CS 3353 Laboratory Problem Set 2

September 29, 2009
Due October 8, 2009

These problems are to be done on an individual basis following the Trinity University Academic Integrity Policy or Trinity University Honor Code.

Academic Integrity and Honor Code

All students are covered by a policy that prohibits dishonesty in academic work. The Academic Integrity Policy (AIP) covers all students who entered Trinity before the Fall of 2004. The Academic Honor Code covers all those who entered the Fall of 2004 or later. The Integrity Policy and the Code share many features: each asserts that the academic community is based on honesty and trust; each contains the same violations; each provides for a procedure to determine if a violation has occurred and what the punishment will be; each provides for an appeal process. The main difference is that the faculty implements the AIP while the Honor Code is implemented by the Academic Honor Council. Under the Academic Integrity Policy, the faculty member determines whether a violation has occurred as well as the punishment for the violation (if any) within certain guidelines. Under the Honor Code, a faculty member will (or a student may) report an alleged violation to the Academic Honor Council. It is the task of the Council to investigate, adjudicate, and assign a punishment within certain guidelines if a violation has been verified. Students who are under the Honor Code are required to pledge all written work that is submitted for a grade: On my honor, I have neither given nor received any unauthorized assistance on this work and heir signature. The pledge may be abbreviated pledged with a signature.

Laboratory problems should be submitted electronically (e-mail to cs3353@ariel.cs.trinity.edu) on or before the due date and should contain a problem write-up, source code to any programs and data sets used in solving the problem. The submitted files should be ASCII text files having Unix end-of-line characters (please convert all Windows and Mac text files to Unix format–I have found that Emacs seems to do a reasonable job of such conversions). If several files need to be submitted, put them in a directory having name *your-last-name-problem-set-number* and create a tar archive of this file system and attach it to your e-mail problem submission.

Extending 3d-move.c

The 3d-move.c, which is shown below, is to be extended in this lab problem.

You should add the following features to the `3d-move.c program.`

1. `Use perspective projection rather than orthographic projection.`
2. `Use dragging of the middle mouse button to zoom in or out along the z-axis rather than scaling the object being drawn.`
3. `Add a key input (use the ``s'' or ``S'' key) which will stop the spinning motion, but not otherwise change the orientation of the object at the time the key was pressed.`
4. `Change the drawing of any of the glut objects so that both the solid object and its polygon structure are visible when the object is drawn..`

5. Change the response to window re-sizing so that the aspect ratio of the object being drawn is not altered.

```
// 3d-move.c

#include <GL/glut.h>
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

double WIDTH = 800;
double HEIGHT = 800;

GLboolean motion = GL_TRUE;
GLfloat xangle = 0.0, yangle = 0.0, xspin = 0.0, yspin = 0.0;
GLfloat size = 1.0, factor = 10.0;
GLboolean leftb = GL_FALSE, middleb = GL_FALSE;

void keyboard(unsigned char key, int x, int y)
{
  switch (key) {
  case 27:            /* ESC */
  case 'q':
  case 'Q':
    exit(0);
  case 'r':
  case 'R':
    xangle = 0.0;
    yangle = 0.0;
    size = 1.0;
    xspin = 0.0;
    yspin = 0.0;
  }
}

void update(void)
{
  if (!leftb){
    xangle -= xspin;
    yangle -= yspin;
  }
  glutPostRedisplay();
}

void init(void)
{

}

void Draw(void)
{
  GLfloat objectColor[] = {1.0, 0.0, 0.0, 1.0};
  glColor3fv(objectColor);

  glPushMatrix();
  {
      glTranslatef(WIDTH/2, HEIGHT/2, 0.0);
//    glutSolidCube(300);
//    glutWireCube(300);
//    glutSolidSphere(150, 15, 15);
//    glutWireSphere(150, 15, 15);
//    glutSolidCone(200, 150, 10, 10);
//    glutWireCone(200, 150, 10, 10);
//    glutSolidTorus(30, 150, 20, 20);
//    glutWireTorus(30, 150, 20, 20);
//    glScalef(100.0, 100.0, 100.0);
//    glutSolidDodecahedron();
//    glScalef(100.0, 100.0, 100.0);
//    glutWireDodecahedron();
//    glutSolidTeapot(100);
      glutWireTeapot(100);
  }
  glPopMatrix();
}

void draw(void)
{
  int t, f;

  glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
```

```
  glPushMatrix();
  {
    glTranslatef(WIDTH / 2, HEIGHT / 2, 0);
    glScalef(size, size, size);
    glRotatef(xangle, 0.0, 1.0, 0.0);
    glRotatef(yangle, 1.0, 0.0, 0.0);
    glTranslatef(-WIDTH / 2, -HEIGHT / 2, 0);
    Draw();
  }
  glPopMatrix();
  glutSwapBuffers();
}

int oldx, oldy;

void mouse(int button, int state, int x, int y)
{
  if (button == GLUT_LEFT_BUTTON) {
    oldx = x;
    oldy = y;
    if (state == GLUT_DOWN)
      leftb = GL_TRUE;
    else
      leftb = GL_FALSE;
  }
  if (button == GLUT_MIDDLE_BUTTON) {
    oldx = x;
    oldy = y;
    if (state == GLUT_DOWN)
      middleb = GL_TRUE;
    else
      middleb = GL_FALSE;
  }
}

void visibility(int state)
{
  if (state == GLUT_VISIBLE && motion) {
    glutIdleFunc(update);
  } else {
    glutIdleFunc(NULL);
  }
}

void mouse_motion(int x, int y)
{
  if (leftb) {
    xspin = x/factor - oldx/factor;
    xangle -= xspin;
    yspin = y/factor - oldy/factor;
    yangle -= yspin;
  }
  if (middleb) {
    size -= y/factor - oldy/factor;
    if (size < 0.0)
       size = 0.0;
  }
  oldx = x;
  oldy = y;
  glutPostRedisplay();
}

int main(int argc, char *argv[])
{
  int i;

  glutInit(&argc, argv);
  glutInitWindowSize(WIDTH, HEIGHT);
  glutInitDisplayMode(GLUT_RGBA | GLUT_DOUBLE | GLUT_DEPTH);
  glutCreateWindow("3D Move Program");
  glutDisplayFunc(draw);
  glutKeyboardFunc(keyboard);

  glViewport(0, 0, WIDTH, HEIGHT);
  glMatrixMode(GL_PROJECTION);
  glLoadIdentity();
  glOrtho(0, WIDTH, 0, HEIGHT, -10000, 10000);
  glMatrixMode(GL_MODELVIEW);
  glLoadIdentity();
  glClearColor(0, 0, 0, 0);
```

```
  glClearDepth(1.0);

  glEnable(GL_CULL_FACE);
  glEnable(GL_DEPTH_TEST);

  glutMouseFunc(mouse);
  glutMotionFunc(mouse_motion);
  glutVisibilityFunc(visibility);

  init();

  glutMainLoop();
  return 0;
}
```

Problem Set 2 Solution [HTML] [PS] [PDF]