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

static GLfloat up=-0.2;
static GLfloat pos=-0.2;
int shoot=0,bang=0;
int counter1=0,counter2=0,count=0;
int game,instruct;
char tmp_str[40];

void display2();
void displost();

void init(void)
{


   GLfloat mat_specular[] = { 1.0, 1.0, 1.0, 1.0 };
   GLfloat mat_shininess[] = { 50.0 };
   GLfloat mat_diffuse[]={ 1.0,1.0,1.0,1.0};
   GLfloat mat_ambient[]={0.0,0.0,0.0,1.0};
   GLfloat light_position[] = { 1.0, 1.0, 0.0, 0.0 };

   glClearColor (0.0, 0.0, 0.0, 0.0);
   glShadeModel (GL_SMOOTH);

   glMaterialfv(GL_FRONT, GL_SPECULAR, mat_specular);
   glMaterialfv(GL_FRONT, GL_SHININESS, mat_shininess);
   glMaterialfv(GL_FRONT, GL_DIFFUSE, mat_diffuse);
   glMaterialfv(GL_FRONT, GL_AMBIENT, mat_ambient);


   glLightfv(GL_LIGHT0, GL_POSITION, light_position);
   glColorMaterial(GL_FRONT_AND_BACK, GL_AMBIENT_AND_DIFFUSE);

   glEnable(GL_LIGHTING);
   glEnable(GL_LIGHT0);
   glEnable(GL_DEPTH_TEST);
   glEnable(GL_COLOR_MATERIAL);
}


void drawhit(const char * message, int x, int y)
{
   glPushMatrix();

   glScalef(0.3,0.2,0.15);
    glTranslatef(x,y,0);
   while (*message)
   {
         glutStrokeCharacter(GLUT_STROKE_ROMAN,*message++);
```

```
    }

    glPopMatrix();
}

void myHit()
{
    glClear(GL_COLOR_BUFFER_BIT);
     glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluOrtho2D(0,200,0,200);
    glMatrixMode(GL_MODELVIEW);
    glClearColor(1.0,0.0,0.5,1.0);
    glColor3f(0.0,0.8,0.80);

    glBlendFunc(GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA);
    glEnable(GL_BLEND);
    glEnable(GL_LINE_SMOOTH);
    glLineWidth(4.0);

     drawhit("WINNER!!",70,550);


}

void draw_instruct(const char *message, int x, int y)
{
    int j;

    glPushMatrix();

    glScalef(0.1,0.1,0.0);
       glTranslatef(x,y,0);
    while (*message)
    {
          glutStrokeCharacter(GLUT_STROKE_ROMAN,*message++);

    }
       for(j=0;j<10000;j++);
    glPopMatrix();
}

void instructions()
{
    glClear(GL_COLOR_BUFFER_BIT);
       glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluOrtho2D(0,200,0,200);
    glMatrixMode(GL_MODELVIEW);
    glClearColor(1.0,0.7,0.0,1.0);
    glColor3f(1.0,0.5,0.1);

    glBlendFunc(GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA);
    glEnable(GL_BLEND);
    glEnable(GL_LINE_SMOOTH);
    glLineWidth(4.0);
```

```
        draw_instruct("Instructions",600,1850); // change1
    draw_instruct("Right click on mouse",300,1700);
    draw_instruct("to play",300,1500);
        draw_instruct("Press f to shoot",300,1300);
    glFlush();


}

void Write(char *string)
{
    glScalef(0.02,0.02,0.0);
       while(*string)
       glutBitmapCharacter(GLUT_BITMAP_HELVETICA_18, *string++);
}

void display1()
{

    int i;

    if(counter1==3)
    {
       display2();
       glFlush();
    }
    else
    {     int j;
          for(j=0;j<10000;j++);

          glClearColor(1.0,0.7,0.0,1.0);
          glClear(GL_COLOR_BUFFER_BIT|GL_DEPTH_BUFFER_BIT);
          glPushMatrix();
          glColor3f(1, 1, 0);
          glRasterPos2f(-0.9, 0.9);
          sprintf(tmp_str, "Arrow count: %d", count);
          Write(tmp_str);
          glPopMatrix();
          if(count>=30)

                                    glutDisplayFunc(displost);
                               glPushMatrix();
          glColor3f(1, 1, 0);
          glRasterPos2f(-0.2, 0.9);
          sprintf(tmp_str, "Score: %d", counter1);
          Write(tmp_str);
          glPopMatrix();
          glPushMatrix();
          glColor3f(1.0,0.0,0.0);
          glLoadIdentity();
          glTranslatef(0.8,-0.869+up,0.0);
          glutSolidSphere(0.15,20,16);

          if(shoot==1)
          {

                glPushMatrix();
```

```
                glLoadIdentity();
                glTranslatef(-0.8+pos,0.0,0.0);
                glColor3f(0.0,0.0,0.0);
                glLineWidth(2.0);
                glBegin(GL_LINES);
                glVertex3f(-0.2,0.0,0.0);
                glVertex3f(0.1,0.0,0.0);
                glVertex3f(0.1,0.0,0.0);
                glVertex3f(0.03,0.05,0.0);
                glVertex3f(0.1,0.0,0.0);
                glVertex3f(0.03,-0.05,0.0);
                glEnd();
                glPopMatrix();
        }
        if(bang==1)
        {

                bang=0;pos=-0.2;

                glPushMatrix();
                glLoadIdentity();
                up=0;
                glColor3f(1.0,0.0,0.0);
                glutSolidSphere(1,20,16);
                glPopMatrix();
        }
        glPopMatrix();


        for( i=0;i<200;i=i+20)
        {
                if(pos>=1.74 && up>0.825 && up<0.975)
//collision detection
                {
                        counter1 ++;
                        for(j=0;j<10000;j++);

                        shoot=0;
                        pos=-0.2;
                        bang=1;
                }
                if(counter1==3)
                        count=0;
                up=(up+0.005);

                if(up>2)
                        up=0;
                if(shoot==1)
                {
                        pos=pos+0.009;
                        if(pos>2)
                        {
                                pos=-0.2;
                                shoot=0;
                        }

                }
```

```
                glutPostRedisplay();
            }

    glFlush();
    }
}


void display2()
{int i;

    if(counter2==3)
    { myHit();
      glFlush();
    }

    else
    {int j;
          for(j=0;j<10000;j++);
    glClearColor(1.0,0.7,0.0,1.0);
    glClear(GL_COLOR_BUFFER_BIT|GL_DEPTH_BUFFER_BIT);
    glLoadIdentity();

    glPushMatrix();
    glColor3f(1, 1, 0);
    glRasterPos2f(-0.9, 0.9);
    sprintf(tmp_str, "Arrow count: %d", count);
    Write(tmp_str);
    glPopMatrix();
    if(count>=20)
    glutDisplayFunc(displost);
    glPushMatrix();
    glColor3f(1, 1, 0);
    glRasterPos2f(-0.2, 0.9);
    sprintf(tmp_str, "Score: %d", counter2);
    Write(tmp_str);
    glPopMatrix();

    glPushMatrix();
    glColor3f(1.0,0.0,0.0);
    glLoadIdentity();
    glTranslatef(0.8,-0.769+up,0.0);
    glutSolidSphere(0.10,20,16);
    glColor3f(0.0,0.0,1.0);


    glPushMatrix();
    glColor3f(0.0,0.0,1.0);
    glLoadIdentity();
    glTranslatef(0.4,0.769-up,0.0);
    glutSolidSphere(0.10,20,16);
    glColor3f(0.0,0.0,1.0);

    if(shoot==1)
    {
```

```
glPushMatrix();
glLoadIdentity();
glTranslatef(-0.8+pos,0.0,0.0);
glColor3f(0.0,0.0,0.0);
glLineWidth(2.0);
   glBegin(GL_LINES);
glVertex3f(-0.2,0.0,0.0);
glVertex3f(0.1,0.0,0.0);
glVertex3f(0.1,0.0,0.0);
glVertex3f(0.03,0.05,0.0);
   glVertex3f(0.1,0.0,0.0);
glVertex3f(0.03,-0.05,0.0);
glEnd();
glPopMatrix();
}
if(bang==1)
{


      bang=0;pos=-0.2;
      glPushMatrix();
         glLoadIdentity();

          up=0;
      glColor3f(1.0,0.0,0.0);
      glutSolidSphere(1,20,16);
         glPopMatrix();
}
glPopMatrix();


for( i=0;i<200;i=i+20)
{


if(pos>=1.75 && up>0.825 && up<0.975)
{

 counter2 ++;
 for(j=0;j<10000;j++);

shoot=0;
pos=-0.2;
bang=1;
}
up=(up+0.005);
if(up>2)
      up=0;
if(shoot==1)
{
      pos=pos+0.009;
if(pos>2)
{
      pos=-0.2;
    shoot=0;
 }
```

```
    }
    glutPostRedisplay();
    }

    }

    glFlush();
}


void display()
{
        glClearColor(1.0,0.7,0.0,1.0);
    glClear(GL_COLOR_BUFFER_BIT|GL_DEPTH_BUFFER_BIT);
    glFlush();
}
void displost()
{
          glClear(GL_COLOR_BUFFER_BIT);
          glMatrixMode(GL_PROJECTION);
          glLoadIdentity();
          gluOrtho2D(0,200,0,200);
          glMatrixMode(GL_MODELVIEW);
          glClearColor(1.0,0.0,0.5,1.0);
          glColor3f(0.0,0.8,0.80);

          glBlendFunc(GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA);
          glEnable(GL_BLEND);
          glEnable(GL_LINE_SMOOTH);
          glLineWidth(4.0);

        drawhit("you lost!!",70,550);
        glFlush();


}

void indisplay()
{
     glClearColor(1.0,0.7,0.0,1.0);
    glClear(GL_COLOR_BUFFER_BIT|GL_DEPTH_BUFFER_BIT);
    instructions();
    glFlush();
}



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

    if (key=='f')
    {
          shoot=1;

          count++;
```

```
    }


}



void choose(int i)
{
   switch(i)
   { case 1: exit(0);
     case 2: glutDisplayFunc(display1);
                break;
     case 3: glutDisplayFunc(display2);
               break;
     default:exit(0);
           }
}


int main(int argc,char **argv)
{
   glutInit(&argc,argv);

          glutInitDisplayMode(GLUT_DEPTH|GLUT_RGB);
   glutInitWindowSize(1500,1500);
   glutInitWindowPosition(0,0);
   instruct=glutCreateWindow("Instructions");
   init();
   glutDisplayFunc(indisplay);


   glutInitDisplayMode(GLUT_DEPTH|GLUT_RGB);
   glutInitWindowSize(1000,1000);
   glutInitWindowPosition(0,0);
   game=glutCreateWindow("Proj");
   init();
   glutDisplayFunc(display);
   glutKeyboardFunc(keyboard);
   glutCreateMenu(choose);
   glutAddMenuEntry("Quit",1);
   glutAddMenuEntry("PlayLevel1",2);
   glutAddMenuEntry("PlayLevel2",3);
   glutAttachMenu(GLUT_RIGHT_BUTTON);
   glutMainLoop();
   return 0;
}
```