EXPERIMENT NO. 1

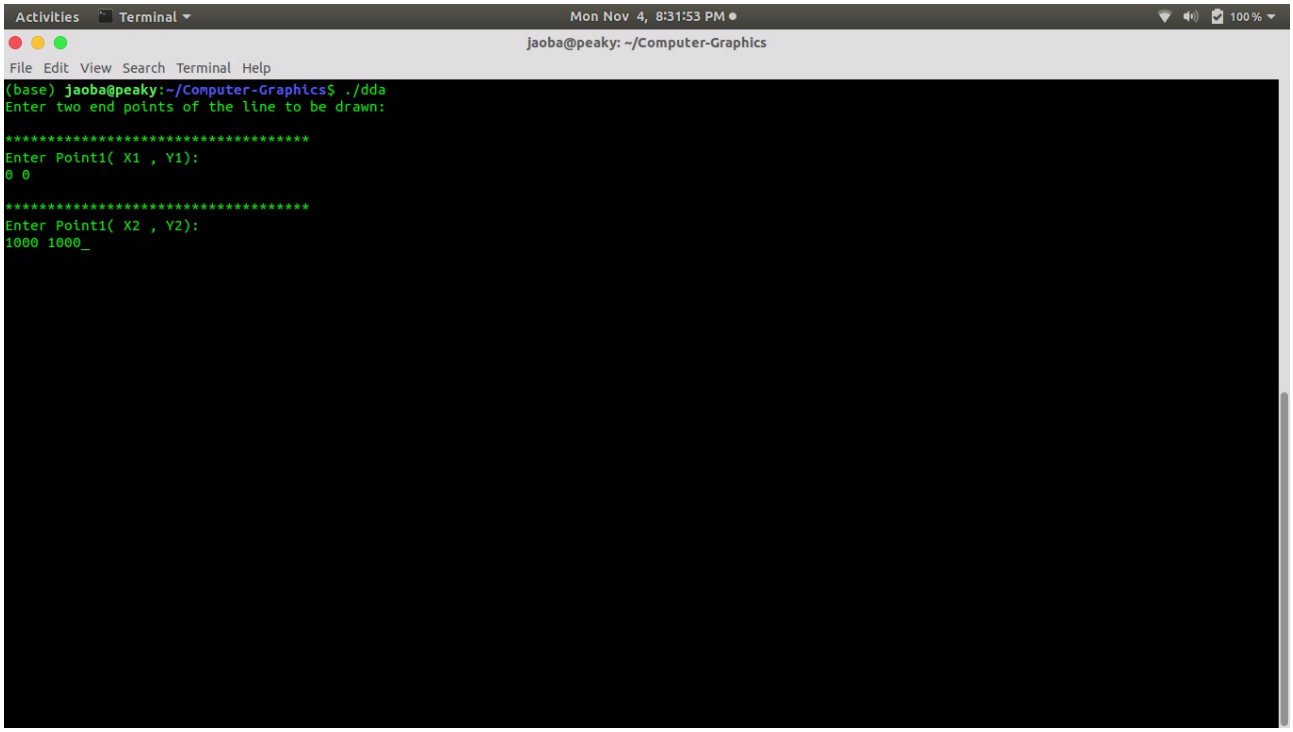
Aim: Program to draw a line using DDA Algorithm.

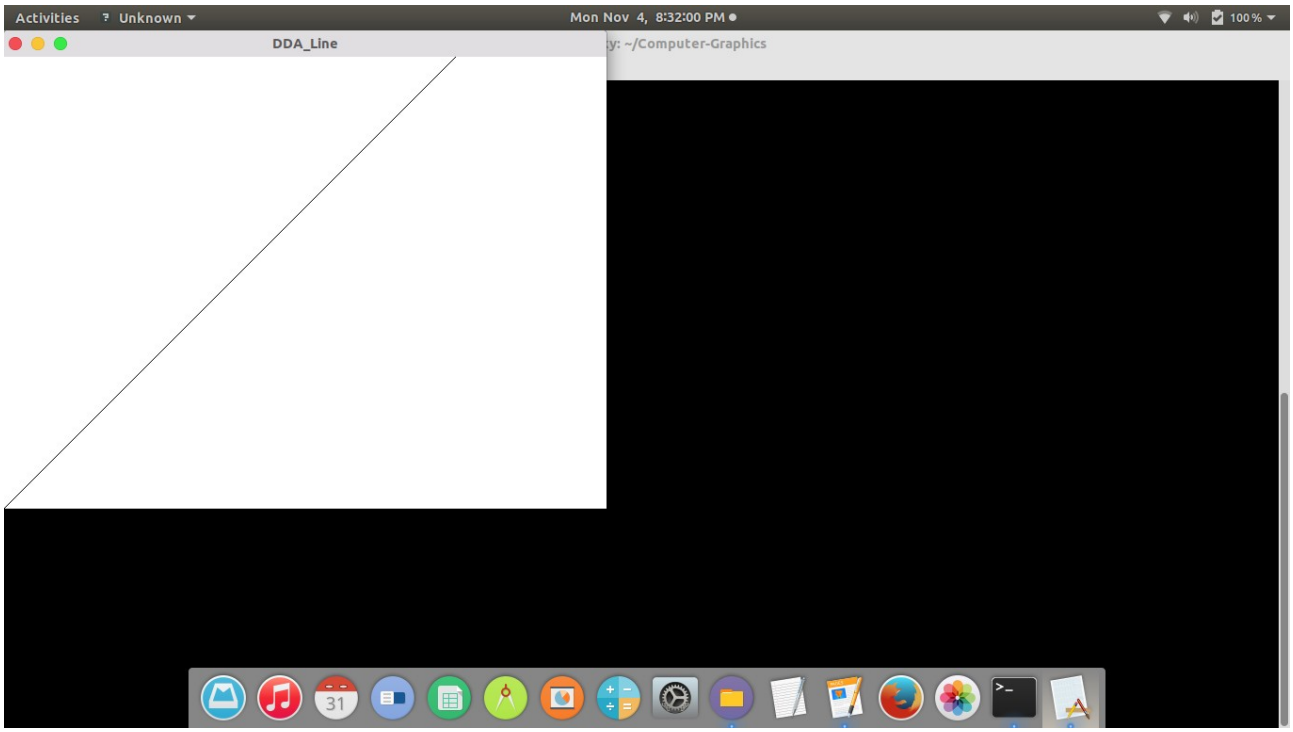
Code:

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

double X1, Y1, X2, Y2;

float round_value(float v)
{
 return floor(v + 0.5);
}
void LineDDA(void)
{
 double dx=(X2-X1);
 double dy=(Y2-Y1);
 double steps;
 float xInc,yInc,x=X1,y=Y1;
 /* Find out whether to increment x or y */
 steps=(abs(dx)>abs(dy))?(abs(dx)):(abs(dy));
 xInc=dx/(float)steps;
 yInc=dy/(float)steps;

 /* Clears buffers to preset values */
 glClear(GL_COLOR_BUFFER_BIT);

 /* Plot the points */
 glBegin(GL_POINTS);
 /* Plot the first point */
 glVertex2d(x,y);
 int k;
 /* For every step, find an intermediate vertex */
 for(k=0;k<steps;k++)
 {
  x+=xInc;
  y+=yInc;
  /* printf("%0.6lf %0.6lf\n",floor(x), floor(y)); */
  glVertex2d(round_value(x), round_value(y));
 }
 glEnd();
```

```
  glFlush();
}
void Init()
{
  /* Set clear color to white */
  glClearColor(1.0,1.0,1.0,0);
  /* Set fill color to black */
  glColor3f(0.0,0.0,0.0);
  /* glViewport(0 , 0 , 640 , 480); */
  /* glMatrixMode(GL_PROJECTION); */
  /* glLoadIdentity(); */
  gluOrtho2D(0 , 640 , 0 , 480);
}
int main(int argc, char **argv)
{
  printf("Enter two end points of the line to be drawn:\n");
  printf("\n*************************************");
  printf("\nEnter Point1( X1 , Y1):\n");
  scanf("%lf%lf",&X1,&Y1);
  printf("\n*************************************");
  printf("\nEnter Point1( X2 , Y2):\n");
  scanf("%lf%lf",&X2,&Y2);

  /* Initialise GLUT library */
  glutInit(&argc,argv);
  /* Set the initial display mode */
  glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
  /* Set the initial window position and size */
  glutInitWindowPosition(0,0);
  glutInitWindowSize(640,480);
  /* Create the window with title "DDA_Line" */
  glutCreateWindow("DDA_Line");
  /* Initialize drawing colors */
  Init();
  /* Call the displaying function */
  glutDisplayFunc(LineDDA);
  /* Keep displaying untill the program is closed */
  glutMainLoop();
}
```

Output:


Activities
Terminal
Mon Nov 4, 8:31:53 PM
100 %
jaoba@peaky: ~/Computer-Graphics
File Edit View Search Terminal Help
(base) jaoba@peaky:~/Computer-Graphics$ ./dda
Enter two end points of the line to be drawn:
****
Enter Point1(X1 , Y1):
0 0
****
Enter Point1(X2 , Y2):
1000 1000_


Activities
Unknown
Mon Nov 4, 8:32:00 PM
100 %
DDA_Line

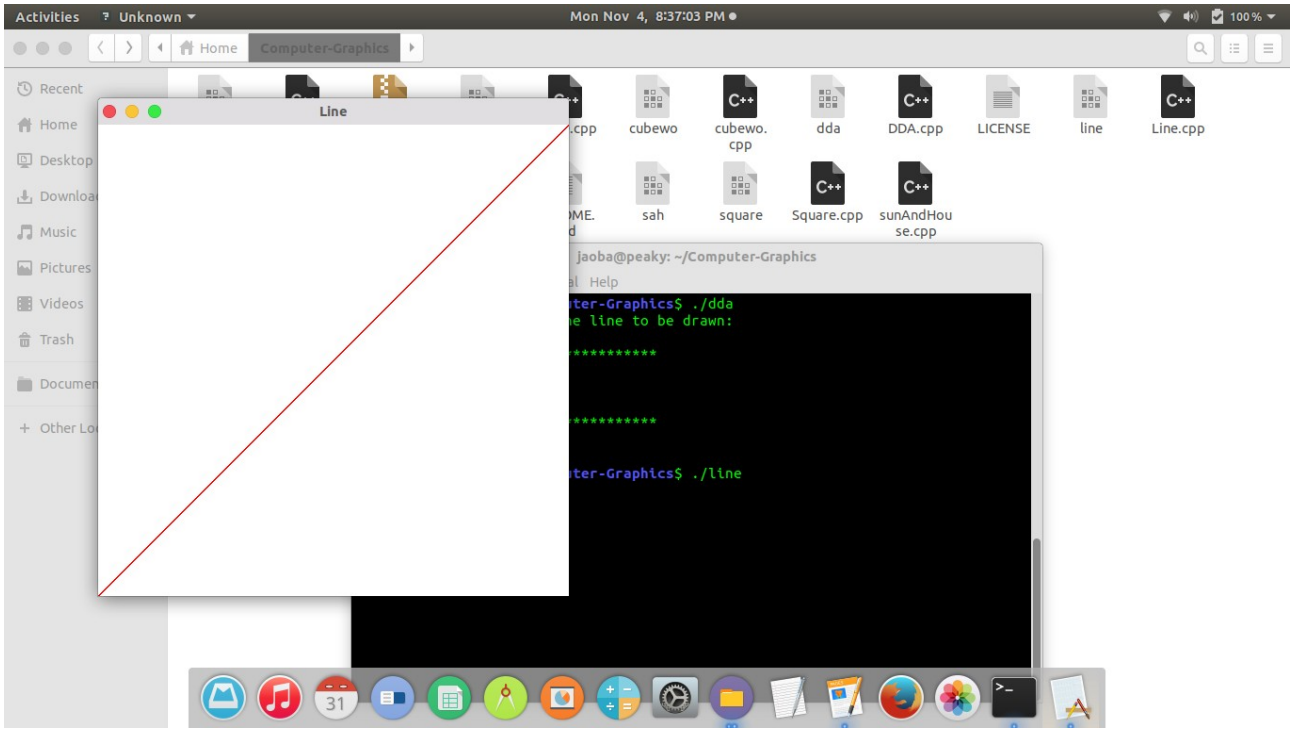
EXPERIMENT NO. 2

Aim: Program to draw a line using OpenGL.

Code:

```
#include <GL/gl.h>
#include <GL/glut.h>
void display(void)
{
glClear (GL_COLOR_BUFFER_BIT);


glLineWidth(2.5);
glColor3f (1.0, 0.0, 0.0);
glBegin(GL_LINES);
glVertex3f(0.0,0.0,0.0);
glVertex3f(1,1,0.0);
glEnd();
glFlush ();
}
void init (void)
{
glClearColor (1.0, 1.0, 1.0, 1.0);
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
glOrtho(0.0, 1.0, 0.0, 1.0, -1.0, 1.0);
}
int main(int argc, char** argv)
{
glutInit(&argc, argv);
glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB);
glutInitWindowSize (500, 500);
glutInitWindowPosition (100, 100);
glutCreateWindow ("Line");
init ();
glutDisplayFunc(display);
glutMainLoop();
return 0;
}
```

Output:


Activities
Unknown
Mon Nov 4, 8:37:03 PM
100 %
Home
Computer-Graphics
Recent
Home
Desktop
Music
Pictures
Videos
Trash
Line
cubewo
cubewo.cpp
dda
DDA.cpp
LICENSE
line
Line.cpp
sah
square
Square.cpp
sunAndHouse.cpp
jaoba@peaky: ~/Computer-Graphics
Help
ter-Graphics$ ./dda
e line to be drawn:
****
****
ter-Graphics$ ./line

EXPERIMENT NO. 3

Aim: Program to draw a line and show translation, rotation and scaling motion of the line using OpenGL.

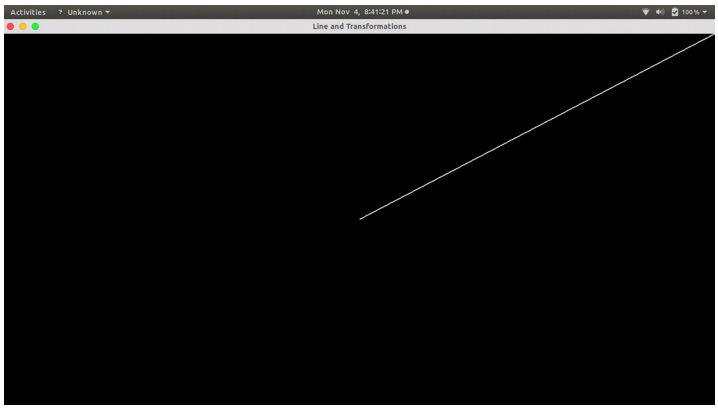
Code:

```
#include <stdio.h>
#include <stdarg.h>
#include <math.h>
#define GL_GLEXT_PROTOTYPES
#include <GL/glut.h>


// -------------------------------------------------------
// Function Prototypes
// -------------------------------------------------------
void display();
void specialKeys();

// -------------------------------------------------------
// Global Variables
// -------------------------------------------------------
double rotate_y = 0.0;
double rotate_x = 0.0;
double scale = 2.5;
double translate = 0.0;

// -------------------------------------------------------
// display() Callback function
// -------------------------------------------------------
void display(){

  //  Clear screen and Z-buffer
  glClear(GL_COLOR_BUFFER_BIT|GL_DEPTH_BUFFER_BIT);

  // Reset transformations
  glLoadIdentity();

  // Translation
  glTranslatef( translate, 0.0, 0.0 );
  // Rotate when user changes rotate_x and rotate_y
  glRotatef( rotate_x, 1.0, 0.0, 0.0 );
  glRotatef( rotate_y, 0.0, 1.0, 0.0 );
```

```
  // Scaling
  glScaled( scale, scale, 0.0 );

  //Line
  glLineWidth(2.5);
  glBegin(GL_LINES);
  glVertex3f(0.0,0.0,0.0);
  glVertex3f(1,1,0.0);
  glEnd();

  glFlush();
  glutSwapBuffers();

}

// ---------------------------------------------------------
// specialKeys() Callback Function
// ---------------------------------------------------------
void specialKeys( int key, int x, int y ) {

  //  Right arrow - increase rotation by 5 degree
  if (key == GLUT_KEY_RIGHT)
    rotate_y += 5.0;

  //  Left arrow - decrease rotation by 5 degree
  else if (key == GLUT_KEY_LEFT)
    rotate_y -= 5.0;

  else if (key == GLUT_KEY_UP)
    rotate_x += 5.0;

  else if (key == GLUT_KEY_DOWN)
    rotate_x -= 5.0;
  else if(key == GLUT_KEY_F1){
          scale += 0.1;
  }
  else if(key == GLUT_KEY_F2){
       scale -= 0.1;
  }
  else if(key == GLUT_KEY_F3){
          translate +=0.1;
  }
  else if(key == GLUT_KEY_F4){
```

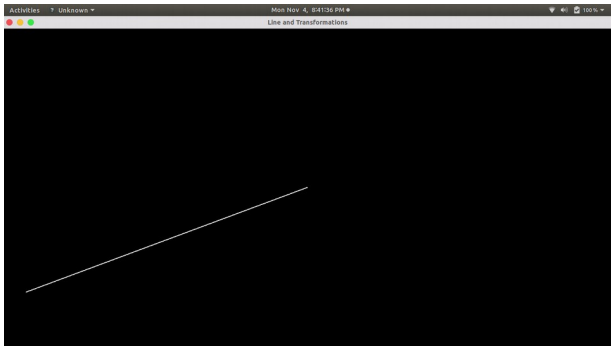
```
        translate -=0.1;
  }


  //  Request display update
  glutPostRedisplay();

}

// -------------------------------------------------------
// main() function
// -------------------------------------------------------
int main(int argc, char* argv[]){

  //  Initialize GLUT and process user parameters
  glutInit(&argc,argv);

  //  Request double buffered true color window with Z-buffer
  glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH);

  // Create window
  glutCreateWindow("Line and Transformations");

  //  Enable Z-buffer depth test
  glEnable(GL_DEPTH_TEST);


  // Callback functions
  glutDisplayFunc(display);
  glutSpecialFunc(specialKeys);


  //  Pass control to GLUT for events
  glutMainLoop();

  //  Return to OS
  return 0;

}
```

Output:

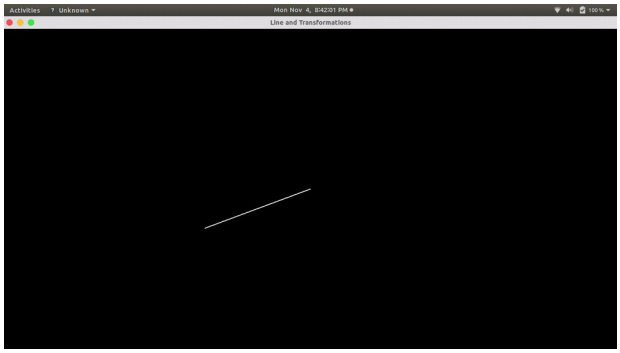


(i) Without Transformations.
(ii) Rotation
(iii) Scaling
(iv) Translation

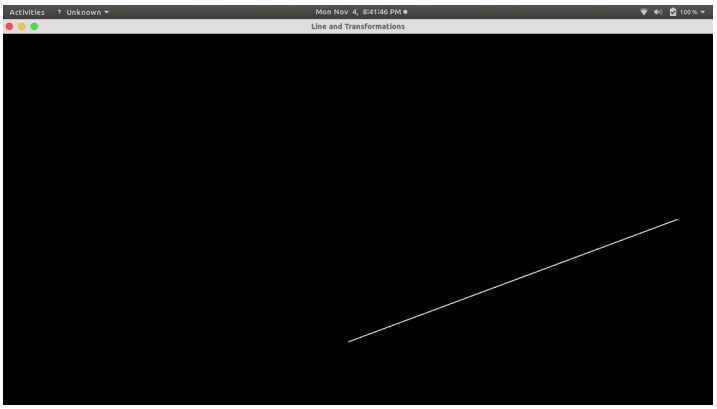
(i)



(ii)



(iii)



(iv)

EXPERIMENT NO. 4

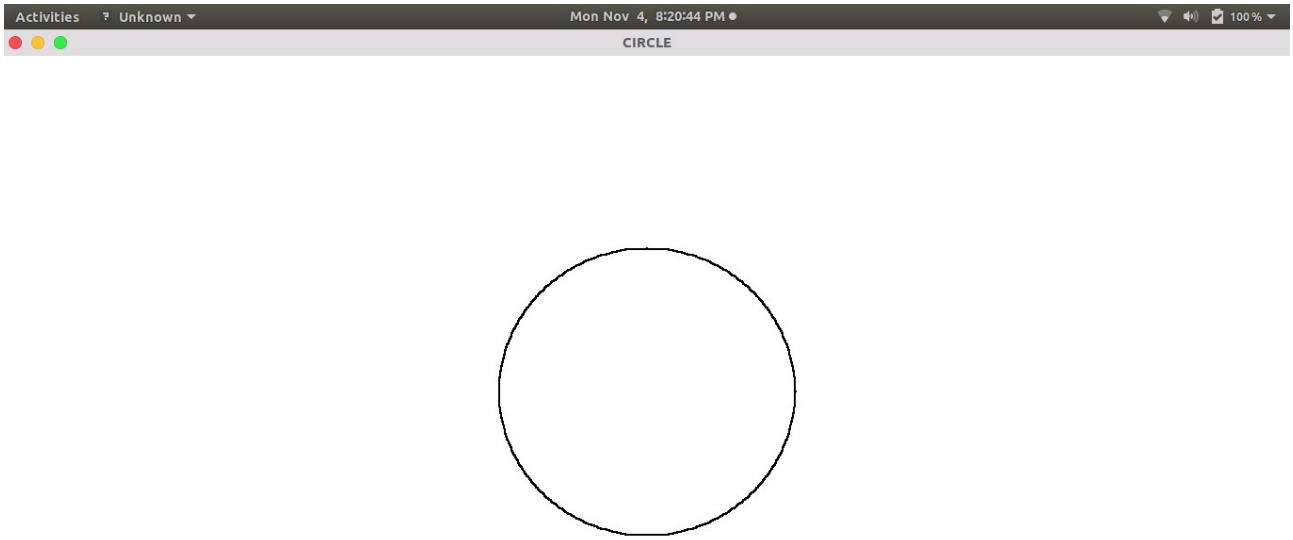
Aim: To draw a circle using OpenGL.

Code:

```
#include<stdio.h>
#include<GL/glut.h>
#include<math.h>
#define pi 3.142857

// function to initialize
void myInit (void)
{
	// making background color black as first
	// 3 arguments all are 0.0
	glClearColor(1.0, 1.0, 1.0, 1.0);

	// making picture color green (in RGB mode), as middle argument is 1.0
	glColor3f(0.0, 0.0, 0.0);

	// breadth of picture boundary is 1 pixel
	glPointSize(1.8);
	glMatrixMode(GL_PROJECTION);
	glLoadIdentity();

	// setting window dimension in X- and Y- direction
	gluOrtho2D(-780, 780, -420, 420);
}

void display (void)
{
	glClear(GL_COLOR_BUFFER_BIT);
	glBegin(GL_POINTS);
	float x, y, i;

	// iterate y up to 2*pi, i.e., 360 degree
	// with small increment in angle as
	// glVertex2i just draws a point on specified co-ordinate
	for ( i = 0; i < (2 * pi); i += 0.001)
	{
		// radius = 180
		// x=r*cos(i) and y=r*sin(i)
		x = 180 * cos(i);
```

```
                y = 180 * sin(i);

                glVertex2i(x, y);
        }
        glEnd();
        glFlush();
}

int main (int argc, char** argv)
{
        glutInit(&argc, argv);
        glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);

        // giving window size in X- and Y- direction
        glutInitWindowSize(1366, 768);
        glutInitWindowPosition(0, 0);

        // Giving name to window
        glutCreateWindow("CIRCLE");
        myInit();

        glutDisplayFunc(display);
        glutMainLoop();
}
```

Output:



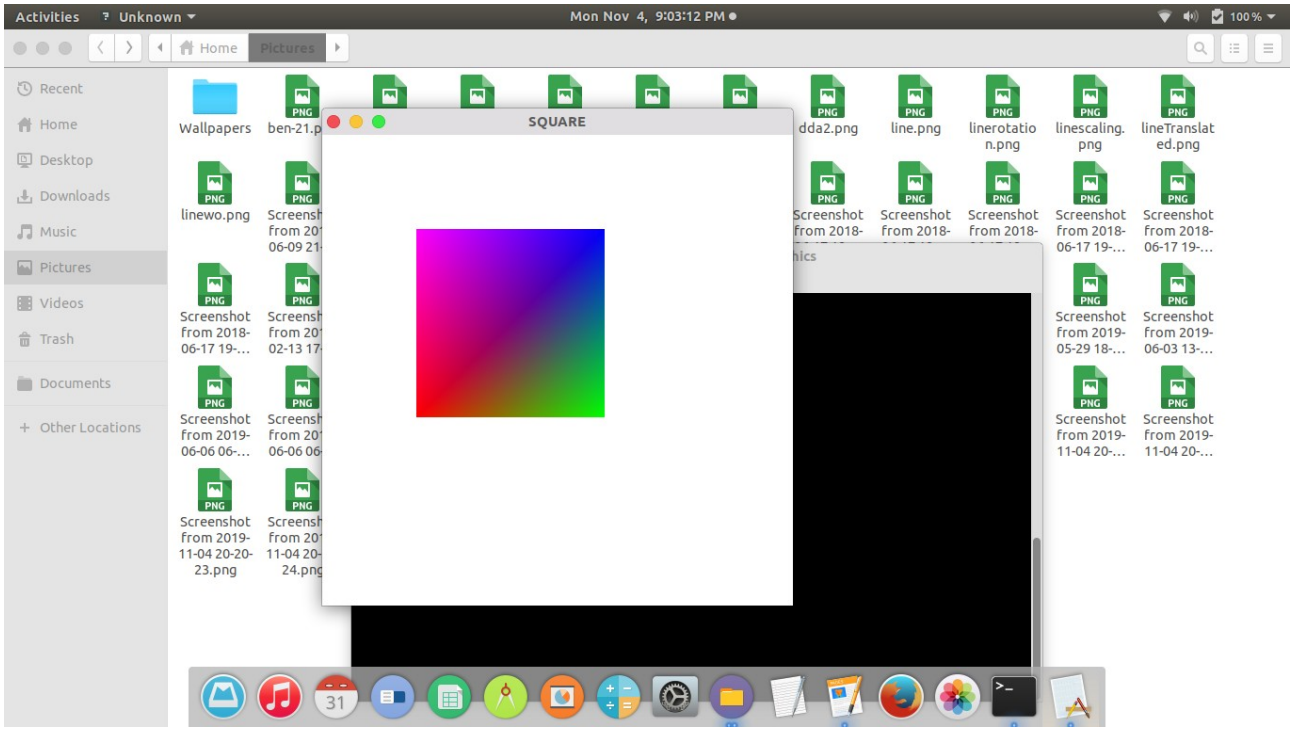
EXPERIMENT NO. 5

Aim: Program to draw a square using OpenGL.

Code:

```
#include <GL/gl.h>
#include <GL/glut.h>
void display(void)
{
glClear (GL_COLOR_BUFFER_BIT);
glBegin(GL_POLYGON);

  glColor3f( 1.0, 0.0, 0.0 );    glVertex3f(  2.0, 4.0, 0.0 );    // P1 is red
  glColor3f( 0.0, 1.0, 0.0 );    glVertex3f(  6.0, 4.0, 0.0 );    // P2 is green
  glColor3f( 0.0, 0.0, 1.0 );    glVertex3f(  6.0, 8.0, 0.0 );    // P3 is blue
  glColor3f( 1.0, 0.0, 1.0 );    glVertex3f(  2.0, 8.0, 0.0 );    // P4 is purple

glEnd();
glFlush ();
}
void init (void)
{
glClearColor (1.0, 1.0, 1.0, 1.0);
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
glOrtho(0.0, 10.0, 0.0, 10.0, -1.0, 1.0);
}
int main(int argc, char** argv)
{
glutInit(&argc, argv);
glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB | GLUT_DEPTH);
glutInitWindowSize (500, 500);
glutInitWindowPosition (100, 100);
glutCreateWindow ("SQUARE");
init ();
glutDisplayFunc(display);
glutMainLoop();
return 0;
}
```

Output:


Activities
Unknown
Mon Nov 4, 9:03:12 PM
100 %
Home
Pictures
Recent
Home
Desktop
Downloads
Music
Pictures
Videos
Trash
Documents
Other Locations
Wallpapers
ben-21.p
linewo.png
Screenshot from 2018-06-17 19-...
Screenshot from 2019-06-06 06-...
Screenshot from 2019-11-04 20-20-23.png
Screenshot from 2019-11-04 20-24.png
dda2.png
line.png
linerotation.png
linescaling.png
lineTranslated.png
Screenshot from 2018-06-17 19-...
Screenshot from 2019-05-29 18-...
Screenshot from 2019-06-03 13-...
Screenshot from 2019-11-04 20-...
SQUARE

EXPERIMENT NO. 6

Aim: Program to draw a cube and show translation, rotation and scaling motion using openGL.

Code:

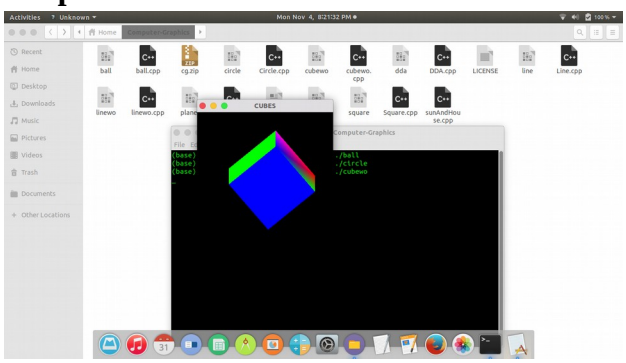
```
#include <stdio.h>
#include <stdarg.h>
#include <math.h>
#define GL_GLEXT_PROTOTYPES
#include <GL/glut.h>

// -------------------------------------------------------
// Function Prototypes
// -------------------------------------------------------
void display();
void specialKeys();

// -------------------------------------------------------
// Global Variables
// -------------------------------------------------------
double rotate_y=0;
double rotate_x=0;
double scale = 0.0;
double translate = 0.0;
// -------------------------------------------------------
// display() Callback function
// -------------------------------------------------------
void display(){

  //  Clear screen and Z-buffer
  glClear(GL_COLOR_BUFFER_BIT|GL_DEPTH_BUFFER_BIT);

  // Reset transformations
  glLoadIdentity();

  // Translations
  glTranslatef( translate, 0.0, 0.0 );

  // Rotate when user changes rotate_x and rotate_y
  glRotatef( rotate_x, 1.0, 0.0, 0.0 );
  glRotatef( rotate_y, 0.0, 1.0, 0.0 );

  // Scaling
```

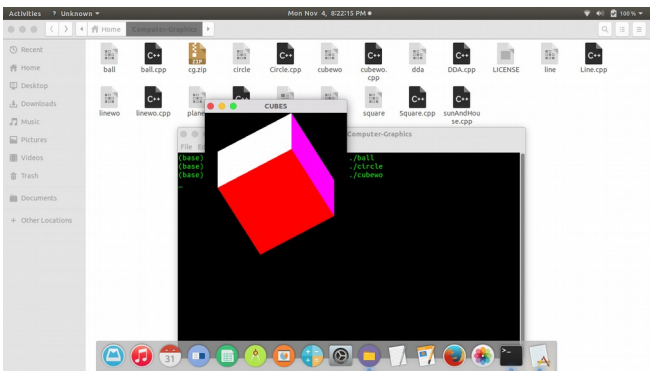
```
glScalef( scale, scale, scale );

//Multi-colored side - FRONT
glBegin(GL_POLYGON);

glColor3f( 1.0, 0.0, 0.0 );     glVertex3f(  0.5, -0.5, -0.5 );      // P1 is red
glColor3f( 0.0, 1.0, 0.0 );     glVertex3f(  0.5,  0.5, -0.5 );      // P2 is green
glColor3f( 0.0, 0.0, 1.0 );     glVertex3f( -0.5,  0.5, -0.5 );      // P3 is blue
glColor3f( 1.0, 0.0, 1.0 );     glVertex3f( -0.5, -0.5, -0.5 );      // P4 is purple

glEnd();

// White side - BACK
glBegin(GL_POLYGON);
glColor3f(   1.0,  1.0, 1.0 );
glVertex3f(  0.5, -0.5, 0.5 );
glVertex3f(  0.5,  0.5, 0.5 );
glVertex3f( -0.5,  0.5, 0.5 );
glVertex3f( -0.5, -0.5, 0.5 );
glEnd();

// Purple side - RIGHT
glBegin(GL_POLYGON);
glColor3f(  1.0,  0.0,  1.0 );
glVertex3f( 0.5, -0.5, -0.5 );
glVertex3f( 0.5,  0.5, -0.5 );
glVertex3f( 0.5,  0.5,  0.5 );
glVertex3f( 0.5, -0.5,  0.5 );
glEnd();

// Green side - LEFT
glBegin(GL_POLYGON);
glColor3f(   0.0,  1.0,  0.0 );
glVertex3f( -0.5, -0.5,  0.5 );
glVertex3f( -0.5,  0.5,  0.5 );
glVertex3f( -0.5,  0.5, -0.5 );
glVertex3f( -0.5, -0.5, -0.5 );
glEnd();

// Blue side - TOP
glBegin(GL_POLYGON);
glColor3f(   0.0,  0.0,  1.0 );
glVertex3f(  0.5,  0.5,  0.5 );
glVertex3f(  0.5,  0.5, -0.5 );
```

```
  glVertex3f( -0.5,  0.5, -0.5 );
  glVertex3f( -0.5,  0.5,  0.5 );
  glEnd();

  // Red side - BOTTOM
  glBegin(GL_POLYGON);
  glColor3f(   1.0,  0.0,  0.0 );
  glVertex3f(  0.5, -0.5, -0.5 );
  glVertex3f(  0.5, -0.5,  0.5 );
  glVertex3f( -0.5, -0.5,  0.5 );
  glVertex3f( -0.5, -0.5, -0.5 );
  glEnd();

  glFlush();
  glutSwapBuffers();

}

// ---------------------------------------------------------
// specialKeys() Callback Function
// ---------------------------------------------------------
void specialKeys( int key, int x, int y ) {

  //  Right arrow - increase rotation by 5 degree
  if (key == GLUT_KEY_RIGHT)
    rotate_y += 5;

  //  Left arrow - decrease rotation by 5 degree
  else if (key == GLUT_KEY_LEFT)
    rotate_y -= 5;

  else if (key == GLUT_KEY_UP)
    rotate_x += 5;

  else if (key == GLUT_KEY_DOWN)
    rotate_x -= 5;
  else if(key == GLUT_KEY_F1){
    scale += 0.1;
  }
  else if(key == GLUT_KEY_F2){
    scale -= 0.1;
  }
  else if(key == GLUT_KEY_F3){
    translate += 0.1;
```

```
  }
  else if(key == GLUT_KEY_F4){
    translate -= 0.1;
  }

  //  Request display update
  glutPostRedisplay();

}

// -------------------------------------------------------
// main() function
// -------------------------------------------------------
int main(int argc, char* argv[]){

  //  Initialize GLUT and process user parameters
  glutInit(&argc,argv);

  //  Request double buffered true color window with Z-buffer
  glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH);

  // Create window
  glutCreateWindow("CUBES");

  //  Enable Z-buffer depth test
  glEnable(GL_DEPTH_TEST);

  // Callback functions
  glutDisplayFunc(display);
  glutSpecialFunc(specialKeys);

  //  Pass control to GLUT for events
  glutMainLoop();

  //  Return to OS
  return 0;
}
```

Output:





EXPERIMENT NO. 7

Aim: Program to draw a house and show the rising and setting of sun using openGL.

Code:

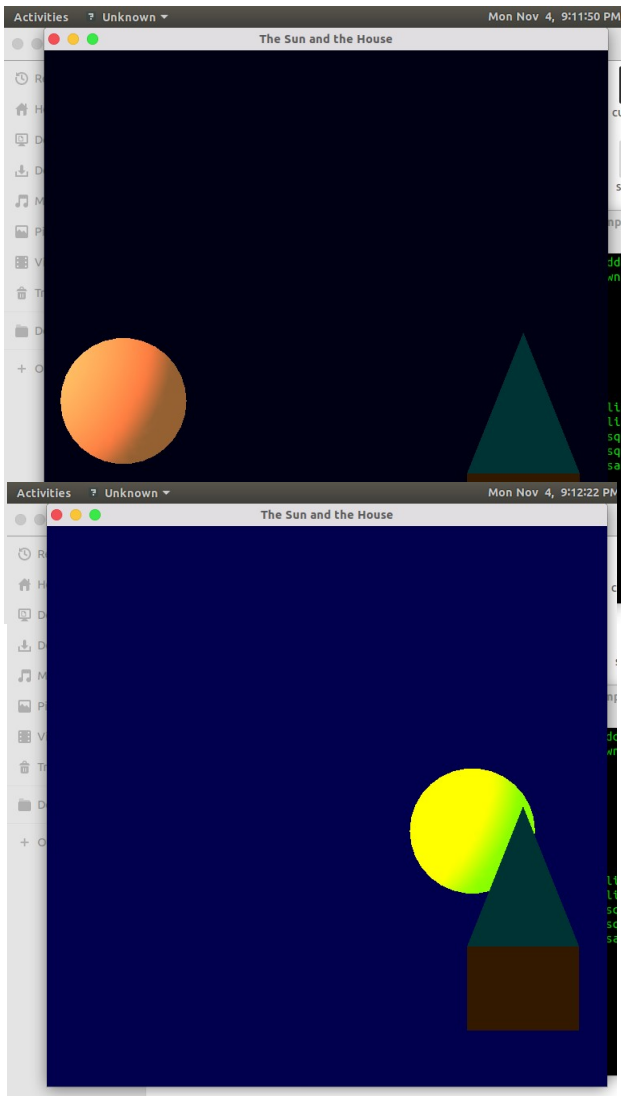
```
#include<iostream>
#include<stdlib.h>
#include<GL/glut.h>
using namespace std;

float ballX = -0.8f;
float ballY = -0.3f;
float ballZ = -1.2f;
float colR = 3.0;
float colG = 1.5;
float colB = 1.0;
float bgColR = 0.0;
float bgColG = 0.0;
float bgColB = 0.0;

static int flag = 1;

void drawBall(void){
glColor3f(colR, colG, colB); // set Ball color
glTranslatef(ballX, ballY, ballZ);
glutSolidSphere(0.3,30,30);
}

void drawHut(void){
glBegin(GL_POLYGON);
glColor3f(0.0,1.0,1.0);
glVertex3f(0.7,0.0,-1.0);
glVertex3f(0.5,-0.5,-1.0);
glVertex3f(0.9,-0.5,-1.0);
glEnd();

glBegin(GL_QUADS);
glColor3f(1.0,0.5,0.0);
glVertex3f(0.5,-0.5,-1.0);
glVertex3f(0.9,-0.5,-1.0);
glVertex3f(0.9,-0.8,-1.0);
glVertex3f(0.5,-0.8,-1.0);
glEnd();
```

```
}

void drawHill(void){
glBegin(GL_QUADS);
glColor3f(1.0,1.0,1.0);
glVertex3f(2.0,1.0,1.0);
glVertex3f(0.9,1.0,1.0);
glVertex3f(0.9,2.0,1.0);
glVertex3f(2.0,2.0,1.0);
glEnd();
}

void initRendering(){
glEnable(GL_DEPTH_TEST);
glEnable(GL_COLOR_MATERIAL);
glEnable(GL_LIGHTING);
glEnable(GL_LIGHT0); //enable light #0
glEnable(GL_LIGHT1); //enable light #1
glEnable(GL_NORMALIZE); //Auto normalize
glShadeModel(GL_SMOOTH); //Enable smooth shading
}

void drawScene(){
glClear(GL_COLOR_BUFFER_BIT|GL_DEPTH_BUFFER_BIT);
glClearColor(bgColR,bgColG,bgColB,0.0);
glMatrixMode(GL_MODELVIEW);

glLoadIdentity();
//Add Ambient Light
GLfloat ambientColor[] = {0.2f, 0.2f,0.2f,1.0f};
glLightModelfv(GL_LIGHT_MODEL_AMBIENT, ambientColor);

//Add positional lighting
GLfloat lightColor0[] = {0.5f,0.5f,0.5f,1.0f};
GLfloat lightPos0[] = {4.0f,0.0f,8.0f,1.0f}; //Positioned at (4,0,8)
glLightfv(GL_LIGHT0, GL_DIFFUSE,lightColor0);
glLightfv(GL_LIGHT0, GL_POSITION,lightPos0);

//Add directed Light
GLfloat lightColor1[] = {0.5f,0.2f,0.2f,1.0f}; //color
GLfloat lightPos1[] = {-1.0f,0.5f,0.5f,0.0f}; //From the direction 10505
glLightfv(GL_LIGHT0, GL_DIFFUSE,lightColor1);
glLightfv(GL_LIGHT0, GL_POSITION,lightPos1);
```

```
//drawing the sun
glPushMatrix();
drawBall();
glPopMatrix();
//the house
glPushMatrix();
drawHut();
glPopMatrix();

//the Hill
glPushMatrix();
drawHill();
glPopMatrix();

glutSwapBuffers();
}

void update(int value){
if(ballX>0.9f){
ballX = -0.8f;
ballY = -0.3f;
flag = 1;
colR = 2.0;
colG = 1.5;
colB = 1.0;
bgColB = 0.0;
}
if(flag){
ballX += 0.001f;
ballY += 0.0007f;
colR -= 0.001;
colG += 0.005;
bgColB += 0.001;

if(ballX>0.01){
flag = 0;
}
}
if(!flag){
ballX += 0.001f;
ballY -= 0.0007f;
colR += 0.001;
colB -= 0.01;
bgColB -= 0.001;
```

```
if(ballX<-0.3){
flag = 1;
}
}
glutPostRedisplay();
glutTimerFunc(25,update,0);//call the method update after 25 msec
}

int main(int argc, char** argv){
glutInit(&argc, argv);
glutInitDisplayMode(GLUT_DOUBLE|GLUT_RGB|GLUT_DEPTH);
glutInitWindowSize(700,700);
glutCreateWindow("The Sun and the House");
initRendering();
glutDisplayFunc(drawScene);
glutTimerFunc(25,update,0);
glutMainLoop();
return(0);
}
```

Output:



EXPERIMENT NO. 8

Aim: Program to draw Solar system showing rotation and revolution of the Sun, the Moon and the Earth.

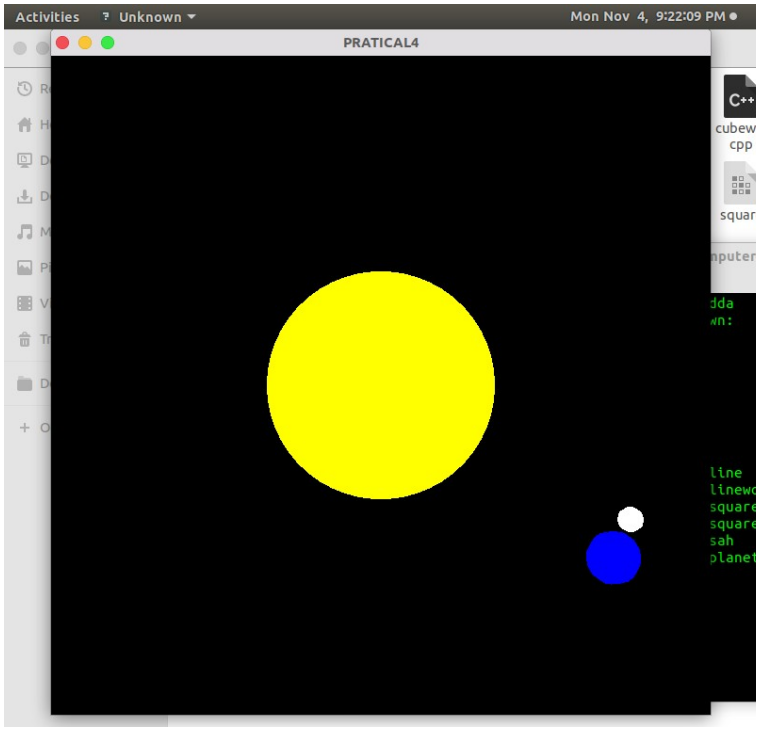
Code:

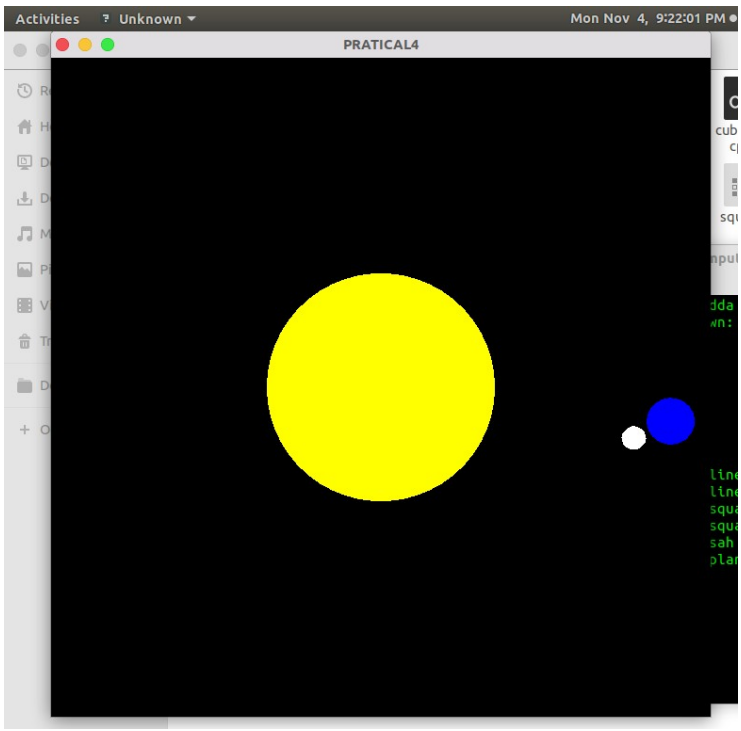
```
#include<stdio.h>
#include <GL/glut.h>
#include<math.h>
#define SunSize 0.5
#define EarthSize 0.10
#define MoonSize 0.05
#define SpeedMultiplicator 2.0
#define DaysPerYear 10
GLfloat year=0.0;   // degrees
GLfloat day=0.0;
GLfloat moonAroundEarth=0.0;
GLfloat moonItself=0.0;
GLfloat EarthOrbitRadius=1.25;
GLfloat MoonOrbitRadius=0.20;
GLfloat daySpeed=5.0*SpeedMultiplicator;
GLfloat yearSpeed= 10/(360.0*10*2);
GLfloat moonAroundEarthSpeed=0.09*SpeedMultiplicator;
GLfloat moonItselfSpeed=1*SpeedMultiplicator;

void RenderScene(void){
glPushMatrix();
//To rotate around drawn object
gluLookAt(0.0,0.0,-4.0,0.0,0.0,1.0,0.0,-3.0,0.0);
glColor3f(1.0,1.0,0.0);
glutSolidSphere(SunSize,50,50);
glRotatef(year,0.0,1.0,1.0);
glTranslatef(EarthOrbitRadius,0.0,0.0);
glPushMatrix();
glRotatef(day,0.25,1.0,1.0);
glColor3f(0.0,0.0,1.0);
glutSolidSphere(EarthSize,10,10);
glPopMatrix();
glRotatef(moonAroundEarth,0.0,1.0,1.0);
glTranslatef(MoonOrbitRadius,0.0,0.0);
glRotatef(moonItself,0.0,1.0,0.0);
glColor3f(1.0,1.0,1.0);
glutSolidSphere(MoonSize,8,8);
```

```
glPopMatrix();
glPopMatrix();
}
void Init(void){

glClearColor(0.0,0.0,0.0,0.0);
glClearDepth(0.0);
glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
}
void Display(void)
{
glClear(GL_COLOR_BUFFER_BIT);
RenderScene();
glFlush();
glutSwapBuffers();
}
void Reshape(int x,int y)
{
if(y==0)return;
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
gluPerspective(40.0,(GLdouble)x/(GLdouble)y,0.5,20.0);
glMatrixMode(GL_MODELVIEW);
glViewport(0,0,x,y);
Display();
}
void Idle(void)
{
day+=daySpeed;
year+= yearSpeed;
moonItself+= moonItselfSpeed;
moonAroundEarth+=moonAroundEarthSpeed;
Display();
}
int main(int argc,char**argv)
{
glutInit(&argc,argv);
glutInitDisplayMode(GLUT_DOUBLE|GLUT_RGB);
glutInitWindowSize(700,700);
glutCreateWindow("PRATICAL4");
Init();
glutReshapeFunc(Reshape);
glutDisplayFunc(Display);
```

```
glutIdleFunc(Idle);
glutMainLoop();
return 0;
}
```

Output:





EXPERIMENT NO. 9

Aim: Program to draw a ball and show its bouncing motion.

Code:

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


bool fullscreen = false;
float posx,posy,movespeedx,movespeedy;


bool init()
{
	glClearColor(0.93f, 0.93f, 0.93f, 0.0f);
	glColor3f(1.0f, 0.0f, 0.0f);

	return true;
}

void display()
{
	glClear(GL_COLOR_BUFFER_BIT);
	float PI=3.14159265;
	float radius=0.09f;

	if (posx+radius>1.0f || posx-radius<-1.0f)
	{
		movespeedx*=-1;
	}
	if (posy+radius>1.0f || posy-radius<-1.0f)
	{
		movespeedy*=-1;
	}
	posx+=movespeedx;
	posy+=movespeedy;

	glBegin(GL_POLYGON);
	for (float angle = 0;angle<360;angle+=5)
```

```
	{
		float xc=sin(angle*PI/180) * radius;
		float yc=cos(angle*PI/180) * radius;
		glVertex3f( xc+posx, yc+posy ,0.0f);
	}
	glEnd();
	glFlush();
	glutSwapBuffers();
	glutPostRedisplay();
}

void keyboard(unsigned char key, int x, int y)
{
	if (key == 27)
		exit(1);
}

void specialKeyboard(int key, int x, int y)
{
	switch(key)
	{
	case GLUT_KEY_F1:
		fullscreen = !fullscreen;
		if (fullscreen)
			glutFullScreen();
		else
		{
			glutReshapeWindow(500, 500);
			glutPositionWindow(50, 50);
		}
		break;
	}
}

int main(int argc, char *argv[])
{
	srand ( time(NULL) );
	int step=100;
	posx=(rand()%step)/step-0.5f;
	posy=(rand()%step)/step-0.5f;
	movespeedx=0.0001f;
	movespeedy=0.0002f;

	glutInit(&argc, argv);
```

```
        glutInitWindowPosition(100, 100);
        glutInitWindowSize(720, 720);

        glutInitDisplayMode(GLUT_RGB | GLUT_DOUBLE);

        glutCreateWindow("05 - Primitives");

        glutDisplayFunc(display);
        glutKeyboardFunc(keyboard);
        glutSpecialFunc(specialKeyboard);
        if (!init())
                return 1;

        glutMainLoop();

        return 0;
}
```

Output: