y);
    for (i = 0; i < text.length(); i++) {
        glutBitmapCharacter(GLUT_BITMAP_HELVETICA_18, text[i]);
    }
}
```

5. Significance

Ahsan Manzil is one of the most significant architectural monuments of Bangladesh. The building was established on a raised platform of 1 meter; the two-storied palace measures 125.4m by 28.75m. The height of the ground floor is 5 meters, and the size of the first floor is 5.8 meters.

In computer graphics, scenery refers to the combined application of transformation, projection, and animation of the computer screen using OpenGL. This will be a first-person view of AHSAN MANZIL. Here, a portion of AHSAN MANZIL middle is focused. In front of the scene, there will be a river and a moving boat in a river. There will be Clouds in the sky, and they are moving. In front of AHSAN MANZIL, there is a road, and some vehicles are moving. Mountain and some buildings have been shown on the backside of Ahsan Manzil. There are two small trees beside the road. In the night scene, the sky will be dimmed. The represented view is from west to east. All of the things mentioned above made the scenery beautiful and impressive.

6. User Interface

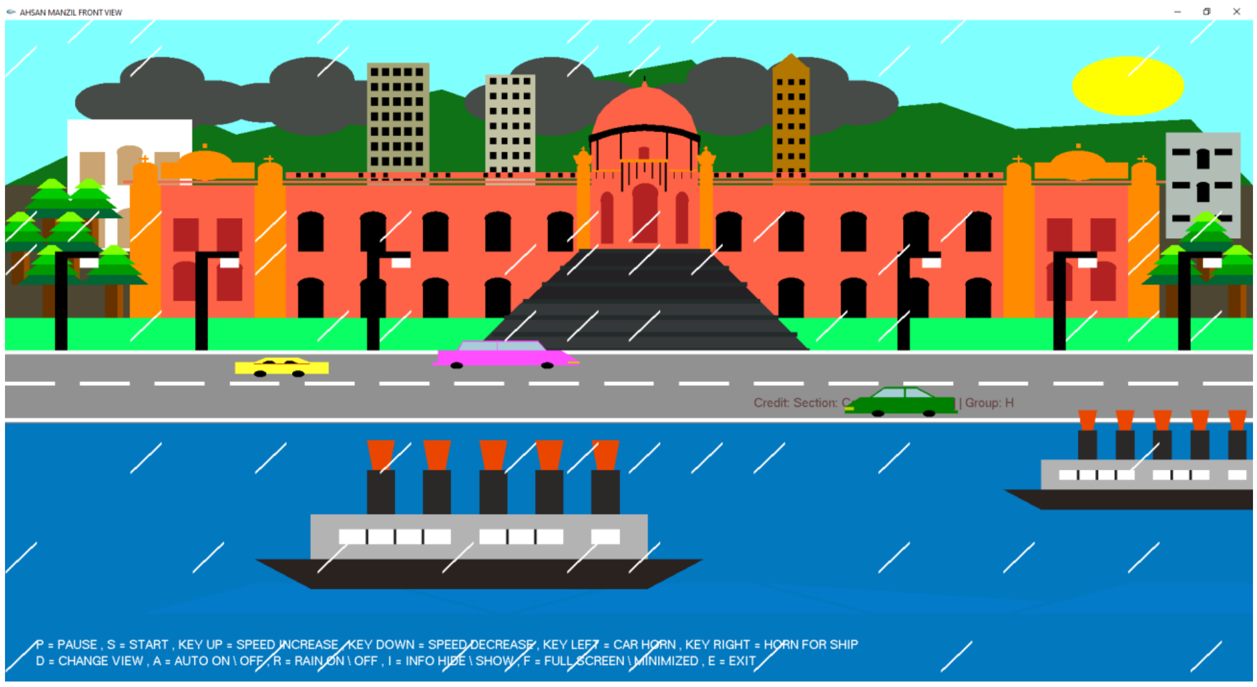


AHSAN MANZIL FRONT VIEW

Credit: Section: Computer Graphics[I] | Group: H

P = PAUSE , S = START , KEY UP = SPEED INCREASE , KEY DOWN = SPEED DECREASE , KEY LEFT = CAR HORN , KEY RIGHT = HORN FOR SHIP
D = CHANGE VIEW , A = AUTO ON \ OFF , R = RAIN ON \ OFF , I = INFO HIDE \ SHOW , F = FULL SCREEN \ MINIMIZED , E = EXIT

Figure 1: Day View of Ahsan Manzil

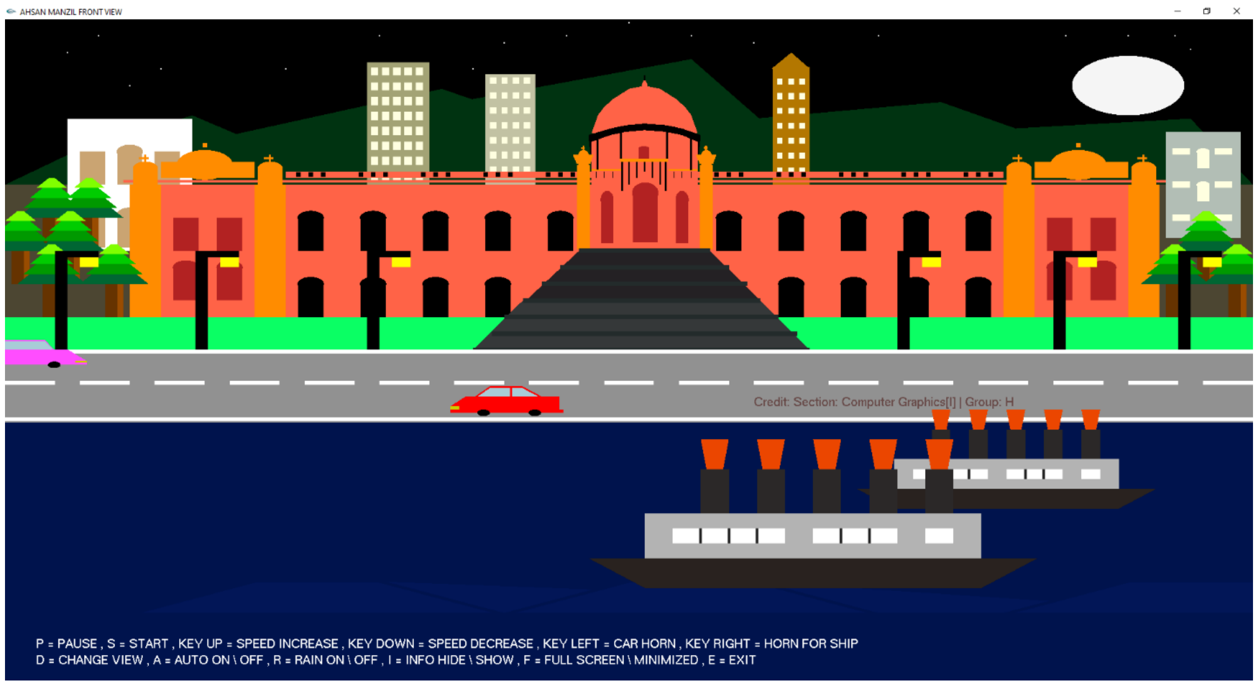


AHSAN MANZIL FRONT VIEW

Credit: Section: C | Group: H

P = PAUSE , S = START , KEY UP = SPEED INCREASE , KEY DOWN = SPEED DECREASE , KEY LEFT = CAR HORN , KEY RIGHT = HORN FOR SHIP
D = CHANGE VIEW , A = AUTO ON \ OFF , R = RAIN ON \ OFF , I = INFO HIDE \ SHOW , F = FULL SCREEN \ MINIMIZED , E = EXIT

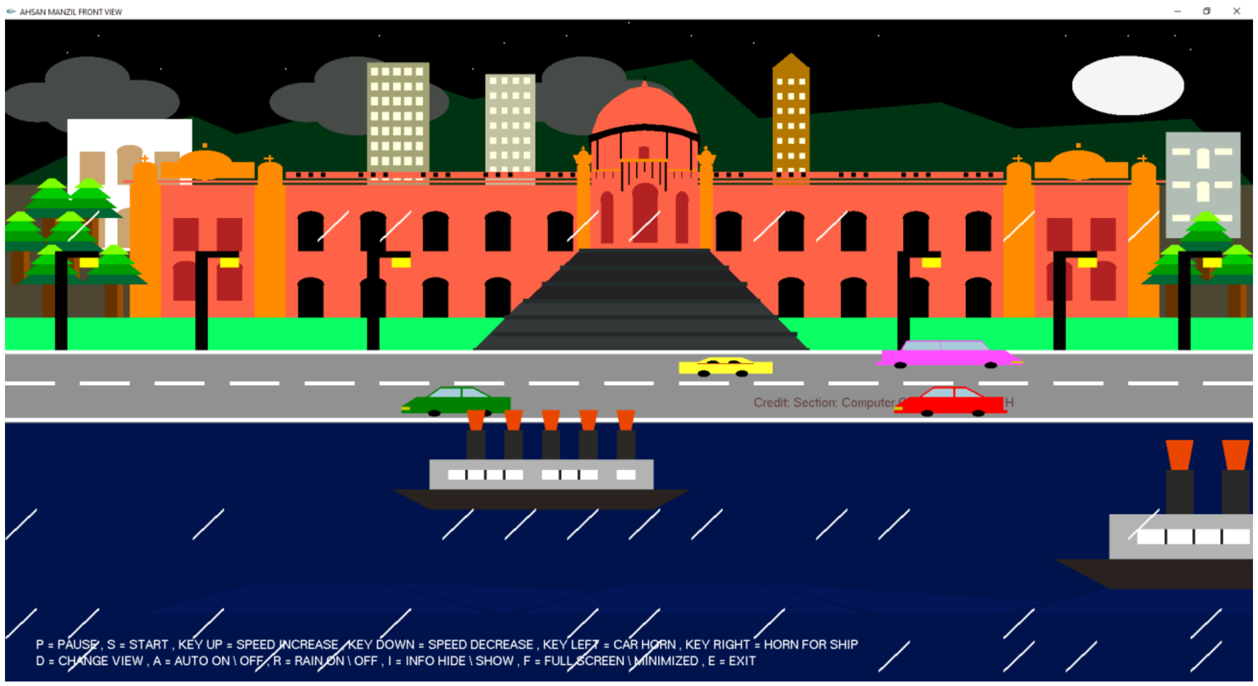
Figure 2: Day View of Ahsan Manzil with Raining



Credit: Section: Computer Graphics[I] | Group: H

P = PAUSE , S = START , KEY UP = SPEED INCREASE , KEY DOWN = SPEED DECREASE , KEY LEFT = CAR HORN , KEY RIGHT = HORN FOR SHIP
D = CHANGE VIEW , A = AUTO ON \ OFF , R = RAIN ON \ OFF , I = INFO HIDE \ SHOW , F = FULL SCREEN \ MINIMIZED , E = EXIT

Figure 3: Night View of Ahsan Manzil



Credit: Section: Computer H

P = PAUSE , S = START , KEY UP = SPEED INCREASE , KEY DOWN = SPEED DECREASE , KEY LEFT = CAR HORN , KEY RIGHT = HORN FOR SHIP
D = CHANGE VIEW , A = AUTO ON \ OFF , R = RAIN ON \ OFF , I = INFO HIDE \ SHOW , F = FULL SCREEN \ MINIMIZED , E = EXIT

Figure 4: Night View of Ahsan Manzil with Raining

7. Conclusion

The project is well suited for designing 2D and 3D objects and carrying out basic graphics functionalities like drawing a simple line, creating a cube, circle. Square. Erasing and filling them, However. If implemented on a large scale with sufficient resources. It has the potential to become a standard stand-alone GUI-based application for Windows Operating System.

Out of the many available features, the project demonstrates some popular and commonly used OpenGL features such as Rendering Transformation, Rotation, Lighting, Scaling, etc. These graphic functions will also work in tandem with various rules involved in the scenario. Since this project works on dynamic values, it can be used for real-time computation.

The project enabled to work with mid-level OpenGL complexity, and the project demonstrates the scope of the OpenGL platform as a premier 2D graphics development launchpad. Hence. It has indeed helped develop many 2D Graphics projects. OpenGL, in its own right, is suitable for low-cost and straightforward 2D graphics development. It serves as an important stepping stone for venturing into other fields of Computer Graphics design and applications.

8. References

[1] https://github.com/sahajunior/Ahsan-Manzil-Museum

[2] https://www.javatpoint.com/computer-graphics-tutorial

[3] https://www.academia.edu/32477177/Ahsan_Manzil.docx