```
#include<stdio.h>
#include<GL/glut.h>
#include<time.h>
#define MAX(a,b) (((a)>(b))?(a):(b))
char t1[5],t2[5],t3[5],t4[5];
int themeflag=0;
int thememenu;
int foodcount=0;
int foodover=0;
int gameend=0;
int win=0;
int rot=0;
int antirot=0;
int rotcount=0;
int antirotcount=0;
int scale=0;
int scalecount=0;
clock_t start,end;
int clockstart=0;
int startscreen=0;
int gamestart=0;
int gamerestart=0;
int score;
int points;
int timer;
int highscore;
static GLfloat theta[]={0.0,0.0,0.0};
static GLint axis;
GLfloat px=51.0, py=93.0, pz=0.0;
GLfloat lightintensity[]={1.0,1.0,0.0,1.0};
GLfloat lightposition[]={100.0,70.0,50.0,0.0};
glLightfv(GL_LIGHT0,GL_POSITION,lightposition);
glLightfv(GL_LIGHT0,GL_AMBIENT,lightintensity);
glColorMaterial(GL_FRONT,GL_DIFFUSE);
GLfloat amb[]={0.7,0.7,0.7,1.0};
GLfloat diff[]={1.0,1.0,1.0,1.0};
GLfloat spec[]={0.6,0.6,0.6,1.0};
GLfloat shininess[]={80.0};
GLfloat lightintensity[]={1.0,1.0,1.0,1.0};
GLfloat lightposition[]={80.0,80.0,30.0,1.0};

GLfloat foodloc[9][3]={{129,129,0},{147,15,0},{9,81,0},{111,45,0},
                 {129,165,0},{69,69,0},{27,159,0},{87,15,0},{135,135,0}};

void display();

void poly(GLfloat x1,GLfloat y1,GLfloat z1,GLfloat x2,GLfloat y2,GLfloat
z2,GLfloat x3,GLfloat y3,GLfloat z3,GLfloat x4,GLfloat y4,GLfloat
z4,GLfloat x,GLfloat y,GLfloat z)
{
  glBegin(GL_POLYGON);
      glNormal3f(x,y,z);
      glVertex3f(x1,y1,z1);
      glVertex3f(x2,y2,z2);
      glVertex3f(x3,y3,z3);
      glVertex3f(x4,y4,z4);
  glEnd();

}

void output(int x, int y, char *string)
{
      int len, i;

            glRasterPos2f(x,y);
            for (i = 0; i < string[i]!='\0'; i++)
            {
               glutBitmapCharacter(GLUT_BITMAP_TIMES_ROMAN_24,string[i]);
            }
}

void frontscreen(void)
{

      glClear(GL_COLOR_BUFFER_BIT);
      glLoadIdentity();
      glColor3f(1.0, 0.0 1.0);
      output(230,15," Press ENTER to continue");
      output(45,15,"Maximize window for better view");
      output(100,170,"BANGALORE INSTITUTE OF TECHNOLOGY");
      output(110,160,"Dept. of Computer Science and Engineering");
      output(90,140,"COMPUTER GRAPHICS AND VISUALIZATION LAB");
      output(142,125,"Mini Project: BINGE");
      output(170,110,"By :");
      glBegin(GL_LINES);
      glVertex2f(170,108);
      glVertex2f(180,108);
      glEnd();
      output(155,100,"Rakshika Raju");
      output(150,90,"USN:1BI14CS131");
      output(153,70,"Lab In-charges:");
      glBegin(GL_LINES);
      glVertex2f(153,68);
      glVertex2f(203,68);
      glEnd();
      output(116,53,"Prof. N.Thanuja");
      output(190,53,"Prof. Kavitha K");
      output(145,43,"Assistant Professors");
      output(148,33,"Dept. of CSE, BIT");
      glFlush();
}

void cubes(GLfloat t1, GLfloat t2, GLfloat t3, GLfloat s1, GLfloat s2,
GLfloat s3)
{
      glPushMatrix();
      glTranslatef(t1,t2,t3);
      glScaled(s1,s2,s3);
      glutSolidCube(1);
      glPopMatrix();
}

void boundary()
{
      glColor3f(0.000, 0.749, 1.000);
      cubes(-15,90,0,6,168,6);
      cubes(69,9,0,162,6,6);
      cubes(153,90,0,6,168,6);
      cubes(69,171,0,162,6,6);
      if(themeflag==0)
      {
            glColor3f(0.000, 0.749, 1.000);
            cubes(33,129,0,6,6,6);
            cubes(27,90,0,6,84,6);
            cubes(60,51,0,60,6,6);
            cubes(99,51,0,6,6,6);
            cubes(105,90,0,6,84,6);
            cubes(72,129,0,60,6,6);
      }
      else
      {
            glColor3f(0.000, 0.749, 1.000);
            cubes(21,132,0,6,72,6);
            cubes(117,123,0,66,6,6);
            cubes(57,48,0,6,72,6);
      }
```

```
}

void eater()
{
      glColor4f(1.000, 0.271, 0.000,0.8);
      cubes(px,py,pz,6,6,6);
}

void specialfood()
{
   if(foodcount==2)
   {
      glMatrixMode(GL_MODELVIEW);
      glPushMatrix();
      axis=0;
      glTranslatef(foodloc[foodcount][0],foodloc[foodcount][1],foodloc[fo
odcount][2]);
      glRotatef(theta[axis],0.0,1.0,1.0);
      glutSolidCube(6);
      glPopMatrix();

   }
   else if(foodcount==5)
   {
      glMatrixMode(GL_MODELVIEW);
      glPushMatrix();
      axis=1;
      glTranslatef(foodloc[foodcount][0],foodloc[foodcount][1],foodloc[fo
odcount][2]);
      glRotatef(theta[axis],1.0,0.0,1.0);
      glutSolidCube(6);
      glPopMatrix();
   }
   else
   {
      glMatrixMode(GL_MODELVIEW);
      glPushMatrix();
      axis=2;
      glTranslatef(foodloc[foodcount][0],foodloc[foodcount][1],foodloc[fo
odcount][2]);
      glRotatef(theta[axis],1.0,1.0,1.0);
      glutSolidCube(6);
      glPopMatrix();
   }
}

void food()
{
      glColor4f(1.0,1.0,0.0,0.7);
      if(foodover==0)
      {
            if((foodcount!=2)&&(foodcount!=5)&&(foodcount!=8))
            {
                  cubes(foodloc[foodcount][0], foodloc[foodcount][1],
foodloc[foodcount][2],4,4,4);
            }
            else
                  specialfood();
      }
}

void calchighscore()
{
      if(gamerestart==0)
      {
            highscore=score;
      }
      else if(gamerestart==1)
      {
            highscore=MAX(highscore,score);
      }
}

void spinfood()
{
      theta[axis]+=6.0;
      if(theta[axis]>360.0)theta[axis]-=360.0;

      if(clockstart==1)
      {
        end=clock();
        timer=(end-start)/CLOCKS_PER_SEC;
      }
      if(timer!=0)
      {
            if(foodcount!=0)
score=((int)points*foodcount*1000)/((int)timer);
      }
   glutPostRedisplay();
}

void display()
{
      glClear(GL_COLOR_BUFFER_BIT|GL_DEPTH_BUFFER_BIT);
      glClearColor(1.0,1.0,1.0,1.0);
      glMaterialfv(GL_FRONT,GL_AMBIENT,amb);
      glMaterialfv(GL_FRONT,GL_DIFFUSE,diff);
      glMaterialfv(GL_FRONT,GL_SPECULAR,spec);
      glMaterialfv(GL_FRONT,GL_SHININESS,shininess);
      glColorMaterial(GL_FRONT,GL_DIFFUSE);
      glLightfv(GL_LIGHT0,GL_POSITION,lightposition);
      glLightfv(GL_LIGHT0,GL_DIFFUSE,lightintensity);

      if(startscreen==0)
      {
            frontscreen();
      }
      else if(startscreen==1)
      {
            if(scale==0)
            {
                  glScaled(0.5,0.5,0.5);
                  scale=2;
            }
            if(scale==1)
            {
                  glScaled(2.0,2.0,2.0);
                  scale=3;
                  scalecount=1;
            }
            if(scale==2)
            {
                  glColor3f(1.0,0.0,1.0);
                  output(220,135,"INSTRUCTIONS");
                  output(220,110,"1. Use the up/down/left/right keys to 
move towards the food.");
                  output(220,95,"2. You lose if you hit the boundary 
wall.");
                  output(220,80,"3. Eating a food item will fetch you 10 
points.");
                  output(220,65,"4. Every 3rd food item is a special one 
for 30 points.");
                  output(220,50,"5. You can press p to restart the game");

                  output(-10,250,"PRESS ‘s’ TO START THE GAME");
            }
```

```
if(rot==1&&rotcount<1)
{
      glRotatef(-40,1,0,0);
      rot=0;
      ++rotcount;
      --antirotcount;
}
if(antirot==1&&antirotcount<1)
{
      glRotatef(40,1,0,0);
      antirot=0;
      ++antirotcount;
      --rotcount;
}
if(scale==3)
{
      sprintf(t1, "%d", highscore);
      output(180,150,"Highscore:");
      output(250,150,t1);
      sprintf(t2, "%d", timer);
      output(180,140,"Timer");
      output(250,140,t2);
      sprintf(t3, "%d", points);
      output(180,130,"Points");
      output(250,130,t3);
      sprintf(t4, "%d", score);
      output(180,120,"Score");
      output(250,120,t4);
}
if(startscreen==1)
{
      glColor3f(1.0,1.0,0.0);
      glPointSize(10.0);
      food();
      glColor3f(0.0,0.0,1.0);
      glPointSize(18.0);
      eater();
      glColor3f(1.0,1.0,1.0);
      boundary();
      glutPostRedisplay();
}
if(gameend==1)
{
      clockstart=0;
      calchighscore();
      glColor3f(1.0,0.0,0.0);
      output(180,60,"Oops, You Hit the Wall");
      if(score>=highscore)
      {
            glColor3f(0.0,0.0,1.0);
            output(180,50,"But You Made a Highscore!!");
      }
      output(180,40,"Press p to Restart the Game");
}
else if(win==1)
{
      clockstart=0;
      calchighscore();
      glColor3f(1.0,0.0,0.0);
      output(180,60,"You Won");
      if(score>=highscore)
      {
            glColor3f(1.0,0.0,0.0);
            output(180,50,"Hurray!! You Made a Highscore");
      }
      glColor3f(0.0,0.0,1.0);
      output(180,40,"Press p to Restart the Game");
```

```
            }
            glFlush();
              glutSwapBuffers();
      }
}

void themeboundary()
{
      if(themeflag==0)
            {
                  if((px==27 || px==105)&&(py>=51 && py<=129))
                  {gameend=1; foodcount=0;}
                  else if((py==51)&&((px>=33 && px<=87)||(px==99)))
                  {gameend=1; foodcount=0;}
                  else if((py==129)&&((px>=48 && px<=99)||(px==33)))
                  {gameend=1; foodcount=0;}
            }
            else if(themeflag==1)
            {
                  if((px==21)&&((py<=165)&&(py>=99)))
                  {gameend=1; foodcount=0;}
                  else if((py==123)&&((px>=87)&&(px<=148)))
                  {gameend=1; foodcount=0;}
                  else if((px==57)&&((py>=15)&&(py<=81)))
                  {gameend=1; foodcount=0;}
            }
}

void SpecialKey(int key, int x, int y)
{
  switch (key)
  {
      case GLUT_KEY_UP:
            if(win==0 && gameend==0)
            py=py+6;
            glutPostRedisplay();
            if(py==171)
            {gameend=1; foodcount=0;}
            else themeboundary();
            break;

      case GLUT_KEY_DOWN:
            if(win==0 && gameend==0)
            py=py-6;
            glutPostRedisplay();
            if(py==9)
            {gameend=1; foodcount=0;}
            else themeboundary();
            break;

      case GLUT_KEY_LEFT:
            if(win==0 && gameend==0)
            px=px-6;
            glutPostRedisplay();
            if(px==-15)
            {gameend=1; foodcount=0;}
            else themeboundary();
            break;

      case GLUT_KEY_RIGHT:
            if(win==0 && gameend==0)
            px=px+6;
            glutPostRedisplay();
            if(px==153)
            {gameend=1; foodcount=0;}
            else themeboundary();
            break;
}
```

```
if((foodcount<9)&&((px==foodloc[foodcount][0])&&(py==foodloc[foodcount][1
])))
    {
      foodcount++;
      if(foodcount==3||foodcount==6||foodcount==9)
      {
            points+=50;
      }
      else
            points+=10;
    }
    if(foodcount==9) {win=1; foodover=1;}
}

void restart()
{
      gamerestart=1;
      score=0;
      timer=0;
      points=0;
      gamestart=1;
      clockstart=1;
      start=clock();
      foodcount=0;
      foodover=0;
      gameend=0;
      win=0;
      px=51.0; py=93.0; pz=0.0;
      glutPostRedisplay();
}

void keyboard(unsigned char ch,int x, int y)
{
      if(ch=='r') rot=1;
      if(ch=='a')antirot=1;
      if(ch=='s'&& scalecount==0)
      {
            scale=1;
            gamestart=1;
            clockstart=1;
            start=clock();
            glutPostRedisplay();
      }
      if(startscreen==0 && ch==13) {startscreen=1;lighting=1;}
      if(ch=='p')
      {
            restart();
      }
}

void myreshape(int w, int h)
{
      glViewport(0,0,w,h);
      glMatrixMode(GL_PROJECTION);
      glLoadIdentity();
      if(w<=h)
            glOrtho(-25.0,175.0,-2.0*(GLfloat)h/(GLfloat)w,
190.0*(GLfloat)h/(GLfloat)w,-200.0,200.0);
      else
            glOrtho(-25.0*(GLfloat)w/(GLfloat)h,
175.0*(GLfloat)w/(GLfloat)h,-2.0,190.0,-200.0,200.0);
  glMatrixMode(GL_MODELVIEW);
  glutPostRedisplay();
}

void topmenu(int id)
{
        switch(id)
        {
        case 1:rot=1;break;
        case 2:antirot=1;break;
        case 3:themeflag=0;break;
        case 4:themeflag=1;break;
        case 5:restart();break;
        case 6:exit(0);break;
        }
}


int main(int argc,char **argv)
{
        glutInit(&argc,argv);
        glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB|GLUT_DEPTH);
        glutInitWindowSize(600,600);
        glutCreateWindow("Binge");
        glutReshapeFunc(myreshape);
        glutDisplayFunc(display);
        glutIdleFunc(spinfood);
        glutSpecialFunc(SpecialKey);
        glutKeyboardFunc(keyboard);
        glEnable(GL_DEPTH_TEST);
        glEnable(GL_LIGHTING);
        glEnable(GL_LIGHT0);
        glShadeModel(GL_SMOOTH);
        glEnable(GL_NORMALIZE);
        glEnable(GL_COLOR_MATERIAL);
        thememenu=glutCreateMenu(topmenu);
        glutAddMenuEntry("Theme 1",3);
        glutAddMenuEntry("Theme 2",4);
        glutCreateMenu(topmenu);
        glutAddSubMenu("Themes",thememenu);
        glutAddMenuEntry("Rotate Forwards",1);
        glutAddMenuEntry("Rotate Backwards",2);
        glutAddMenuEntry("Restart",5);
        glutAddMenuEntry("Quit",6);
        glutAttachMenu(GLUT_RIGHT_BUTTON);
        glutMainLoop();
        return 0;
}
```