Practical – 2

AIM: Study to run 2D & Animated OpenGL Program in Visual Studio.

Code:

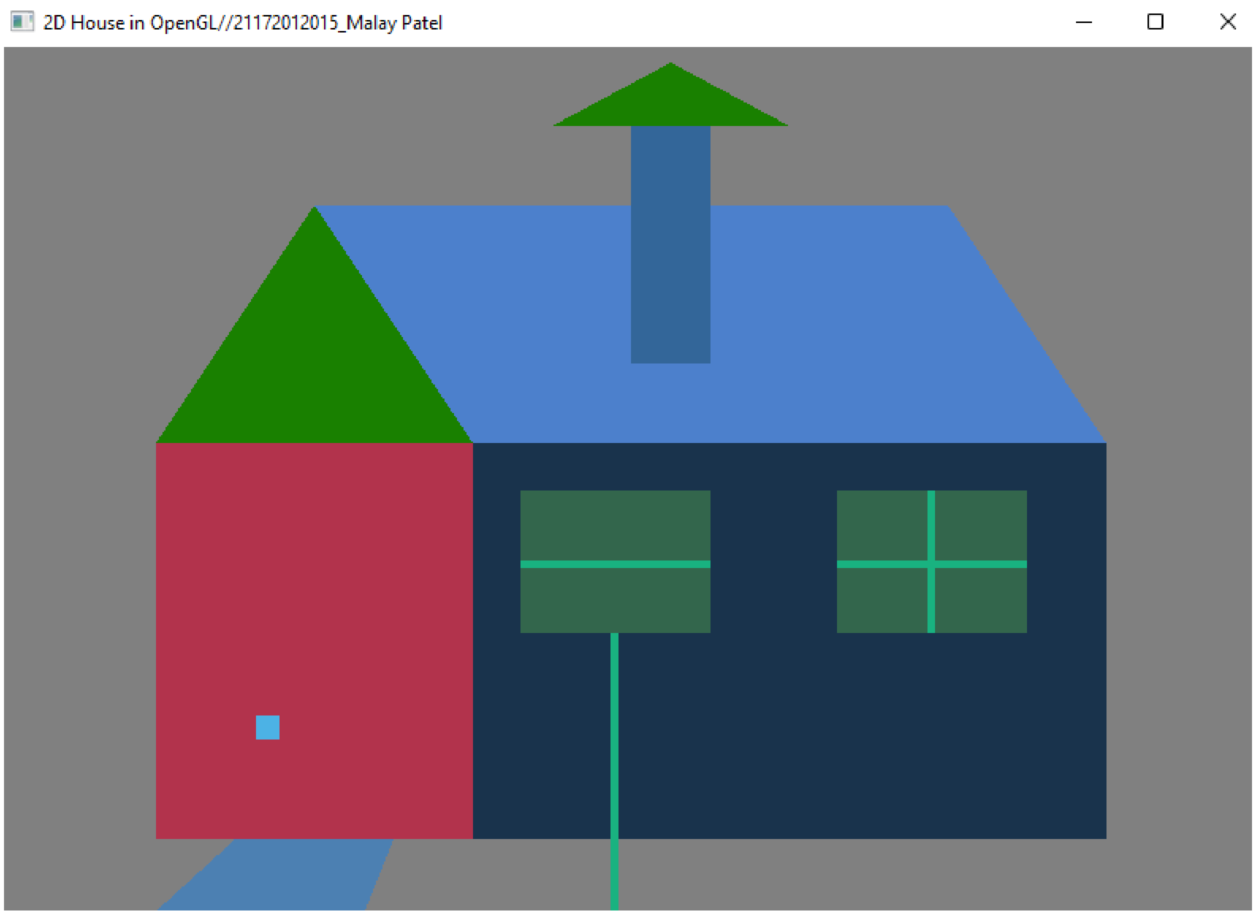
1)

```
namespace gp21 {
	void init()

	{
		// Set display window color to as glClearColor(R,G,B,Alpha)
		glClearColor(0.5, 0.5, 0.5, 0.5);
		// Set projection parameters.
		glMatrixMode(GL_PROJECTION);
		// Set 2D Transformation as gluOrtho2D(Min Width, Max Width, Min
Height,MaxHeight)
		gluOrtho2D(0.0, 800, 0.0, 600);
	}
	void home()
	{
		//Roof
		glClear(GL_COLOR_BUFFER_BIT); // Clear display window
		// Set line segment color as glColor3f(R,G,B)
		glColor3f(0.3, 0.5, 0.8);
		glBegin(GL_POLYGON);
		glVertex2i(200, 500);
		glVertex2i(600, 500);
		glVertex2i(700, 350);
		glVertex2i(300, 350);
			glEnd();
		// Top of Front Wall
		glColor3f(0.1, 0.5, 0.0);
		glBegin(GL_TRIANGLES);
		glVertex2i(200, 500);
		glVertex2i(100, 350);
		glVertex2i(300, 350);
		glEnd();
		// Front Wall
		glColor3f(0.7, 0.2, 0.3);
		glBegin(GL_POLYGON);
		glVertex2i(100, 350);
		glVertex2i(300, 350);
		glVertex2i(300, 100);
		glVertex2i(100, 100);
```

```
glEnd();
// Front Door
glColor3f(0.7, 0.2, 0.9);
glBegin(GL_POLYGON);
glVertex2i(150, 250);
glVertex2i(250, 250);
glEnd();// Front Door Lock
glColor3f(0.3, 0.7, 0.9);
glPointSize(15);
glBegin(GL_POINTS);
glVertex2i(170, 170);
glEnd();
//side Wall
glColor3f(0.1, 0.2, 0.3);
glBegin(GL_POLYGON);
glVertex2i(300, 350);
glVertex2i(700, 350);
glVertex2i(700, 100);
glVertex2i(300, 100);
glEnd();
// window one
glColor3f(0.2, 0.4, 0.3);
glBegin(GL_POLYGON);
glVertex2i(330, 320);
glVertex2i(450, 320);
glVertex2i(450, 230);
glVertex2i(330, 230);
glEnd();
// line of window one
glColor3f(0.1, 0.7, 0.5);
glLineWidth(5);
glBegin(GL_LINES);
glVertex2i(390, 3);
glVertex2i(390, 230);
glVertex2i(330, 273);
glVertex2i(450, 273);
glEnd();
// window two
glColor3f(0.2, 0.4, 0.3);
glBegin(GL_POLYGON);
glVertex2i(530, 320);
glVertex2i(650, 320);
glVertex2i(650, 230);
glVertex2i(530, 230);
glEnd();
// lines of window two
glColor3f(0.1, 0.7, 0.5);
glLineWidth(5);
glBegin(GL_LINES);
glVertex2i(590, 320);
glVertex2i(590, 230);
glVertex2i(530, 273);
glVertex2i(650, 273);
glEnd();
// Entrance Path
glColor3f(0.3, 0.5, 0.7);
glLineWidth(3);
glBegin(GL_POLYGON);
```

```
		glVertex2i(150, 100);
		glVertex2i(250, 100);
		glVertex2i(210, 0);
		glVertex2i(40, 0);
		glEnd();
		glColor3f(0.2, 0.4, 0.6);
		glBegin(GL_POLYGON);
		glVertex2i(400, 550);
		glVertex2i(450, 550);
		glVertex2i(450, 400);
		glVertex2i(400, 400);
		glEnd();
		glColor3f(0.1, 0.5, 0.0);
		glBegin(GL_TRIANGLES);
		glVertex2i(425, 590);
		glVertex2i(500, 550);
		glVertex2i(350, 550);
		glEnd();
		// Process all OpenGL routines as quickly as possible
		glFlush();
	}
	void main(int argc, char** argv)

	{
		glutInit(&argc, argv);
		// Initialize GLUglutInit(&argc, argv);
// Set display mode
		glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
		// Set top - left display window position.
		glutInitWindowPosition(100, 100);
		// Set display window width and height
		glutInitWindowSize(800, 600);
		// Create display window with the given title
		glutCreateWindow("2D House in OpenGL//21172012015_Malay Patel ");
		// Execute initialization procedure
		init();
		// Send graphics to display window
		glutDisplayFunc(home);
		// Display everything and wait.
		glutMainLoop();
	}

}


namespace gp21 {
	void main(int argc, char** argv);
```

OUTPUT:



Code:

2)

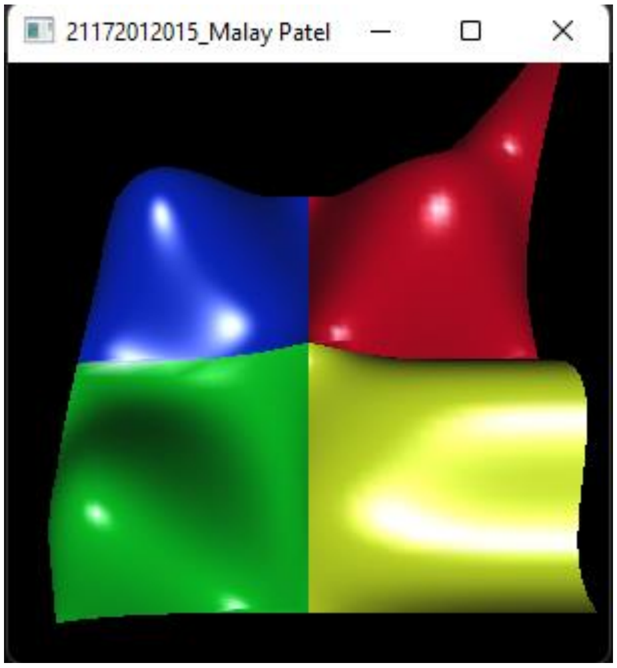
```
namespace gp22 {
	GLfloat mat_red_diffuse[] = { 0.7, 0.0, 0.1, 1.0 };
	GLfloat mat_green_diffuse[] = { 0.0, 0.7, 0.1, 1.0 };
	GLfloat mat_blue_diffuse[] = { 0.0, 0.1, 0.7, 1.0 };
	GLfloat mat_yellow_diffuse[] = { 0.7, 0.8, 0.1, 1.0 };
	GLfloat mat_specular[] = { 1.0, 1.0, 1.0, 1.0 };
	GLfloat mat_shininess[] = { 100.0 };
	GLfloat knots[8] = { 0.0, 0.0, 0.0, 0.0, 1.0, 1.0, 1.0, 1.0 };
	GLfloat pts1[4][4][3], pts2[4][4][3];
	GLfloat pts3[4][4][3], pts4[4][4][3];
	GLUnurbsObj* nurb;
```

```
int u, v;
static void display(void)

{
	glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
	glCallList(1);
	glFlush();
}
void main(int argc, char** argv)
{
	glutInit(&argc, argv);
	glutCreateWindow("21172012015_Malay Patel");
	glMaterialfv(GL_FRONT, GL_SPECULAR, mat_specular);
	glMaterialfv(GL_FRONT, GL_SHININESS, mat_shininess);
	glEnable(GL_LIGHTING);
	glEnable(GL_LIGHT0);
	glEnable(GL_DEPTH_TEST);
	glEnable(GL_AUTO_NORMAL);
	glEnable(GL_NORMALIZE);
	nurb = gluNewNurbsRenderer();
	gluNurbsProperty(nurb, GLU_SAMPLING_TOLERANCE, 25.0);
	gluNurbsProperty(nurb, GLU_DISPLAY_MODE, GLU_FILL);
	/* Build control points for NURBS mole hills. */
	for (u = 0; u < 4; u++) {
		for (v = 0; v < 4; v++) {
			/* Red. */
			pts1[u][v][0] = 2.0 * ((GLfloat)u);
			pts1[u][v][1] = 2.0 * ((GLfloat)v);
			if ((u == 1 || u == 2) && (v == 1 || v == 2))
				/* Stretch up middle. */

				pts1[u][v][2] = 6.0;
			else
				pts1[u][v][2] = 0.0;
			/* Green. */
			pts2[u][v][0] = 2.0 * ((GLfloat)u - 3.0);
			pts2[u][v][1] = 2.0 * ((GLfloat)v - 3.0);
			if ((u == 1 || u == 2) && (v == 1 || v == 2))
				if (u == 1 && v == 1)
					/* Pull hard on single middle square. */
					pts2[u][v][2] = 15.0;
				else
					/* Push down on other middle squares. */
					pts2[u][v][2] = -2.0;
			else
				pts2[u][v][2] = 0.0;
			/* Blue. */
			pts3[u][v][0] = 2.0 * ((GLfloat)u - 3.0);
			pts3[u][v][1] = 2.0 * ((GLfloat)v);
			if ((u == 1 || u == 2) && (v == 1 || v == 2))
				if (u == 1 && v == 2)
					/* Pull up on single middple square. */
					pts3[u][v][2] = 11.0;
				else
					/* Pull up slightly on other middle squares*/
					pts3[u][v][2] = 2.0;
			else
				pts3[u][v][2] = 0.0;
```

```
                        /* Yellow. */

                        pts4[u][v][0] = 2.0 * ((GLfloat)u);
                    pts4[u][v][1] = 2.0 * ((GLfloat)v - 3.0);
                    if ((u == 1 || u == 2 || u == 3) && (v == 1 || v == 2))
                        if (v == 1)
                            /* Push down front middle and right squares. */
                            pts4[u][v][2] = -2.0;
                        else
                            /* Pull up back middle and right squares. */
                            pts4[u][v][2] = 5.0;
                    else
                        pts4[u][v][2] = 0.0;
                    }
                }
                /* Stretch up red's far right corner. */
                pts1[3][3][2] = 6;
                /* Pull down green's near left corner a little. */
                pts2[0][0][2] = -2;
                /* Turn up meeting of four corners. */
                pts1[0][0][2] = 1;
                pts2[3][3][2] = 1;
                pts3[3][0][2] = 1;
                pts4[0][3][2] = 1;
                glMatrixMode(GL_PROJECTION);
                gluPerspective(55.0, 1.0, 2.0, 24.0);
                glMatrixMode(GL_MODELVIEW);
                glTranslatef(0.0, 0.0, -15.0);
                glRotatef(330.0, 1.0, 0.0, 0.0);

                glNewList(1, GL_COMPILE);
            /* Render red hill. */
            glMaterialfv(GL_FRONT, GL_DIFFUSE, mat_red_diffuse);
            gluBeginSurface(nurb);
            gluNurbsSurface(nurb, 8, knots, 8, knots,
                4 * 3, 3, &pts1[0][0][0],
                4, 4, GL_MAP2_VERTEX_3);
            gluEndSurface(nurb);
            /* Render green hill. */
            glMaterialfv(GL_FRONT, GL_DIFFUSE, mat_green_diffuse);
            gluBeginSurface(nurb);
            gluNurbsSurface(nurb, 8, knots, 8, knots,
                4 * 3, 3, &pts2[0][0][0],
                4, 4, GL_MAP2_VERTEX_3);
            gluEndSurface(nurb);
            /* Render blue hill. */
            glMaterialfv(GL_FRONT, GL_DIFFUSE, mat_blue_diffuse);
            gluBeginSurface(nurb);
            gluNurbsSurface(nurb, 8, knots, 8, knots,
                4 * 3, 3, &pts3[0][0][0],
                4, 4, GL_MAP2_VERTEX_3);
            gluEndSurface(nurb);
            /* Render yellow hill. */
            glMaterialfv(GL_FRONT, GL_DIFFUSE, mat_yellow_diffuse);
            gluBeginSurface(nurb);
            gluNurbsSurface(nurb, 8, knots, 8, knots,
                4 * 3, 3, &pts4[0][0][0],
                4, 4, GL_MAP2_VERTEX_3);
```

```
gluEndSurface(nurb);
glEndList();
glutDisplayFunc(display);
glutMainLoop();
//return 0; /* ANSI C requires main to return int. */
        }
}



        namespace gp22 {
                void main(int argc, char** argv);
        }
```

OUTPUT:


21172012015_Malay Patel