1. **Write a program to rotate a square with its center.**

#include<stdlib.h>

#ifdef \_\_APPLE\_\_

#include<openGL/openGL.h>

#include<GLUT/glut.h>

#else

#include<GL/glut.h>

#endif

float ballX = -0.3f;

float ballY = 0.0f;

float ballZ = -1.0f;

static int flag=1;

void drawBall(void) {

glColor3f(1.0, 1.0, 0.0); //set ball colour

glTranslatef(ballX,ballY,ballZ); //moving it toward the screen a bit on creation

glutSolidSphere (.1,40,40); //create ball.

}

void keyPress(int key, int x, int y)

{

if(key==GLUT\_KEY\_RIGHT)

ballX += 0.05f;

if(key==GLUT\_KEY\_LEFT)

ballX -= 0.05f;

if(key==GLUT\_KEY\_UP)

ballY +=0.05f;

if(key==GLUT\_KEY\_DOWN)

ballY -=0.05f;

glutPostRedisplay();

}

void initRendering()

{

glEnable(GL\_DEPTH\_TEST);

}

//Called when the window is resized

void handleResize(int w, int h) {

//Tell OpenGL how to convert from coordinates to pixel values

glViewport(0, 0, w, h);

glMatrixMode(GL\_PROJECTION); //Switch to setting the camera perspective

//Set the camera perspective

glLoadIdentity(); //Reset the camera

gluPerspective(45.0, //The camera angle

(double)w / (double)h, //The width-to-height ratio

1.0, //The near z clipping coordinate

200.0); //The far z clipping coordinate

}

void drawScene()

{

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

glMatrixMode(GL\_MODELVIEW);

glLoadIdentity();

drawBall();

glutSwapBuffers();

}

void update(int value) {

if(flag)

{

ballX += 0.001f;

if(ballX>0.3)

{

flag=0;

}

}

if (!flag)

{

ballX -= 0.001f;

if(ballX<-0.3)

{

flag=1;

}

}

glutPostRedisplay(); //Tell GLUT that the display has changed

//Tell GLUT to call update again in 25 milliseconds

glutTimerFunc(25, update, 0);

}

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

{

glutInit(&argc,argv);

glutInitDisplayMode(GLUT\_DOUBLE|GLUT\_RGB|GLUT\_DEPTH);

glutInitWindowSize(400,400);

glutCreateWindow("SArfraz 32");

initRendering();

glutDisplayFunc(drawScene);

glutSpecialFunc(keyPress);

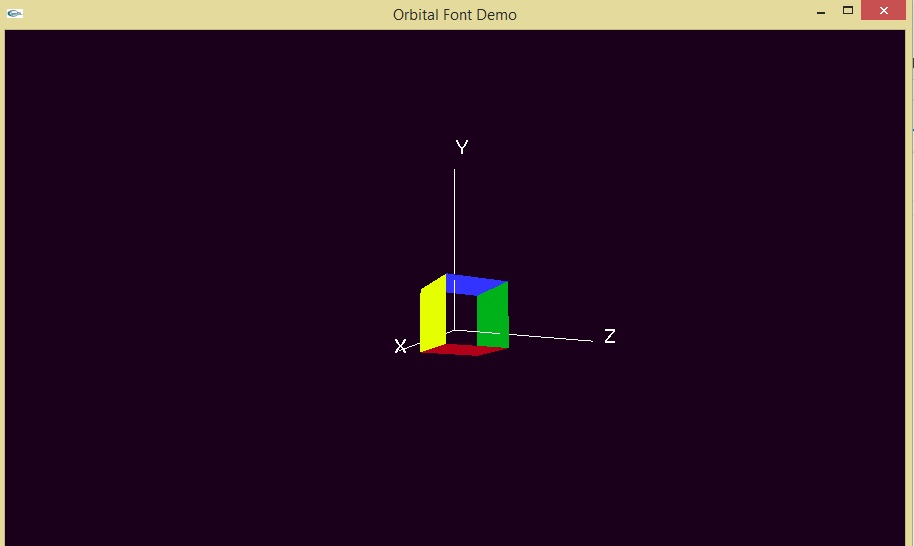
glutReshapeFunc(handleResize);

glutTimerFunc(25, update, 0);

glutMainLoop();

return(0);

}



1. **Write a program make a ball rotate on circumference of circle.**

#include <GLUT/glut.h>

#include <OpenGL/gl.h>

#include <OpenGL/glu.h>

#include <math.h>

#include <stdio.h>

#include <string.h>

#define WIDTH 640

#define HEIGHT 480

#ifdef \_\_APPLE\_\_

#include<GLUT/glut.h>

#include<openGL/openGL.h>

#include<math.h>

#include<stdio.h>

#else

#include<GL/glut.h>

#endif

float x1=1.0,x2=1.0,an=0,bn=0,ca=0.0009,da=0.0009;

float x3=-99.0,x4=-99.0;

void initRendering()

{

glDisable(GL\_DEPTH\_TEST);

}

void reshape(int w,int h)

{

glViewport(0,0,w,h);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluPerspective(45,w/h,1,20);

}

void keyPressed(int key,int x,int y)

{

if(key==GLUT\_KEY\_LEFT)

{}

}

void Ball()

{

glColor3f(0.0,0.0,1.0);

glPushMatrix();

glTranslatef(0,0,-5.0);

glutSolidSphere(0.9,45,9);

glPopMatrix();

glColor3f(0.0,1.0,0.0);

glPushMatrix();

glTranslatef(-x2,-x1,-6.5);

glutSolidSphere(0.2,9,9);

glPopMatrix();

}

void update()

{

an += da;

x1=sin(an);

x2=cos(an);

}

void display()

{

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

glMatrixMode(GL\_MODELVIEW);

glLoadIdentity();

printf("HELLO WORLD\n");

Ball();

update();

glutSwapBuffers();

}

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

{

glutInit(&argc,argv);

glutInitDisplayMode(GLUT\_DOUBLE|GLUT\_RGB|GLUT\_DEPTH);

glutInitWindowSize(400,400);

glutCreateWindow("Collision Window");

initRendering();

glutDisplayFunc(display);

glutIdleFunc(display);

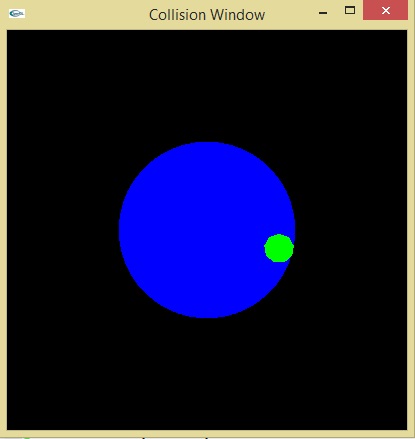
glutReshapeFunc(reshape);

glutSpecialFunc(keyPressed);

glutMainLoop();

return(0);

}



1. **Write a program to move a square on a cursor position given by user in opengl.**

#ifdef \_\_APPLE\_\_

#include<GLUT/glut.h>

#include<openGL/openGL.h>

#include<string.h>

#include<stdio.h>

#else

#include<GL/glut.h>

#endif

double arr[5000][4];

int z=0;

int flag=0;

float radius=0.03;

int ptr=0;

int faltu\_bit=1;

float color[3][3]={{1.0,1.0,1.0},{1.0,1.0,0.0},{0.0,1.0,0.0}};

void init()

{

glClearColor( 0.0, 0.0, 0.0, 1.0);

glMatrixMode( GL\_PROJECTION);

gluOrtho2D(0.0,600,0.0,600);

memset(arr,0,5000);

glPointSize(20.0);

}

void resetAll()

{

memset(arr,0,5000);

z=0;

}

///OPENGL MAPPING///

float getOpenGLX(int x)

{

double ox = x/ (double)600\*(600);

return ox;

}

float getOpenGLY(int y)

{

double oy = (1 - y/ (double) 600)\*600;

return oy;

}

void drawPoints()

{

glBegin( GL\_POINTS );

glColor3f( 0.0,1.0,0.0 );

for ( int i = 0; i < z; i++ )

{

glVertex2f( arr[i][0],arr[i][1]);

}

glEnd();

}

void drawBall(float x,float y)

{

glBegin( GL\_POINTS);

glColor3f( 1.0,1.0,0.0 );

glVertex2f(x,y);

glEnd();

}

void drawLines()

{

glBegin(GL\_LINES);

glColor3f(1.0,0.0,0.0);

for(int i=0;i<z;i++)

{

glVertex2f(arr[i][0],arr[i][1]);

}

glEnd();

}

void addValue(int x,int y)

{

arr[z][0]=getOpenGLX(x);

arr[z++][1]=getOpenGLY(y);

}

void trackBall()

{

drawPoints();

}

void myDisplay()

{

glClear( GL\_COLOR\_BUFFER\_BIT);

if(!flag)

{

drawLines();

if(!faltu\_bit)

drawBall(arr[ptr][0],arr[ptr][1]);

}

glutSwapBuffers();

glutPostRedisplay();

glFlush();

}

void myMouseStat(int button,int state,int x, int y)

{

if(button==GLUT\_LEFT\_BUTTON && state==GLUT\_DOWN)

{

if(!flag)

{

if(faltu\_bit)

{

faltu\_bit=0;

}

resetAll();

flag=1;

}

}

else if(button==GLUT\_LEFT\_BUTTON && state==GLUT\_UP)

{

if(flag)

{

ptr=0;

flag=0;

}

}

}

void myPressedMove(int x,int y)

{

if(flag)

{

addValue(x,y);

}

}

void myTimer(int t)

{

if(ptr!=z)

{

ptr++;

}

else

{

ptr=0;

}

glutTimerFunc(100,myTimer,0);

}

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

{

glutInit( &argc, argv);

glutInitDisplayMode( GLUT\_DOUBLE| GLUT\_RGB);

glutInitWindowPosition( 100, 100);

glutInitWindowSize(600,600);

glutCreateWindow( "Testing");

init();

glutDisplayFunc(myDisplay);

glutMouseFunc(myMouseStat);

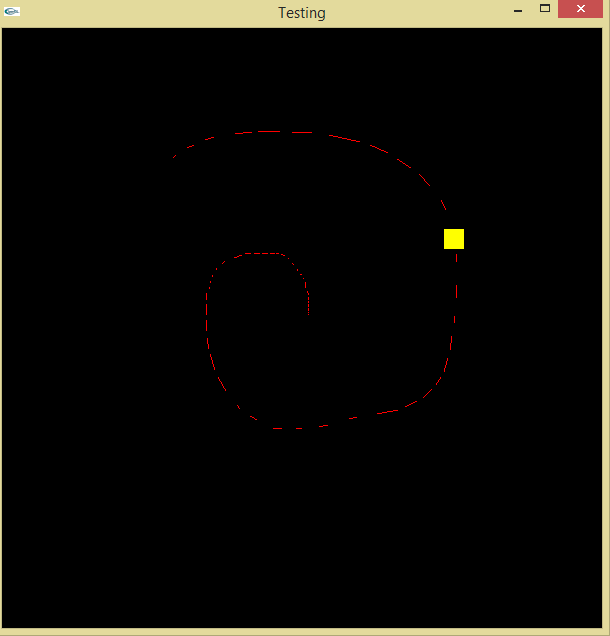
glutMotionFunc(myPressedMove);

glutTimerFunc(100,myTimer,0);

glutMainLoop();

return 0;

}



1. **Write a program to inscribe the desired number of square inside a square in opengl.**

#include<stdlib.h>

#ifdef \_\_APPLE\_\_

#include<openGL/openGL.h>

#include<GLUT/glut.h>

#else

#include <GL/glut.h>

#include <math.h>

#include <stdio.h>

#include <string.h>

#endif

using namespace std;

//using namespace std;

int n;

/\*float ballX = -0.3f;

float ballY = 0.0f;

float ballZ = -1.0f;\*/

float ballX=-0.3f;

float ballY=-0.3f;

float ballZ=-1.0f;

float ballX2 = 0.3f;

float ballY2 = 0.0f;

float ballZ2 = -1.0f;

float c1,c2,c3;

void drawBall(void) {

glColor3f(1.5, 0.5, 0.3); //set ball colour

glTranslatef(ballX,ballY,ballZ); //moving it toward the screen a bit on creation

glutSolidSphere (0.1, 40, 40); //create ball.

}

void init()

{

//cout<< "Enter the number of figures"<< endl;

//cin>>"%d">>n;

printf("Enter the number of figures ");

//scanf("%d",&n);

scanf\_s("%d",&n);

}

void drawBall2(int k) {

glColor3f(1.0, 1.0, 0.0); //set ball colour

glVertex3f(0.05,0.05,0.0);

glVertex3f(0.5,0.05,0.0);

glVertex3f(0.5,0.5,0.0);

glVertex3f(0.05,0.5,0.0);

glEnd();

\*/

glBegin(GL\_LINE\_LOOP);

glVertex3f(-0.25/k,-0.25/k,0.0);

glVertex3f(0.25/k,-0.25/k,0.0);

glVertex3f(0.25/k,0.25/k,0.0);

glVertex3f(-0.25/k,0.25/k,0.0);

glEnd();

glBegin(GL\_LINE\_LOOP);

glVertex3f(0.0,-0.25/k,0.0);

glVertex3f(0.25/k,0.0,0.0);

glVertex3f(0.0,0.25/k,0.0);

glVertex3f(-0.25/k,0.0,0.0);

glEnd();

}

void drawRectangle() {

glColor3f(0.0,1.0,0.0);

glTranslatef(ballX2,ballY2,ballZ2); //moving it toward the screen a bit on creation

//glTranslatef(0.3,0.0,-1.0);

glBegin(GL\_POLYGON);

glVertex3f(0.25,0.25,0.0);

glVertex3f(0.75,0.25,0.0);

glVertex3f(0.75,0.75,0.0);

glVertex3f(0.25,0.75,0.0);

glEnd();

void initRendering()

{

glEnable(GL\_DEPTH\_TEST);

}

//Called when the window is resized

void handleResize(int w, int h) {

//Tell OpenGL how to convert from coordinates to pixel values

glViewport(0, 0, w, h);

glMatrixMode(GL\_PROJECTION); //Switch to setting the camera perspective

//Set the camera perspective

glLoadIdentity(); //Reset the camera

gluPerspective(45.0, //The camera angle

(double)w / (double)h, //The width-to-height ratio

1.0, //The near z clipping coordinate

200.0); //The far z clipping coordinate

}

void drawScene()

{

int x=1;

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

glMatrixMode(GL\_MODELVIEW);

glLoadIdentity();

glTranslatef(0.0,0.0,-1.0);

drawBall2(1.0);

for(int i=1;i<n/2;i++){

drawBall2(2.0\*x);

x=x\*2;}

glutSwapBuffers();

}

//float \_angle = 30.0f;

void update(int value) {

glutPostRedisplay(); //Tell GLUT that the display has changed

//Tell GLUT to call update again in 25 milliseconds

glutTimerFunc(25, update, 0);

}

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

{

init();

glutInit(&argc,argv);

glutInitDisplayMode(GLUT\_DOUBLE|GLUT\_RGB|GLUT\_DEPTH);

glutInitWindowSize(800,800);

glutCreateWindow("lab");

initRendering();

glutDisplayFunc(drawScene);

//glutKeyBoardFunc(keyPress);

//glutKeyboardFunc(keyPress);

//glutSpecialFunc(keyPress);

glutReshapeFunc(handleResize);

//autorot();

glutTimerFunc(25, update, 0);

glutMainLoop();

return(0);

}

