Computer Graphics Fall 2020
Assignment – S2 Solution

Note that in this case the objects are already in the center so there is no need to translate them to the center before rotation or scaling.

Nevertheless for a general object that is not in the center we have to perform T-1RT or T-1ST.

For example:

For rotation about z axis by 30 degrees with a fixed point of (1.0, 2.0, 3.0) use:

```
glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
glTranslatef(1.0, 2.0, 3.0);
glRotatef(30.0, 0.0, 0.0, 1.0);
glTranslatef(-1.0, -2.0, -3.0);
```

In this assignment the basic cube from S1 is already in the center. Hence, we only need to apply:

```
glPushMatrix();
glTranslatef(-0.3, -0.7, 0);
glRotatef(45, 1.0, 1.0, 1.0);
drawCube();
glPopMatrix();
```

The following Solution is due to Stephen Rapp

```
#include <GL/gl.h>
#include <GL/glu.h>
#include <GL/glut.h>
#include <math.h>
void exit(int) ;


//*****************************************************************
//Display setup
//*****************************************************************
 void display(void)
 {
   glClearColor(1.0,1.0,1.0,0.0);       // set white background color on clear
   glColor3f(200.0f, 0.0f, 255.0f);          // set the drawing color
   glPointSize(4.0);
   glLineWidth(4.0);
   glMatrixMode(GL_PROJECTION);
   glLoadIdentity();
   glFrustum(-2.0,2.0, -2.0,2.0, 2.0,6.0); //Define 4x4x4 view volume with near edge 2 far edge 6
```

```
   glMatrixMode(GL_MODELVIEW);
   glLoadIdentity();
   gluLookAt(0.0,0.0,2.0, 0.0,0.0,-2.0, 0.0,1.0,0.0); //Place camera on scene egde and invert Z
   glEnable(GL_CULL_FACE);        //Enable face culling for cube face rendering
   glClear(GL_COLOR_BUFFER_BIT);   //Starting with cleared screen

}
//*********************************************************************
//myKeyboard
//*********************************************************************
void myKeyboard(unsigned char theKey, int mouseX, int mouseY)
{
   int tmp=0; // variable to control second cube goto statement
   switch(theKey)
   {
      case 27:
      exit(-1); //terminate the program
      // New fresh scene
      case 'N':
      case 'n':
      glLoadIdentity();
      gluLookAt(0.0,0.0,2.0, 0.0,0.0,-2.0, 0.0,1.0,0.0);
//       glFlush();
//       break;
      //square
      //Draw scene
      case 'D':
      case 'd':
      glClear(GL_COLOR_BUFFER_BIT);
      case 'S':
      case 's':
      glPushMatrix();
      glTranslatef(-1,1,0);
      glScalef(.75,.75,.75);
      glRotatef(270, 0, 0, 1);
      glColor3f(0.0f, 255.0f, 0.0f);         // set the drawing color
      glBegin(GL_POLYGON);
      glVertex3f(-0.5, -0.5, 0.0);
      glVertex3f(-0.5, 0.5, 0.0);
      glVertex3f(0.5, 0.5, 0.0);
      glVertex3f(0.5, -0.5, 0.0);
      glEnd();
      glPopMatrix();
//       glFlush();         // send all output to display
//       break;
      //point
      case 'P':
      case 'p':
      glPushMatrix();
      glTranslatef(0,1,0);
      glScalef(2,2,2);
```

```
        glColor3f(0.0f, 0.0f, 0.0f);          // set the drawing color
        glBegin(GL_POINTS);
        glVertex3f(0.0, 0.0, 0.0);
        glEnd();
        glPopMatrix();
//        glFlush();          // send all output to display
//        break;
        //line
        case 'L':
        case 'l':
        glPushMatrix();
        glTranslatef(0,-1,0);
        glScalef(1.5,1.5,1.5);
        glRotatef(90, 0, 0, 1);
        glColor3f(0.0f, 255.0f, 0.0f);          // set the drawing color
        glBegin(GL_LINE_STRIP);
        glVertex3f(-0.5, 0.0, 0.0);
        glVertex3f(0.5, 0.0, 0.0);
        glEnd();
        glPopMatrix();
//        glFlush();          // send all output to display
//        break;
        //triangle
        case 'T':
        case 't':
        glPushMatrix();
        glTranslatef(-1,-1.5,-1);
        glScalef(1,1,1);
        glRotatef(45, 0, 0, 1);
        glColor3f(0.0f, 0.0f, 255.0f);          // set the drawing color
        glBegin(GL_POLYGON);
        glVertex3f(-0.5, -0.5, 0.0);        // testing nonesense
        glVertex3f(0.5, -0.5, 0.0);        // testing nonesense
        glVertex3f(0.0, 0.5, 0.0);        // testing nonesense
        glEnd();
        glPopMatrix();
//        glFlush();          // send all output to display
//        break;
        //cube
        case 'U':
        case 'u':
        glPushMatrix();
        glTranslatef(1, 1, -1);
        glScalef(.25,.25,.25);
        second:
        //top face
        glColor3f(255.0f, 0.0f, 255.0f);          // set the drawing color PURPLE
        glBegin(GL_POLYGON);
        glVertex3f(0.5, 0.5, -0.5); //top right front
        glVertex3f(0.5, 0.5, 0.5); //top right back
        glVertex3f(-0.5, 0.5, 0.5); //top left back
```

```
        glVertex3f(-0.5, 0.5, -0.5); //top left front
        glEnd();
        //bottom face
        glColor3f(0.0f, 0.0f, 0.0f);          // set the drawing color BLACK
        glBegin(GL_POLYGON);
        glVertex3f(-0.5, -0.5, -0.5); //bottom left front
        glVertex3f(-0.5, -0.5, 0.5); //bottom left back
        glVertex3f(0.5, -0.5, 0.5); //bottom right back
        glVertex3f(0.5, -0.5, -0.5); //bottom right front
        glEnd();
        //left face
        glColor3f(255.0f, 0.0f, 0.0f);          // set the drawing color RED
        glBegin(GL_POLYGON);
        glVertex3f(-0.5, 0.5, -0.5); //top left front
        glVertex3f(-0.5, 0.5, 0.5); //top left back
        glVertex3f(-0.5, -0.5, 0.5); //bottom left back
        glVertex3f(-0.5, -0.5, -0.5); //bottom left front
        glEnd();
        //right face
        glColor3f(255.0f, 255.0f, 0.0f);          // set the drawing color YELLOW
        glBegin(GL_POLYGON);
        glVertex3f(0.5, 0.5, -0.5); //top right front
        glVertex3f(0.5, -0.5, -0.5); //bottom right front
        glVertex3f(0.5, -0.5, 0.5); //bottom right back
        glVertex3f(0.5, 0.5, 0.5); //top right back
        glEnd();
        //rear face
        glColor3f(0.0f, 255.0f, 0.0f);          // set the drawing color GREEN
        glBegin(GL_POLYGON);
        glVertex3f(0.5, 0.5, 0.5); //top right back
        glVertex3f(0.5, -0.5, 0.5); //bottom right back
        glVertex3f(-0.5, -0.5, 0.5); //bottom left back
        glVertex3f(-0.5, 0.5, 0.5); //top left back
        glEnd();
        //front face
        glColor3f(0.0f, 0.0f, 255.0f);          // set the drawing color BLUE
        glBegin(GL_POLYGON);
        glVertex3f(0.5, 0.5, -0.5); //right top front
        glVertex3f(-0.5, 0.5, -0.5); //left top front
        glVertex3f(-0.5, -0.5, -0.5); //bottom left front
        glVertex3f(0.5, -0.5, -0.5); //bottom right front
        glEnd();
        glPopMatrix();
//        glFlush();          // send all output to display

        if(tmp == 0 ){  //check to see if second cube has been drawn
           tmp = 1;
           glPushMatrix();
           glTranslatef(-1, 1, 0);
           glScalef(.25,.25,.25);
           glRotatef(45, 1, 1, 1);
```

```
            goto second;
        }

//        break;
        //circle
        case 'C':
        case 'c':
        glPushMatrix();
        glTranslatef(1, -1, -0.5);
        glScalef(.75,1.25,1);
        glRotatef(225, 0, 0, 1);
        glColor3f(255.0f, 255.0f, 0.0f);          // set the drawing color
 //       glClear(GL_COLOR_BUFFER_BIT);     // clear the screen
        glBegin(GL_POLYGON);
        //using angles to find radius to derive point vertex position
        for(int i =0; i <= 360; i++){
            double x = cos(i*(M_PI/180))/2;
            double y = sin(i*(M_PI/180))/2;
            glVertex3d(x,y,0.0);
        }
        glEnd();
        glPopMatrix();
//        glFlush();          // send all output to display
//        break;
        //hexagon
        case 'H':
        case 'h':
        glPushMatrix();
        glTranslatef(-1, 0, 0);
        glScalef(1,.5,1);
        glRotatef(30, 0, 0, 1);
        glColor3f(0.0f, 0.0f, 0.0f);          // set the drawing color
  //      glClear(GL_COLOR_BUFFER_BIT);     // clear the screen
        glBegin(GL_POLYGON);
        //using angles to derive point vertex position
        for(int i = 0; i <= 360; i+=60){
            double x = cos(i* (M_PI/180))/2;
            double y = sin(i* (M_PI/180))/2;
            glVertex3d(x,y,0.0);
        }
        glEnd();
        glPopMatrix();
        glFlush();          // send all output to display
        break;
        default:
        break; // do nothing
    }
}
//*********************************************************************
//Special Key Function
//*********************************************************************
```

```
void mySpecial( int special, int mouseX, int mouseY)
{
   if (special == GLUT_KEY_LEFT){
      glRotatef( 45, 0.0, 0.0, 1.0 );
   //    glLoadIdentity();
   //    gluLookAt(-2.0,0.0,2.0, 0.0,0.0,-2.0, 0.0,1.0,0.0);
   }
   else if (special == GLUT_KEY_RIGHT){
      glRotatef( -45, 0.0, 0.0, 1.0 );
   //    glLoadIdentity();
   //    gluLookAt(2.0,0.0,2.0, 0.0,0.0,-2.0, 0.0,1.0,0.0);
   }
   else if (special == GLUT_KEY_UP){
      glLoadIdentity();
      gluLookAt(0.0,-2.0,2.0, 0.0,0.0,-2.0, 0.0,1.0,0.0);
   }
   else if (special == GLUT_KEY_DOWN){
      glLoadIdentity();
      gluLookAt(0.0,2.0,2.0, 0.0,0.0,-2.0, 0.0,1.0,0.0);
   }
   myKeyboard('d',0,0); // redraw everything with new perspective
}

//*****************************************************************
//main
//*****************************************************************
int         main(int argc, char** argv)
{
   glutInit(&argc, argv);          // initialize the toolkit
   glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB); // set display mode
   glutInitWindowSize(800,600);     // set window size
   glutInitWindowPosition(100, 150); // set window position on screen
   glViewport(50,50,500,500);
   glutCreateWindow("Type N for new scene, esc to quit") ; // open the screen window
   glutDisplayFunc(display);     // register redraw function
   glutKeyboardFunc(myKeyboard); // register the keyboard action function
   glutSpecialFunc(mySpecial); // register the keyboard action function
   glutMainLoop();                         // go into a perpetual loop
}
```