```
#include<stdio.h>
#include<GL/glut.h>
#include<string.h>
int maxy=600;
int count=0;
int maxx=500;
int n=3;
int m=3;
int
count1=0,count2=0,count3=0,count4=0,count5=0,count6=0,count7=0,count8=
0,count9=0,count10=0;
int x=25,y=50;
void id1();
void id();
void draw_target();
void redraw();


/*to display bitmap char*/
void bitmap_output(int x,int y,char *string,void *font)
{
int len,i;
glRasterPos2f(x,y);
len=(int)strlen(string);
for(i=0;i<len;i++)
{
glutBitmapCharacter(font,string[i]);
}
return;
}


/*based on count display no of arrows and result of game*/
void counting()
{
if(count==0)
bitmap_output(40,40,"No of Arrows:0",GLUT_BITMAP_HELVETICA_18);
else if(count==1)
bitmap_output(40,40,"No of Arrows:1",GLUT_BITMAP_HELVETICA_18);
else if(count==2)
bitmap_output(40,40,"No of Arrows:2",GLUT_BITMAP_HELVETICA_18);
else if(count==3)
bitmap_output(40,40,"No of Arrows:3",GLUT_BITMAP_HELVETICA_18);
if(count==4)
bitmap_output(40,40,"No of Arrows:4",GLUT_BITMAP_HELVETICA_18);
else if(count==5)
bitmap_output(40,40,"No of Arrows:5",GLUT_BITMAP_HELVETICA_18);
else if(count==6)
bitmap_output(40,40,"No of Arrows:6",GLUT_BITMAP_HELVETICA_18);
else if(count==7)
bitmap_output(40,40,"No of Arrows:7",GLUT_BITMAP_HELVETICA_18);
else if(count==8)
```

```
bitmap_output(40,40,"No of Arrows:8",GLUT_BITMAP_HELVETICA_18);
else if(count==9)
bitmap_output(40,40,"No of Arrows:9 one arrow
remaining",GLUT_BITMAP_HELVETICA_18);
else
{
if(count1==1&&count2==1&&count3==1&&count4==1&&count5==1&&count6==1&&c
ount7==1&&count8==1&&count9==1&&count10==1)
bitmap_output(5,300,"CONGRAGULATION U
WON",GLUT_BITMAP_TIMES_ROMAN_24);
else
bitmap_output(5,300,"No of Arrows:10,NO ARROWS GAME OVER U
LOST",GLUT_BITMAP_TIMES_ROMAN_24);
glColor3f(0,0,0);
glBegin(GL_LINES);
glVertex2d(x,y);
glVertex2d(x+100,y);
glEnd();
glLineWidth(2);
glBegin(GL_LINES);
glVertex2d(x,y+2);
glVertex2d(x+100,y+2);
glEnd();
glBegin(GL_LINES);
glVertex2d(x,y-2);
glVertex2d(x+100,y-2);
glEnd();
glBegin(GL_TRIANGLES);
glVertex2d(x+100,y+3);
glVertex2d(x+110,y);
glVertex2d(x+100,y-3);
glEnd();
glBegin(GL_QUADS);
glVertex2d(x,y+3);
glVertex2d(x,y-3);
glVertex2d(x-10,y-5);
glVertex2d(x-10,y+5);
glEnd();
}
}


/*to redraw target which ar hit by player*/
void redraw()
{
glClear(GL_COLOR_BUFFER_BIT|GL_DEPTH_BUFFER_BIT);
if(count1==0)
{
glColor3f(1,1,0);
bitmap_output(150,450,"BLOCK SHOOTING",GLUT_BITMAP_TIMES_ROMAN_24);
counting();
glPointSize(30);
```

```
glColor3f(1,0,1);
glBegin(GL_POINTS);
glVertex2d(300,450);
glEnd();
glColor3f(0,1,1);
glBegin(GL_LINE_LOOP);
glVertex2d(285,465);
glVertex2d(315,465);
glVertex2d(315,435);
glVertex2d(285,435);
glEnd();
}
else
{
glColor3f(1,1,0);
bitmap_output(150,450,"BLOOK SHOOTING",GLUT_BITMAP_TIMES_ROMAN_24);
counting();
glColor3f(1,1,1);
glPointSize(20);
glBegin(GL_POINTS);
glVertex2d(300,450);
glEnd();
}

if(count2==0)
{
glColor3f(1,1,0);
bitmap_output(150,450,"BLOCK SHOOTING",GLUT_BITMAP_TIMES_ROMAN_24);
counting();
glPointSize(30);
glColor3f(1,0,1);
glBegin(GL_POINTS);
glVertex2d(400,480);
glEnd();
glColor3f(0,1,1);
glBegin(GL_LINE_LOOP);
glVertex2d(385,495);
glVertex2d(415,495);
glVertex2d(415,465);
glVertex2d(385,465);
glEnd();
}
else
{
glColor3f(1,1,0);
bitmap_output(150,450,"BLOOK SHOOTING",GLUT_BITMAP_TIMES_ROMAN_24);
counting();
glColor3f(1,1,1);
glPointSize(20);
glBegin(GL_POINTS);
glVertex2d(400,480);
glEnd();
```

```
}

if(count3==0)
{
glColor3f(1,1,0);
bitmap_output(150,450,"BLOCK SHOOTING",GLUT_BITMAP_TIMES_ROMAN_24);
counting();
glPointSize(30);
glColor3f(1,0,1);
glBegin(GL_POINTS);
glVertex2d(375,400);
glEnd();
glColor3f(0,1,1);
glBegin(GL_LINE_LOOP);
glVertex2d(360,415);
glVertex2d(390,415);
glVertex2d(390,385);
glVertex2d(360,385);
glEnd();
}
else
{
glColor3f(1,1,0);
bitmap_output(150,450,"BLOOK SHOOTING",GLUT_BITMAP_TIMES_ROMAN_24);
counting();
glColor3f(1,1,1);
glPointSize(20);
glBegin(GL_POINTS);
glVertex2d(375,400);
glEnd();
}
if(count4==0)
{
glColor3f(1,1,0);
bitmap_output(150,450,"BLOCK SHOOTING",GLUT_BITMAP_TIMES_ROMAN_24);
counting();
glPointSize(30);
glColor3f(1,0,1);
glBegin(GL_POINTS);
glVertex2d(250,370);
glEnd();
glColor3f(0,1,1);
glBegin(GL_LINE_LOOP);
glVertex2d(235,385);
glVertex2d(265,385);
glVertex2d(265,355);
glVertex2d(235,355);
glEnd();
}
else
{
glColor3f(1,1,0);
```

```
bitmap_output(150,450,"BLOOK SHOOTING",GLUT_BITMAP_TIMES_ROMAN_24);
counting();
glColor3f(1,1,1);
glPointSize(20);
glBegin(GL_POINTS);
glVertex2d(250,370);
glEnd();
}
if(count5==0)
{
glColor3f(1,1,0);
bitmap_output(150,450,"BLOCK SHOOTING",GLUT_BITMAP_TIMES_ROMAN_24);
counting();
glPointSize(30);
glColor3f(1,0,1);
glBegin(GL_POINTS);
glVertex2d(350,330);
glEnd();

glColor3f(0,1,1);
glBegin(GL_LINE_LOOP);
glVertex2d(335,345);
glVertex2d(365,345);
glVertex2d(365,315);
glVertex2d(335,315);
glEnd();
}
else
{
glColor3f(1,1,0);
bitmap_output(150,450,"BLOOK SHOOTING",GLUT_BITMAP_TIMES_ROMAN_24);
counting();
glColor3f(1,1,1);
glPointSize(20);
glBegin(GL_POINTS);
glVertex2d(350,330);
glEnd();
}

if(count6==0)
{
glColor3f(1,1,0);
bitmap_output(150,450,"BLOCK SHOOTING",GLUT_BITMAP_TIMES_ROMAN_24);
counting();
glPointSize(30);
glColor3f(1,0,1);
glBegin(GL_POINTS);
glVertex2d(450,290);
glEnd();

glColor3f(0,1,1);
glBegin(GL_LINE_LOOP);
```

```
glVertex2d(435,305);
glVertex2d(465,305);
glVertex2d(465,275);
glVertex2d(435,275);
glEnd();
}
else
{
glColor3f(1,1,0);
bitmap_output(150,450,"BLOOK SHOOTING",GLUT_BITMAP_TIMES_ROMAN_24);
counting();
glColor3f(1,1,1);
glPointSize(20);
glBegin(GL_POINTS);
glVertex2d(450,290);
glEnd();
}
if(count7==0)
{
glColor3f(1,1,0);
bitmap_output(150,450,"BLOCK SHOOTING",GLUT_BITMAP_TIMES_ROMAN_24);
counting();
glPointSize(30);
glColor3f(1,0,1);
glBegin(GL_POINTS);
glVertex2d(330,245);
glEnd();

glColor3f(0,1,1);
glBegin(GL_LINE_LOOP);
glVertex2d(315,260);
glVertex2d(345,260);
glVertex2d(345,230);
glVertex2d(315,230);
glEnd();
}else
{
glColor3f(1,1,0);
bitmap_output(150,450,"BLOOK SHOOTING",GLUT_BITMAP_TIMES_ROMAN_24);
counting();
glColor3f(1,1,1);
glPointSize(20);
glBegin(GL_POINTS);
glVertex2d(330,245);
glEnd();
}


if(count8==0)
{
glColor3f(1,1,0);
bitmap_output(150,450,"BLOCK SHOOTING",GLUT_BITMAP_TIMES_ROMAN_24);
```

```
counting();
glPointSize(30);
glColor3f(1,0,1);
glBegin(GL_POINTS);
glVertex2d(200,200);
glEnd();

glColor3f(0,1,1);
glBegin(GL_LINE_LOOP);
glVertex2d(185,215);
glVertex2d(215,215);
glVertex2d(215,185);
glVertex2d(185,185);
glEnd();
}
else
{
glColor3f(1,1,0);
bitmap_output(150,450,"BLOOK SHOOTING",GLUT_BITMAP_TIMES_ROMAN_24);
counting();
glColor3f(1,1,1);
glPointSize(20);
glBegin(GL_POINTS);
glVertex2d(200,200);
glEnd();
}
if(count9==0)
{
glColor3f(1,1,0);
bitmap_output(150,450,"BLOCK SHOOTING",GLUT_BITMAP_TIMES_ROMAN_24);
counting();
glPointSize(30);
glColor3f(1,0,1);
glBegin(GL_POINTS);
glVertex2d(400,150);
glEnd();

glColor3f(0,1,1);
glBegin(GL_LINE_LOOP);
glVertex2d(385,165);
glVertex2d(415,165);
glVertex2d(415,135);
glVertex2d(385,135);
glEnd();
}
else
{
glColor3f(1,1,0);
bitmap_output(150,450,"BLOOK SHOOTING",GLUT_BITMAP_TIMES_ROMAN_24);
counting();
glColor3f(1,1,1);
glPointSize(20);
```

```
glBegin(GL_POINTS);
glVertex2d(400,150);
glEnd();
}
if(count10==0)
{
glColor3f(1,1,0);
bitmap_output(150,450,"BLOCK SHOOTING",GLUT_BITMAP_TIMES_ROMAN_24);
counting();
glPointSize(30);
glColor3f(1,0,1);
glBegin(GL_POINTS);
glVertex2d(300,100);
glEnd();

glColor3f(0,1,1);
glBegin(GL_LINE_LOOP);
glVertex2d(285,115);
glVertex2d(315,115);
glVertex2d(315,85);
glVertex2d(285,85);
glEnd();
}
else
{
glColor3f(1,1,0);
bitmap_output(150,450,"BLOOK SHOOTING",GLUT_BITMAP_TIMES_ROMAN_24);
counting();
glColor3f(1,1,1);
glPointSize(20);
glBegin(GL_POINTS);
glVertex2d(300,100);
glEnd();
}
glColor3f(0,1,1);
glBegin(GL_LINES);
glVertex2d(x,y);
glVertex2d(x=100,y);
glEnd();
glLineWidth(2);
glBegin(GL_LINES);
glVertex2d(x,y+2);
glVertex2d(x+100,y+2);
glEnd();
glBegin(GL_LINES);
glVertex2d(x,y-2);
glVertex2d(x+100,y-2);
glEnd();
glBegin(GL_TRIANGLES);
glVertex2d(x+100,y+3);
glVertex2d(x+110,y);
glVertex2d(x+100,y-3);
```

```
glEnd();
glBegin(GL_QUADS);
glVertex2d(x,y+3);
glVertex2d(x,y-3);
glVertex2d(x-10,y-5);
glVertex2d(x-10,y+5);
glEnd();
glFinish();
glutSwapBuffers();
}


/*TO CHECK WHETHER ARROW HITS TARGET*/
void disa(redraw)
{
if((x+110==300)&&(y>=435&&y<=465))
{
count1=1;
glutIdleFunc(NULL);
glutDisplayFunc(redraw);
}
else if((x+110==375)&&(y>=385&&y<=415))
{
count3=1;
glutIdleFunc(NULL);
glutDisplayFunc(redraw);
}
else if((x+110==399)&&(y>=465&&y<=495))
{
count2=1;
glutIdleFunc(NULL);
glutDisplayFunc(redraw);
}
else if((x+110==249)&&(y>=357&&y<=385))
{
count4=1;
glutIdleFunc(NULL);
glutDisplayFunc(redraw);
}
else if((x+110==351)&&(y>=315&&y<=345))
{
count5=1;
glutIdleFunc(NULL);
glutDisplayFunc(redraw);
}
else if((x+110==450)&&(y>=275&&y<=305))
{
count6=1;
glutIdleFunc(NULL);
glutDisplayFunc(redraw);
}
else if((x+110==330)&&(y>=230&&y<=260))
```

```
{
count7=1;
glutIdleFunc(NULL);
glutDisplayFunc(redraw);
}
else if((x+110==201)&&(y>=185&&y<=215))
{
count8=1;
glutIdleFunc(NULL);
glutDisplayFunc(redraw);
}
else if((x+110==399)&&(y>=135&&y<=165))
{
count9=1;
glutIdleFunc(NULL);
glutDisplayFunc(redraw);
}
else if((x+110==300)&&(y>=85&&y<=115))
{
count10=1;
glutIdleFunc(NULL);
glutDisplayFunc(redraw);
}
}


/*to move arrow up*/
void id()
{
y+=n;
disa();
if(y>maxy)y=0;
glutPostRedisplay();
}


/*to draw the arrow*/
void disp()
{
glClear(GL_COLOR_BUFFER_BIT|GL_DEPTH_BUFFER_BIT);
glLoadIdentity();
glColor3f(1,1,0);
bitmap_output(150,450,"BLOCKSHOOTING",GLUT_BITMAP_TIMES_ROMAN_24);
counting();
glColor3f(0,1,1);
glBegin(GL_LINES);
glVertex2d(x,y);
glVertex2d(x+100,y);
glEnd();
glLineWidth(2);
glBegin(GL_LINES);
glVertex2d(x,y+2);
```

```
glVertex2d(x+100,y+2);
glEnd();
glBegin(GL_LINES);
glVertex2d(x,y-2);
glVertex2d(x+100,y-2);
glEnd();
glBegin(GL_TRIANGLES);
glVertex2d(x+100,y+3);
glVertex2d(x+110,y);
glVertex2d(x+100,y-3);
glEnd();
glBegin(GL_QUADS);
glVertex2d(x,y+3);
glVertex2d(x,y-3);
glVertex2d(x-10,y-5);
glVertex2d(x-10,y+5);
glEnd();
draw_target();
glFlush();
glutSwapBuffers();
}


/*to clear screen & set projection mode*/
void init()
{
glClearColor(0,0,0,1);
glColor3f(1,0,0);
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
gluOrtho2D(0,500,0,500);
glMatrixMode(GL_MODELVIEW);
}


/*to draw the target inside line loop*/
void draw_target()
{
glColor3f(1,0,1);
glPointSize(30);
glBegin(GL_POINTS);
glVertex2d(300,450);
glVertex2d(375,400);
glVertex2d(400,480);
glVertex2d(250,370);
glVertex2d(350,330);
glVertex2d(450,290);
glVertex2d(330,245);
glVertex2d(200,200);

glVertex2d(400,150);
glVertex2d(300,100);
```

```
glEnd();
glColor3f(0,1,1);
glBegin(GL_LINE_LOOP);
glVertex2d(285,465);
glVertex2d(315,465);
glVertex2d(315,435);
glVertex2d(285,435);
glEnd();

glBegin(GL_LINE_LOOP);
glVertex2d(360,415);
glVertex2d(390,415);
glVertex2d(390,385);
glVertex2d(360,385);
glEnd();
glBegin(GL_LINE_LOOP);
glVertex2d(385,495);
glVertex2d(415,495);
glVertex2d(415,465);
glVertex2d(385,465);
glEnd();
glBegin(GL_LINE_LOOP);
glVertex2d(235,385);
glVertex2d(265,385);
glVertex2d(265,355);
glVertex2d(235,355);
glEnd();

glBegin(GL_LINE_LOOP);
glVertex2d(335,345);
glVertex2d(365,345);
glVertex2d(365,315);
glVertex2d(335,315);
glEnd();

glBegin(GL_LINE_LOOP);
glVertex2d(435,305);
glVertex2d(465,305);
glVertex2d(465,275);
glVertex2d(435,275);
glEnd();

glBegin(GL_LINE_LOOP);
glVertex2d(315,260);
glVertex2d(345,260);
glVertex2d(345,230);
glVertex2d(315,230);
glEnd();
glBegin(GL_LINE_LOOP);
glVertex2d(185,215);
glVertex2d(215,215);
glVertex2d(215,185);
```

```
glVertex2d(185,185);
glEnd();
glBegin(GL_LINE_LOOP);
glVertex2d(385,165);
glVertex2d(415,165);
glVertex2d(415,135);
glVertex2d(385,135);
glEnd();
glBegin(GL_LINE_LOOP);
glVertex2d(285,115);
glVertex2d(315,115);
glVertex2d(315,85);
glVertex2d(285,85);
glEnd();
glFlush();
}


/* to move the arrow left wen 'r' pressed*/
void id1()
{
x+=m;
disa();
if(x+110>maxx)
{
x=25;
y=0;
count++;
glutIdleFunc(id);
}
glutPostRedisplay();
}


/*set key to perform desired operation*/
void keys(unsigned char k,int x,int y)
{
if(k=='r')
glutIdleFunc(id1);
}


/*sub menu to display instructions*/
void demo_menu(int i)
{
switch(i)
{
case 5:
case 6:
case 7:
case 8:break;
}
```

```
}


/*sub menu to display designer names*/
void demo(int i)
{
switch(i)
{
case 9:
case 10:
case 11:break;
}
}

void game(int id)
{
     switch(id)
     {

     }
}

/*main to call display,keyboard and idle func*/
int main(int argc,char **argv)
{
int sub_menu,submenu;
glutInit(&argc,argv);
glutInitDisplayMode(GLUT_DOUBLE|GLUT_RGB|GLUT_DEPTH);
glutInitWindowSize(900,900);
glutCreateWindow("ARCHERY GAME BLOCK SHOOTING");
sub_menu=glutCreateMenu(demo_menu);
glutAddMenuEntry("r to move right",5);
glutAddMenuEntry("10arrows and 10 blocks present",6);
glutAddMenuEntry("lost if arrow count exceeds blocks",7);
glutAddMenuEntry("otherwise win",8);
submenu=glutCreateMenu(demo);
glutAddMenuEntry("mamtha",9);
glutAddMenuEntry("priyanka",10);
glutCreateMenu(game);
glutAddSubMenu("INSTRUCTION",sub_menu);
glutAddSubMenu("ABOUT",submenu);
glutAddMenuEntry("QUIT",2);
glutAttachMenu(GLUT_RIGHT_BUTTON);
glutDisplayFunc(disp);
glutIdleFunc(id);
glutKeyboardFunc(keys);
init();
glEnable(GL_DEPTH_TEST);
glutMainLoop();
return 0;
}
```