CGV-THIRSTY CROW

```
#include <windows.h>
#include <GL/gl.h>
#include <GL/glu.h>
#include <GL/glut.h>
#include <cstdio>
#include<iostream>
#include<math.h>
#include<MMSystem.h>

# define PI 3.14159265358979323846

void Display(void);
void Display1();

GLfloat position = -1.2f, position2 = 0.0f, skypos = -1.2f, xpos = -1.0f, ypos = 0.45f, yuppos, ydown;
GLfloat speed = 0.08f, skyspeed = 0.03f, crowspeed = 0.5f;


int donef = 0, dtwof = 0, dthreef = 0, dfourf = 0, dfivef = 0, dsixf = 0, dsevenf = 0,
waterflag = 0, ideaflag = 0, ideamusic = 0, febbleflag = 0, stonereturnflag = 0,
deightf = 0, flyaway = 0, drankwater = 0, happilygone = 0;

void update(int value) {

   if (position > 1.3)
   {
      position = -1.2f;
      skypos = -1.2f;
      position2 = 1.2f;
   }

   position += speed;
   position2 -= 0.05;
```

```
    skypos += skyspeed;

    if (!waterflag)
    {
        xpos += crowspeed;
        ypos -= crowspeed;
    }
    if (stonereturnflag)
    {
        xpos += crowspeed;
        yuppos += crowspeed;
    }
    ydown -= speed;
    glutPostRedisplay();

    glutTimerFunc(100, update, 0);
}

void StartingText()
{
    char text[120]="THIRSTY CROW";
    glColor3f(255, 0, 0);
    glRasterPos2f(-28, 32);
    for (int i = 0; text[i] != '\0'; i++)
    {
        if (text[i] == ' ' && text[i + 1] == ' ')
        {
            glutBitmapCharacter(GLUT_BITMAP_TIMES_ROMAN_24, text[i]);
            glRasterPos2f(-20, 22);
        }
        else glutBitmapCharacter(GLUT_BITMAP_TIMES_ROMAN_24, text[i]);
    }
```

```
   char moral[120]= "IF YOU ARE DETERMINED ENOUGH,YOU CAN FIND A WAY TO ACHEIEVE WHAT YOU WANT  EVEN IF IT IS VERY DIFFICULT";
   glColor3f(0, 0, 250);
   glRasterPos2f(-100, 0);
   for (int i = 0; moral[i] != '\0'; i++)
   {
      if (moral[i] == ' ' && moral[i + 1] == ' ')
      {
         glutBitmapCharacter(GLUT_BITMAP_TIMES_ROMAN_24, moral[i]);
         glRasterPos2f(-40, -10);
      }
      else glutBitmapCharacter(GLUT_BITMAP_TIMES_ROMAN_24, moral[i]);
   }
   char press[50]="Press 1,2,3,4,5,6,7 to continue the story";
   glColor3f(0, 0, 0);
   glRasterPos2f(-42, -30);
   for (int i = 0; press[i] != '\0'; i++)
   {
      if (press[i] == ' ' && press[i + 1] == ' ')
      {
         glutBitmapCharacter(GLUT_BITMAP_TIMES_ROMAN_24, press[i]);
         glRasterPos2f(-40, -10);
      }
      else glutBitmapCharacter(GLUT_BITMAP_TIMES_ROMAN_24, press[i]);
   }
}

void Home()
{
   glBegin(GL_POLYGON);
   glColor3ub(244, 164, 96);
   glVertex2f(-0.3f, 0.0f);
   glVertex2f(0.1f, 0.0f);
   glVertex2f(0.02f, 0.3f);
```

```
glVertex2f(-0.38f, 0.3f);
glEnd();

glBegin(GL_POLYGON);
glColor3ub(244, 164, 96);
glVertex2f(-0.29f, 0.0f);
glVertex2f(0.07f, 0.0f);
glVertex2f(0.07f, -0.4f);
glVertex2f(-0.29f, -0.4f);
glEnd();

glBegin(GL_POLYGON);
glColor3ub(244, 164, 96);
glVertex2f(-0.38f, 0.3f);
glVertex2f(-0.43f, 0.0f);
glVertex2f(-0.41f, 0.0f);
glVertex2f(-0.41f, -0.35f);
glVertex2f(-0.29f, -0.4f);
glVertex2f(0.07f, -0.4f);

glEnd();

glBegin(GL_LINES);
glColor3ub(0, 0, 0);
glVertex2f(-0.3f, 0.0f);
glVertex2f(0.1f, 0.0f);

glColor3ub(0, 0, 0);
glVertex2f(-0.29f, 0.0f);
glVertex2f(-0.29f, -0.4f);

glColor3ub(0, 0, 0);
glVertex2f(0.07f, -0.4f);
glVertex2f(-0.29f, -0.4f);
```

```
glColor3ub(0, 0, 0);
glVertex2f(0.07f, -0.4f);
glVertex2f(0.07f, -0.0f);

glColor3ub(0, 0, 0);
glVertex2f(-0.3f, 0.0f);
glVertex2f(-0.38f, 0.3f);

glColor3ub(0, 0, 0);
glVertex2f(-0.38f, 0.3f);
glVertex2f(-0.43f, 0.0f);

glColor3ub(0, 0, 0);
glVertex2f(-0.37f, 0.245f);
glVertex2f(-0.41f, 0.0f);

glColor3ub(0, 0, 0);
glVertex2f(-0.41f, 0.0f);
glVertex2f(-0.41f, -0.35f);

glColor3ub(0, 0, 0);
glVertex2f(-0.41f, -0.35f);
glVertex2f(-0.29f, -0.4f);

glColor3ub(0, 0, 0);
glVertex2f(0.1f, 0.0f);
glVertex2f(0.02f, 0.3f);

glColor3ub(0, 0, 0);
glVertex2f(0.02f, 0.3f);
glVertex2f(-0.38f, 0.3f);

glEnd();
```

```
    glBegin(GL_POLYGON);
    glColor3ub(23, 14, 9);
    glVertex2f(-0.15f, -0.1f);
    glVertex2f(-0.02f, -0.1f);
    glVertex2f(-0.02f, -0.4f);
    glVertex2f(-0.15f, -0.4f);
    glEnd();

    glBegin(GL_POLYGON);
    glColor3ub(23, 14, 9);
    glVertex2f(-0.24f, -0.05f);
    glVertex2f(-0.18f, -0.05f);
    glVertex2f(-0.18f, -0.15f);
    glVertex2f(-0.24f, -0.15f);
    glEnd();

    glBegin(GL_POLYGON);
    glColor3ub(23, 14, 9);
    glVertex2f(-0.31f, 0.0f);
    glVertex2f(-0.38f, 0.02f);
    glVertex2f(-0.38f, -0.37f);
    glVertex2f(-0.31f, -0.4f);
    glEnd();

    glBegin(GL_POLYGON);
    glColor3ub(23, 14, 9);
    glVertex2f(0.06f, -0.05f);
    glVertex2f(-0.0f, -0.05f);
    glVertex2f(-0.0f, -0.15f);
    glVertex2f(0.06f, -0.15f);
    glEnd();
}
```

```
void sun()
{
   int triangleAmount = 20;
   GLfloat k = -.7f; GLfloat l = .8f;
   GLfloat radius = .10f;

   GLfloat twicePi = 2.0f * PI;

   glBegin(GL_TRIANGLE_FAN);
   glColor3ub(255, 255, 0);
   glVertex2f(k, l); // center of circle
   for (int i = 0; i <= triangleAmount; i++) {
      glVertex2f(
         k + (radius * cos(i * twicePi / triangleAmount)),
         l + (radius * sin(i * twicePi / triangleAmount))
      );
   }
   glEnd();
}

void Moving_Stone()
{
   int triangleAmount = 20;
   GLfloat radius = .02f;

   GLfloat twicePi = 2.0f * PI;

   GLfloat m = .6f; GLfloat n = 0.0f;
   glBegin(GL_TRIANGLE_FAN);
   glColor3ub(107, 101, 93);
   glVertex2f(m, n); // center of circle
   for (int i = 0; i <= triangleAmount; i++) {
      glVertex2f(
         m + (radius * cos(i * twicePi / triangleAmount)),
```

```
            n + (radius * sin(i * twicePi / triangleAmount))
        );
    }
    glEnd();
}

void Bird()
{
    glBegin(GL_POLYGON);
    glColor3ub(0, 0, 0);

    glVertex2f(-0.5, 0.15);
    glVertex2f(-0.3, 0.25);
    glVertex2f(-0.1, 0.15);
    glVertex2f(-0.3, 0.09);

    glEnd();

    //WING ONE

    glBegin(GL_POLYGON);
    glVertex2f(-0.4, 0.2);
    glVertex2f(-0.4, 0.25);
    glVertex2f(-0.25, 0.35);
    glVertex2f(-0.3, 0.28);
    glVertex2f(-0.35, 0.2);
    glEnd();

    //WING TWO

    glBegin(GL_POLYGON);
    glVertex2f(-0.4, 0.20);
    glVertex2f(-0.2, 0.31);
    glVertex2f(-0.1, 0.22);
```

```
glEnd();

//LIP

glBegin(GL_LINES);
glColor3ub(0, 0, 0);
glVertex2f(-0.55, 0.1);
glVertex2f(-0.49, 0.15);
glVertex2f(-0.55, 0.1);
glVertex2f(-0.48, 0.14);
glVertex2f(-0.48, 0.14);
glVertex2f(-0.5, 0.1);
glVertex2f(-0.5, 0.1);
glVertex2f(-0.45, 0.15);
glEnd();

//TAIL

glLineWidth(2);
glBegin(GL_LINES);
glVertex2f(-0.15, 0.15);
glVertex2f(-0.08, 0.15);
glEnd();

//EYE

glPointSize(25.0);
glTranslatef(-0.45f, 0.18f, 0);
glBegin(GL_POINTS);
glColor3ub(0, 0, 0);
glVertex2f(-0.0f, -0.0f);
glEnd();
```

```
int triangleAmount = 20;
GLfloat k = -.009f; GLfloat l = .0f;
GLfloat radius = .01f;

GLfloat twicePi = 2.0f * PI;

glBegin(GL_TRIANGLE_FAN);
glColor3ub(255, 255, 255);
glVertex2f(k, l); // center of circle
for (int i = 0; i <= triangleAmount; i++) {
   glVertex2f(
      k + (radius * cos(i * twicePi / triangleAmount)),
      l + (radius * sin(i * twicePi / triangleAmount))
   );
}
glEnd();



if (febbleflag)
{
   radius = .02f;

   twicePi = 2.0f * PI;

   GLfloat m = -.075f; GLfloat n = -0.075f;
   glBegin(GL_TRIANGLE_FAN);
   glColor3ub(107, 101, 93);
   glVertex2f(m, n); // center of circle
   for (int i = 0; i <= triangleAmount; i++) {
      glVertex2f(
         m + (radius * cos(i * twicePi / triangleAmount)),
         n + (radius * sin(i * twicePi / triangleAmount))
      );
   }
```

```
        glEnd();

    }

}

void fullsky()
{
    glLoadIdentity();
    glBegin(GL_POLYGON);
    glColor3ub(155, 215, 232);
    glVertex2f(-1.0f, 1.0f);
    glVertex2f(1.0f, 1.0f);
    glVertex2f(1.0f, 0.1f);
    glVertex2f(-1.0f, 0.1f);

    glEnd();
}

void road()
{
    glBegin(GL_POLYGON);
    glColor3ub(54, 15, 0);
    glVertex2f(-1.0f, -0.85f);
    glVertex2f(1.0f, -0.7f);
    glVertex2f(1.0f, -1.0);
    glVertex2f(-1.0f, -1.0);
    glEnd();
}



void pitcher()
{
    int triangleAmount = 20;
```

```
GLfloat k = .4f; GLfloat l = -.7f;
GLfloat radius = .13f;

GLfloat twicePi = 2.0f * PI;

glBegin(GL_TRIANGLE_FAN);
glColor3ub(107, 101, 93);
glVertex2f(k, l); // center of circle
for (int i = 0; i <= triangleAmount; i++) {
   glVertex2f(
      k + (radius * cos(i * twicePi / triangleAmount)),
      l + (radius * sin(i * twicePi / triangleAmount))
   );
}
glEnd();


glBegin(GL_POLYGON);
glColor3ub(107, 101, 93);
glVertex2f(0.47f, -0.65f);
glVertex2f(0.32f, -0.65f);
glVertex2f(0.32f, -0.55f);
glVertex2f(0.47f, -0.55f);

glEnd();



glBegin(GL_POLYGON);
glColor3ub(191, 180, 164);
glVertex2f(0.47f, -0.55f);
glVertex2f(0.32f, -0.55f);
glVertex2f(0.30f, -0.52f);
glVertex2f(0.50f, -0.52f);
```

```
    glEnd();

}

void tree()
{
    int triangleAmount = 20;
    GLfloat k = .7f; GLfloat l = -.1f;
    GLfloat radius = .25f;

    GLfloat twicePi = 2.0f * PI;

    glBegin(GL_TRIANGLE_FAN);
    glColor3ub(34, 139, 34);
    glVertex2f(k, l); // center of circle
    for (int i = 0; i <= triangleAmount; i++) {
        glVertex2f(
            k + (radius * cos(i * twicePi / triangleAmount)),
            l + (radius * sin(i * twicePi / triangleAmount))
        );
    }
    glEnd();

    GLfloat m = .8f; GLfloat n = -.0f;
    glBegin(GL_TRIANGLE_FAN);
    glColor3ub(34, 139, 34);
    glVertex2f(m, n); // center of circle
    for (int i = 0; i <= triangleAmount; i++) {
        glVertex2f(
            m + (radius * cos(i * twicePi / triangleAmount)),
            n + (radius * sin(i * twicePi / triangleAmount))
        );
    }
```

```
glEnd();

GLfloat o = .64f; GLfloat p = .12f;
glBegin(GL_TRIANGLE_FAN);
glColor3ub(34, 139, 34);
glVertex2f(o, p); // center of circle
for (int i = 0; i <= triangleAmount; i++) {
   glVertex2f(
      o + (radius * cos(i * twicePi / triangleAmount)),
      p + (radius * sin(i * twicePi / triangleAmount))
   );
}
glEnd();

GLfloat q = .8f; GLfloat r = .3f;
glBegin(GL_TRIANGLE_FAN);
glColor3ub(34, 139, 34);
glVertex2f(q, r); // center of circle
for (int i = 0; i <= triangleAmount; i++) {
   glVertex2f(
      q + (radius * cos(i * twicePi / triangleAmount)),
      r + (radius * sin(i * twicePi / triangleAmount))
   );
}
glEnd();



glBegin(GL_POLYGON);
glColor3ub(83, 53, 10.);
glVertex2f(0.87f, -0.3f);
glVertex2f(0.85f, 0.0f);
glVertex2f(1.0f, 0.0f);
glVertex2f(1.0f, -0.85f);
glVertex2f(0.85f, -0.85f);
```

```
glEnd();

glBegin(GL_POLYGON);
glColor3ub(83, 53, 10.);
glVertex2f(0.87f, -0.3f);
glVertex2f(0.70f, -0.2f);
glVertex2f(0.70f, -0.15f);
glVertex2f(0.8f, -0.17f);
glVertex2f(0.87f, -0.2f);
glEnd();

glBegin(GL_POLYGON);
glColor3ub(83, 53, 10.);
glVertex2f(0.70f, -0.2f);
glVertex2f(0.70f, -0.15f);
glVertex2f(0.50f, -0.17f);
glVertex2f(0.50f, -0.2f);
glEnd();

glBegin(GL_POLYGON);
glColor3ub(83, 53, 10.);
glVertex2f(0.9f, -0.03f);
glVertex2f(0.60f, 0.05f);
glVertex2f(0.64f, 0.09f);
glVertex2f(0.9f, 0.05f);
glVertex2f(0.95f, -0.03f);
glEnd();

glBegin(GL_POLYGON);
glColor3ub(83, 53, 10.);
glVertex2f(0.75f, 0.07f);
glVertex2f(0.70f, 0.13f);
glVertex2f(0.58f, 0.20f);
glVertex2f(0.65f, 0.17f);
```

```
    glVertex2f(0.79f, 0.09f);
    glVertex2f(0.85f, 0.0f);
    glEnd();

    glBegin(GL_POLYGON);
    glColor3ub(83, 53, 10.);
    glVertex2f(.95f, 0.0f);
    glVertex2f(.92f, 0.0f);
    glVertex2f(.85f, 0.29f);
    glVertex2f(.90f, 0.27f);
    glVertex2f(.91f, 0.24f);
    glVertex2f(.99f, 0.0f);
    //glVertex2f();


    glBegin(GL_POLYGON);
    glColor3ub(83, 53, 10.);
    glVertex2f(.96f, 0.2f);
    glVertex2f(.97f, 0.25f);
    glVertex2f(.99f, 0.30f);
    glVertex2f(.96f, 0.27f);
    glVertex2f(.97f, 0.24f);
    glVertex2f(.96f, 0.25f);
    glEnd();
}

void waterdrop()
{
    glBegin(GL_POLYGON);
    glColor3ub(135, 206, 250);
    glVertex2f(0.16f, -0.72f);
    glVertex2f(0.11f, -0.70f);
    glVertex2f(.08f, -0.67f);
    glVertex2f(0.11f, -0.64f);
```

```
    glVertex2f(0.13f, -0.64f);
    glVertex2f(0.15f, -0.67f);
    glVertex2f(0.16f, -0.70f);
    glVertex2f(0.17f, -0.72f);
    glEnd();

    glBegin(GL_POLYGON);
    glColor3ub(135, 206, 250);
    glVertex2f(0.20f, -0.62f);
    glVertex2f(0.11f, -0.60f);
    glVertex2f(.08f, -0.57f);
    glVertex2f(0.11f, -0.54f);
    glVertex2f(0.16f, -0.54f);
    glVertex2f(0.18f, -0.57f);
    glVertex2f(0.18f, -0.60f);
    glEnd();

    glLoadIdentity();

    glBegin(GL_POLYGON);
    glColor3ub(135, 206, 250);
    glVertex2f(0.47f, -0.58f);
    glVertex2f(0.32f, -0.58f);
    glVertex2f(0.32f, -0.55f);
    glVertex2f(0.47f, -0.55f);

    glEnd();

}

void stone()
{
    int triangleAmount = 20;
    GLfloat k = -.7f; GLfloat l = -.7f;
```

```
GLfloat radius = .02f;

GLfloat twicePi = 2.0f * PI;

glBegin(GL_TRIANGLE_FAN);
glColor3ub(107, 101, 93);
glVertex2f(k, l); // center of circle
for (int i = 0; i <= triangleAmount; i++) {
   glVertex2f(
      k + (radius * cos(i * twicePi / triangleAmount)),
      l + (radius * sin(i * twicePi / triangleAmount))
   );
}
glEnd();

GLfloat m = -.67f; GLfloat n = -.75f;

glBegin(GL_TRIANGLE_FAN);
glColor3ub(107, 101, 93);
glVertex2f(m, n); // center of circle
for (int i = 0; i <= triangleAmount; i++) {
   glVertex2f(
      m + (radius * cos(i * twicePi / triangleAmount)),
      n + (radius * sin(i * twicePi / triangleAmount))
   );
}
glEnd();

GLfloat c = -.67f; GLfloat d = -.67f;

glBegin(GL_TRIANGLE_FAN);
glColor3ub(107, 101, 93);
glVertex2f(c, d); // center of circle
for (int i = 0; i <= triangleAmount; i++) {
```

```
        glVertex2f(
            c + (radius * cos(i * twicePi / triangleAmount)),
            d + (radius * sin(i * twicePi / triangleAmount))
        );
    }
    glEnd();

    GLfloat e = -.63f; GLfloat f = -.7f;

    glBegin(GL_TRIANGLE_FAN);
    glColor3ub(107, 101, 93);
    glVertex2f(e, f); // center of circle
    for (int i = 0; i <= triangleAmount; i++) {
        glVertex2f(
            e + (radius * cos(i * twicePi / triangleAmount)),
            f + (radius * sin(i * twicePi / triangleAmount))
        );
    }
    glEnd();

    GLfloat x = -.66f; GLfloat y = -.70f;

    glBegin(GL_TRIANGLE_FAN);
    glColor3ub(107, 101, 93);
    glVertex2f(x, y); // center of circle
    for (int i = 0; i <= triangleAmount; i++) {
        glVertex2f(
            x + (radius * cos(i * twicePi / triangleAmount)),
            y + (radius * sin(i * twicePi / triangleAmount))
        );
    }
    glEnd();
}
```

```
void grass()
{
   glBegin(GL_POLYGON);
   glColor3ub(124, 252, 0);
   glVertex2f(-1.0f, 0.1f);
   glVertex2f(1.0f, .1f);
   glVertex2f(1.0f, -0.7f);
   glVertex2f(-1.0f, -0.85f);

   glEnd();
}

void background()
{
   glBegin(GL_POLYGON);
   glColor3ub(1, 132, 42);
   glVertex2f(-1.0f, 0.1f);
   glVertex2f(-0.95f, .15f);
   glVertex2f(-.93f, 0.15f);
   glVertex2f(-0.9f, 0.1f);

   glEnd();

   glBegin(GL_POLYGON);
   glColor3ub(1, 132, 42);
   glVertex2f(-0.93f, 0.1f);
   glVertex2f(-0.83f, .25f);
   glVertex2f(-.79f, 0.255f);
   glVertex2f(-0.74f, 0.20f);
   glVertex2f(-0.70f, 0.1f);

   glEnd();
```

```
glBegin(GL_POLYGON);
glColor3ub(1, 132, 42);
glVertex2f(-0.73f, 0.1f);
glVertex2f(-0.73f, 0.15f);
glVertex2f(-0.7f, .20f);
glVertex2f(-.65f, 0.18f);
glVertex2f(-0.60f, 0.15f);
glVertex2f(-0.55f, 0.1f);

glEnd();

glBegin(GL_POLYGON);
glColor3ub(1, 132, 42);
glVertex2f(-.57f, 0.1f);
glVertex2f(-0.52f, .15f);
glVertex2f(-.50f, 0.15f);
glVertex2f(-0.47f, 0.1f);

glEnd();

glBegin(GL_POLYGON);
glColor3ub(1, 132, 42);
glVertex2f(-.50f, 0.1f);
glVertex2f(-0.45f, .15f);
glVertex2f(-.43f, 0.15f);
glVertex2f(-0.40f, 0.1f);

glEnd();

glBegin(GL_POLYGON);
glColor3ub(1, 132, 42);
glVertex2f(-0.43f, 0.1f);
glVertex2f(-0.33f, .25f);
```

```
glVertex2f(-.29f, 0.255f);
glVertex2f(-0.24f, 0.20f);
glVertex2f(-0.20f, 0.1f);

glEnd();

glBegin(GL_POLYGON);
glColor3ub(1, 132, 42);
glVertex2f(-0.23f, 0.1f);
glVertex2f(-0.23f, 0.15f);
glVertex2f(-0.20f, .20f);
glVertex2f(-.15f, 0.18f);
glVertex2f(-0.10f, 0.15f);
glVertex2f(-0.05f, 0.1f);

glEnd();

glBegin(GL_POLYGON);
glColor3ub(1, 132, 42);
glVertex2f(-0.08f, 0.1f);
glVertex2f(.08f, .25f);
glVertex2f(.12f, 0.255f);
glVertex2f(0.17f, 0.20f);
glVertex2f(0.22f, 0.1f);

glEnd();

glBegin(GL_POLYGON);
glColor3ub(1, 132, 42);
glVertex2f(.21f, 0.1f);
glVertex2f(0.28f, .15f);
glVertex2f(.30f, 0.15f);
glVertex2f(0.33f, 0.1f);
```

```
    glEnd();
}

void sky1()
{
    // glLoadIdentity();
    int i;

    GLfloat x = .5f; GLfloat y = .8f; GLfloat radius = .05f;
    int triangleAmount = 20;
    GLfloat twicePi = 2.0f * PI;

    glBegin(GL_TRIANGLE_FAN);
    glColor3ub(0, 0, 250);
    glVertex2f(x, y); // center of circle
    for (i = 0; i <= triangleAmount; i++) {
        glVertex2f(
            x + (radius * cos(i * twicePi / triangleAmount)),
            y + (radius * sin(i * twicePi / triangleAmount))
        );
    }
    glEnd();

    GLfloat a = .55f; GLfloat b = .78f;

    glBegin(GL_TRIANGLE_FAN);
    glColor3ub(0, 0, 250);
    glVertex2f(a, b); // center of circle
    for (i = 0; i <= triangleAmount; i++) {
        glVertex2f(
            a + (radius * cos(i * twicePi / triangleAmount)),
            b + (radius * sin(i * twicePi / triangleAmount))
        );
    }
```

```
glEnd();

GLfloat c = .45f; GLfloat d = .78f;

glBegin(GL_TRIANGLE_FAN);
glColor3ub(0, 0, 250);
glVertex2f(c, d); // center of circle
for (i = 0; i <= triangleAmount; i++) {
   glVertex2f(
      c + (radius * cos(i * twicePi / triangleAmount)),
      d + (radius * sin(i * twicePi / triangleAmount))
   );
}
glEnd();

GLfloat e = .52f; GLfloat f = .75f;

glBegin(GL_TRIANGLE_FAN);
glColor3ub(0, 0, 250);
glVertex2f(e, f); // center of circle
for (i = 0; i <= triangleAmount; i++) {
   glVertex2f(
      e + (radius * cos(i * twicePi / triangleAmount)),
      f + (radius * sin(i * twicePi / triangleAmount))
   );
}
glEnd();

GLfloat g = .6f; GLfloat h = .77f;

glBegin(GL_TRIANGLE_FAN);
glColor3ub(0, 0, 250);
glVertex2f(g, h); // center of circle
for (i = 0; i <= triangleAmount; i++) {
```

```
        glVertex2f(
            g + (radius * cos(i * twicePi / triangleAmount)),
            h + (radius * sin(i * twicePi / triangleAmount))
        );
    }
    glEnd();



}

void sky2()
{
    // glLoadIdentity();
    int i;

    GLfloat x = -.5f; GLfloat y = .8f; GLfloat radius = .05f;
    int triangleAmount = 20;
    GLfloat twicePi = 2.0f * PI;

    glBegin(GL_TRIANGLE_FAN);
    glColor3ub(0, 0, 250);
    glVertex2f(x, y); // center of circle
    for (i = 0; i <= triangleAmount; i++) {
        glVertex2f(
            x + (radius * cos(i * twicePi / triangleAmount)),
            y + (radius * sin(i * twicePi / triangleAmount))
        );
    }
    glEnd();

    GLfloat a = -.55f; GLfloat b = .78f;

    glBegin(GL_TRIANGLE_FAN);
    glColor3ub(0, 0, 250);
```

```
glVertex2f(a, b); // center of circle
for (i = 0; i <= triangleAmount; i++) {
   glVertex2f(
      a + (radius * cos(i * twicePi / triangleAmount)),
      b + (radius * sin(i * twicePi / triangleAmount))
   );
}
glEnd();

GLfloat c = -.45f; GLfloat d = .78f;

glBegin(GL_TRIANGLE_FAN);
glColor3ub(0, 0, 250);
glVertex2f(c, d); // center of circle
for (i = 0; i <= triangleAmount; i++) {
   glVertex2f(
      c + (radius * cos(i * twicePi / triangleAmount)),
      d + (radius * sin(i * twicePi / triangleAmount))
   );
}
glEnd();

GLfloat e = -.52f; GLfloat f = .75f;

glBegin(GL_TRIANGLE_FAN);
glColor3ub(0, 0, 250);
glVertex2f(e, f); // center of circle
for (i = 0; i <= triangleAmount; i++) {
   glVertex2f(
      e + (radius * cos(i * twicePi / triangleAmount)),
      f + (radius * sin(i * twicePi / triangleAmount))
   );
}
glEnd();
```

```
    GLfloat g = -.6f; GLfloat h = .77f;

    glBegin(GL_TRIANGLE_FAN);
    glColor3ub(0, 0, 250);
    glVertex2f(g, h); // center of circle
    for (i = 0; i <= triangleAmount; i++) {
        glVertex2f(
            g + (radius * cos(i * twicePi / triangleAmount)),
            h + (radius * sin(i * twicePi / triangleAmount))
        );
    }
    glEnd();
}

void Normal_Tree()
{
    glBegin(GL_POLYGON);
    glColor3ub(83, 53, 10);
    glVertex2f(-0.62f, -0.24f);
    glVertex2f(-0.58f, -0.24f);
    glVertex2f(-0.58f, -0.8f);
    glVertex2f(-0.62f, -0.8f);
    glEnd();

    int triangleAmount = 20;
    GLfloat k = -.67f; GLfloat l = -.11f;
    GLfloat radius = .15f;

    GLfloat twicePi = 2.0f * PI;

    glBegin(GL_TRIANGLE_FAN);
    glColor3ub(34, 139, 34);
    glVertex2f(k, l); // center of circle
```

```
for (int i = 0; i <= triangleAmount; i++) {
   glVertex2f(
      k + (radius * cos(i * twicePi / triangleAmount)),
      l + (radius * sin(i * twicePi / triangleAmount))
   );
}
glEnd();

GLfloat m = -0.7f; GLfloat n = 0.1f;
glBegin(GL_TRIANGLE_FAN);
glColor3ub(34, 139, 34);
glVertex2f(m, n); // center of circle
for (int i = 0; i <= triangleAmount; i++) {
   glVertex2f(
      m + (radius * cos(i * twicePi / triangleAmount)),
      n + (radius * sin(i * twicePi / triangleAmount))
   );
}
glEnd();

GLfloat o = -.59f; GLfloat p = .23f;
glBegin(GL_TRIANGLE_FAN);
glColor3ub(34, 139, 34);
glVertex2f(o, p); // center of circle
for (int i = 0; i <= triangleAmount; i++) {
   glVertex2f(
      o + (radius * cos(i * twicePi / triangleAmount)),
      p + (radius * sin(i * twicePi / triangleAmount))
   );
}
glEnd();
radius = .18f;
GLfloat q = -.5f; GLfloat r = 0.05f;
glBegin(GL_TRIANGLE_FAN);
```

```
    glColor3ub(34, 139, 34);
    glVertex2f(q, r); // center of circle
    for (int i = 0; i <= triangleAmount; i++) {
        glVertex2f(
            q + (radius * cos(i * twicePi / triangleAmount)),
            r + (radius * sin(i * twicePi / triangleAmount))
        );
    }
    glEnd();

    radius = 0.15f;
    GLfloat qq = -.53f; GLfloat rr = -0.12f;
    glBegin(GL_TRIANGLE_FAN);
    glColor3ub(34, 139, 34);
    glVertex2f(qq, rr); // center of circle
    for (int i = 0; i <= triangleAmount; i++) {
        glVertex2f(
            qq + (radius * cos(i * twicePi / triangleAmount)),
            rr + (radius * sin(i * twicePi / triangleAmount))
        );
    }
    glEnd();
}



void reshape(int w, int h)
{
    std::cout << "Reshape is called" << std::endl;
    float aspectRatio = (float)w / (float)h;
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluPerspective(145, aspectRatio, 1.0, 100.0);
    glMatrixMode(GL_MODELVIEW);
```

```
}

void Display(void)
{
   //std::cout<<"Display 1 called"<<std::endl;
   glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
   glLoadIdentity();
   glTranslatef(0, 0, -20);
   StartingText();
   glFlush();
   glutSwapBuffers();
}

void init(void)
{
   glClearColor(1.0f, 1.0f, 1.0f, 1.0f);
   glClearDepth(1.0);
   glEnable(GL_DEPTH_TEST);
   glEnable(GL_LIGHTING);
   glShadeModel(GL_SMOOTH);
   glEnable(GL_COLOR_MATERIAL);
   glColorMaterial(GL_FRONT, GL_AMBIENT_AND_DIFFUSE);
   glEnable(GL_LIGHT0);
   std::cout << "Init is called" << std::endl;
}

void Display8()
{
   std::cout << "Display 8 called" << std::endl;
   glClearColor(1.0f, 1.0f, 1.0f, 1.0f); // Set background color to black and opaque
   glClear(GL_COLOR_BUFFER_BIT);

   road();
```

```
fullsky();
grass();
background();
pitcher();
sky1();
sky2();
glTranslatef(0.1f, 0.0f, 0.0f);
waterdrop();
glLoadIdentity();
tree();

glLoadIdentity();
glTranslatef(-0.17, 0.0, 0.0);
sun();
glLoadIdentity();
stone();

glPushMatrix();
if (!flyaway)
{
   glTranslatef(-0.2f, ydown, 0.0f);
   std::cout << xpos << " " << yuppos << std::endl;
   glRotatef(180, 0, 1, 0);
   Bird();
}
else
{
   glTranslatef(xpos - 0.2, yuppos - 0.4, 0.0f);
   glRotatef(180, 0, 1, 0);
   Bird();
   std::cout << "Xpos=" << xpos << std::endl;
   if (drankwater == 0)
   {
      PlaySound(TEXT("Drank Water.wav"), NULL, SND_SYNC);
```

```
            drankwater = 1;
        }
        if (xpos > 1.4 && happilygone == 0)
        {
            PlaySound(TEXT("Happily gone.wav"), NULL, SND_SYNC);
            happilygone = 1;
        }

    }
    glPopMatrix();
    glFlush();
    if (ydown < -0.6 && deightf == 0)
    {
        PlaySound(TEXT("Water came up.wav"), NULL, SND_SYNC);
        deightf = 1;
        flyaway = 1;
        stonereturnflag = 1;
    }
}

void Display7()
{
    glClearColor(1.0f, 1.0f, 1.0f, 1.0f); // Set background color to black and opaque
    glClear(GL_COLOR_BUFFER_BIT);

    road();
    fullsky();
    grass();
    background();
    pitcher();
    sky1();
    sky2();
    glLoadIdentity();
    tree();
```

```
glLoadIdentity();
glTranslatef(-0.17, 0.0, 0.0);
sun();
glLoadIdentity();
stone();

glPushMatrix();
glTranslatef(xpos, yuppos, 0.0f);
std::cout << xpos << " " << yuppos << std::endl;
glRotatef(180, 0, 1, 0);
Bird();
glPopMatrix();
if (!febbleflag)
{
   glPushMatrix();
   glTranslatef(-0.2, ydown, 0.0f);
   Moving_Stone();
   glPopMatrix();
   if (ydown < -0.7)
   {
      ydown = 0.8;
      glutDisplayFunc(Display8);

   }
}

glFlush();
if ((xpos >= -0.2 && yuppos >= -0.25) && dsevenf == 0)
{
   waterflag = 1;
   stonereturnflag = 0;
   PlaySound(TEXT("Drop pebble into pot.wav"), NULL, SND_SYNC);
   dsevenf = 1;
```

```
        febbleflag = 0;
        ydown = 0.2;
    }
}


void Display6()
{
    glClearColor(1.0f, 1.0f, 1.0f, 1.0f); // Set background color to black and opaque
    glClear(GL_COLOR_BUFFER_BIT);

    road();
    fullsky();
    grass();
    background();
    pitcher();
    sky1();
    sky2();
    glLoadIdentity();
    tree();
    glLoadIdentity();
    glTranslatef(-0.17, 0.0, 0.0);
    sun();
    glLoadIdentity();
    stone();
    glPushMatrix();
    glTranslatef(xpos, ypos, 0.0f);
    if (xpos >= -0.2f && ypos <= -0.55f && dsixf == 0)
    {
        std::cout << "**********Equal**********" << std::endl;
        PlaySound(TEXT("Crow went Pot.wav"), NULL, SND_SYNC);
        PlaySound(TEXT("Very Little Water In Pot.wav"), NULL, SND_SYNC);
        PlaySound(TEXT("Can't reach the water.wav"), NULL, SND_SYNC);
        dsixf = 1;
```

```
        waterflag = 1;
    }

    std::cout << xpos << " " << ypos << std::endl;

    glRotatef(180, 0, 1, 0);
    if (waterflag == 1)
    {
        glRotatef(180, 0, 1, 0);
        Bird();
        ideaflag = 1;
    }
    else
        Bird();
    glPopMatrix();
    glFlush();
    if (waterflag == 1 && ideaflag == 1 && ideamusic == 0)
    {
        PlaySound(TEXT("Saw pebble.wav"), NULL, SND_SYNC);
        ideamusic = 1;
    }
}

void Display5()
{
    glClearColor(1.0f, 1.0f, 1.0f, 1.0f); // Set background color to black and opaque
    glClear(GL_COLOR_BUFFER_BIT);

    road();
    fullsky();
    grass();
    background();
    pitcher();
    sky1();
```

```
    sky2();
    glLoadIdentity();
    tree();
    glLoadIdentity();
    glTranslatef(-0.17, 0.0, 0.0);
    sun();
    glLoadIdentity();
    stone();
    glPushMatrix();
    glTranslatef(-1.0, 0.45f, 0.0f);
    glRotatef(180, 0, 1, 0);
    Bird();
    glPopMatrix();

    glFlush();

    if (dfivef == 0)
    {
        PlaySound(TEXT("Crow saw a pot.wav"), NULL, SND_SYNC);
        dfivef = 1;
    }
}

void Display4()
{
    std::cout << "Display 4 displayed" << std::endl;
    glClearColor(1.0f, 1.0f, 1.0f, 1.0f); // Set background color to black and opaque
    glClear(GL_COLOR_BUFFER_BIT);
    glBegin(GL_POLYGON);
    glColor3ub(124, 252, 0);
    glVertex2f(-1.0f, -0.85f);
    glVertex2f(1.0f, -0.7f);
    glVertex2f(1.0f, -1.0);
    glVertex2f(-1.0f, -1.0);
```

```
glEnd();

fullsky();
grass();
background();
glPushMatrix();
glTranslatef(skypos, 0.0, 0.0);
sky1();
sky2();
glPopMatrix();

glLoadIdentity();
glTranslatef(position2, 0.0, 0.0);
Normal_Tree();
glLoadIdentity();
glTranslatef(position2 + 0.6, 0.0, 0.0);
Normal_Tree();
glLoadIdentity();
glTranslatef(position2 + 1.2, 0.0, 0.0);
Normal_Tree();
glLoadIdentity();
glTranslatef(-0.17, 0.0, 0.0);
sun();

glLoadIdentity();
glPushMatrix();
glTranslatef(position, 0.4f, 0.0f);
glRotatef(180, 0, 1, 0);
Bird();
std::cout << position << std::endl;
glPopMatrix();
glLoadIdentity();
glFlush();
```

```
    if (dfourf == 0)
    {
        //std::cout<<"audio played"<<std::endl;
        Sleep(1000);
        PlaySound(TEXT("Crow Sound.wav"), NULL, SND_SYNC);
        dfourf = 1;
    }
}

void Display3()
{
    std::cout << "Display 2 displayed" << std::endl;
    glClearColor(1.0f, 1.0f, 1.0f, 1.0f); // Set background color to black and opaque
    glClear(GL_COLOR_BUFFER_BIT);
    glBegin(GL_POLYGON);
    glColor3ub(124, 252, 0);
    glVertex2f(-1.0f, -0.85f);
    glVertex2f(1.0f, -0.7f);
    glVertex2f(1.0f, -1.0);
    glVertex2f(-1.0f, -1.0);
    glEnd();

    fullsky();
    grass();
    background();
    sky1();
    sky2();

    glLoadIdentity();
    glTranslatef(0.0, 0.0, 0.0);
    Normal_Tree();
    glTranslatef(0.5, 0.0, 0.0);
    Home();
    glLoadIdentity();
```

```
    glTranslatef(-0.17, 0.0, 0.0);
    sun();

    //glLoadIdentity();
    glPushMatrix();
    glTranslatef(position, 0.4f, 0.0f);
    glRotatef(180, 0, 1, 0);
    Bird();
    glPopMatrix();
    glLoadIdentity();
    glFlush();

    if (dthreef == 0)
    {
        //std::cout<<"audio played"<<std::endl;
        PlaySound(TEXT("Crow search Water.wav"), NULL, SND_SYNC);
        dthreef = 1;
    }

}

void Display2()
{
    std::cout << "Display 2 displayed" << std::endl;
    glClearColor(1.0f, 1.0f, 1.0f, 1.0f); // Set background color to black and opaque
    glClear(GL_COLOR_BUFFER_BIT);
    glBegin(GL_POLYGON);
    glColor3ub(124, 252, 0);
    glVertex2f(-1.0f, -0.85f);
    glVertex2f(1.0f, -0.7f);
    glVertex2f(1.0f, -1.0);
    glVertex2f(-1.0f, -1.0);
    glEnd();
```

```
fullsky();
grass();
background();
sky1();
sky2();

glLoadIdentity();
glTranslatef(0.0, 0.0, 0.0);
Normal_Tree();
glLoadIdentity();
glTranslatef(0.6, 0.0, 0.0);
Normal_Tree();
glLoadIdentity();
glTranslatef(1.2, 0.0, 0.0);
Normal_Tree();
glLoadIdentity();
glTranslatef(-0.17, 0.0, 0.0);
sun();

glPushMatrix();
glTranslatef(position, 0.4f, 0.0f);
glRotatef(180, 0, 1, 0);
Bird();
std::cout << position << std::endl;
glPopMatrix();
glLoadIdentity();
glFlush();

if (dtwof == 0)
{
   //std::cout<<"audio played"<<std::endl;
   PlaySound(TEXT("Crow was thirsty.wav"), NULL, SND_SYNC);
   dtwof = 1;
}
```

```
}

void Display1()
{
   glClearColor(1.0f, 1.0f, 1.0f, 1.0f); // Set background color to black and opaque
   glClear(GL_COLOR_BUFFER_BIT);
   glBegin(GL_POLYGON);
   glColor3ub(124, 252, 0);
   glVertex2f(-1.0f, -0.85f);
   glVertex2f(1.0f, -0.7f);
   glVertex2f(1.0f, -1.0);
   glVertex2f(-1.0f, -1.0);
   glEnd();

   fullsky();
   grass();
   background();
   sky1();
   sky2();

   glLoadIdentity();
   glTranslatef(0.0, 0.0, 0.0);
   Normal_Tree();
   glLoadIdentity();
   glTranslatef(0.6, 0.0, 0.0);
   Normal_Tree();
   glLoadIdentity();
   glTranslatef(1.2, 0.0, 0.0);
   Normal_Tree();
   glLoadIdentity();
   glTranslatef(-0.17, 0.0, 0.0);
   sun();
   glLoadIdentity();
   glFlush();
```

```
    if (donef == 0)
    {
        PlaySound(TEXT("It was a hot.wav"), NULL, SND_SYNC);
        donef = 1;
    }
}

void handleKeypress(unsigned char key, int x, int y) {
    switch (key) {
    case '1':
        std::cout << "1 Pressed" << std::endl;
        glutDestroyWindow(1);
        glutInitWindowSize(1240, 680);
        glutInitWindowPosition((glutGet(GLUT_SCREEN_WIDTH) - 1240) / 2, (glutGet(GLUT_SCREEN_HEIGHT) - 680) / 2);
        glutCreateWindow("MORAL STORY");
        glutKeyboardFunc(handleKeypress);
        glutDisplayFunc(Display1);
        break;
    case '2':
        std::cout << "2 Pressed" << std::endl;
        position = -1.0f;
        glutDisplayFunc(Display2);
        break;
    case '3':
        std::cout << "3 Pressed" << std::endl;
        position = -1.0f;
        glutDisplayFunc(Display3);
        break;
    case '4':
        std::cout << "4 Pressed" << std::endl;
        position = -1.0f;
        glutDisplayFunc(Display4);
        break;
    case '5':
```

```
        std::cout << "5 Pressed" << std::endl;
        position = -0.9f;
        glutDisplayFunc(Display5);
        break;
    case '6':
        std::cout << "6 Pressed" << std::endl;
        position = -0.9f;
        xpos = -1.0;
        ypos = 0.45;
        glutDisplayFunc(Display6);
        break;
    case '7':
        std::cout << "7 Pressed" << std::endl;
        position = -0.9f;
        xpos = -1.4;
        ypos = -0.85;
        yuppos = ypos;
        febbleflag = 1;
        stonereturnflag = 1;
        glutDisplayFunc(Display7);
        break;
    }
}

int main(int argc, char** argv)
{
    glutInit(&argc, argv);
    glutInitWindowSize(1240, 680);
    glutInitWindowPosition((glutGet(GLUT_SCREEN_WIDTH) - 1240) / 2, (glutGet(GLUT_SCREEN_HEIGHT) - 680) / 2);
    glutCreateWindow("MORAL STORY");
    init();
    glutReshapeFunc(reshape);
    glutDisplayFunc(Display);
    glutKeyboardFunc(handleKeypress);
```

```
    glutTimerFunc(100, update, 0);
    glutMainLoop();
    return 0;
}
```