```
// Tic Tac Toe or X's and O's.
// Keyboard input
// 'v' = view ortho/perspective
// 'l' = lighting on/of


#include <GL/glut.h>  // glut (gl utility toolkit) basic windows functions, keyboard, mouse.
#include <stdio.h>    // standard (I/O library)
#include <stdlib.h>   // standard library (set of standard C functions
#include <math.h>     // Math library (Higher math functions )
#include<string.h>

// lighting
GLfloat LightAmbient[]= { 0.5f, 0.5f, 0.5f, 1.0f };
GLfloat LightDiffuse[]= { 0.5f, 0.5f, 0.5f, 1.0f };
GLfloat LightPosition[]= { 5.0f, 25.0f, 5.0f, 1.0f };
GLfloat mat_specular[] = { 1.0, 1.0, 1.0, 1.0 };
int abc=0;
// mouse variables: Win = windows size, mouse = mouse position
int mouse_x, mouse_y, Win_x, Win_y, object_select;

// state variables for Orho/Perspective view, lighting on/off
static int view_state = 0, light_state = 0;

// Use to spin X's and O's
int spin, spinboxes;

// Win = 1 player wins, -1 computer wins, 2 tie.
// player or computer; 1 = X, -1 = O
// start_game indicates that game is in play.
int player, computer, win, start_game;


// alingment of boxes in which one can win
// We have 8 posiblities, 3 accross, 3 down and 2 diagnally
//
// 0 | 1 | 2
// 3 | 4 | 5
// 6 | 7 | 8
//
// row, columm, diagnal information

static int box[8][3] = {{0, 1, 2}, {3, 4, 5}, {6, 7, 8}, {0, 3, 6},
                        {1, 4, 7}, {2, 5, 8}, {0, 4, 8}, {2, 4, 6}};

// Storage for our game board
// 1 = X's, -1 = O's, 0 = open space

int box_map[9];
// center x,y location for each box
int object_map[9][2] = {{-6,6},{0,6},{6,6},{-6,0},{0,0},{6,0},{-6,-6},{0,-6},{6,-6}};
```

```
// quadric pointer for build our X
GLUquadricObj *Cylinder;

// Begin game routine
void init_game(void)
{
int i;

// Clear map for new game
for( i = 0; i < 9; i++)
       {
       box_map[i] = 0;
    }

// Set 0 for no winner
win = 0;
start_game = 1;
}



// Check for three in a row/colunm/diagnal
// returns 1 if there is a winner
int check_move( void )
{

int i, t = 0;

//Check for three in a row
for( i = 0; i < 8; i++)
       {
        t = box_map[box[i][0]] + box_map[box[i][1]] + box_map[box[i][2]];
        if ( (t == 3) || ( t == -3) )
         {
          spinboxes = i;
          return( 1 );
         }
    }
t = 0;
// check for tie
for( i = 0; i < 8; i++)
       {
       t = t + abs(box_map[box[i][0]]) + abs( box_map[box[i][1]]) + abs( box_map[box[i][2]]);
    }

if ( t == 24 ) return( 2 );

return( 0 );
}

// Do we need to block other player?
int blocking_win(void)
```

```
{
int i, t;
for( i = 0; i < 8; i++)
       {
        t = box_map[box[i][0]] + box_map[box[i][1]] + box_map[box[i][2]];
        if ( (t == 2) || ( t == -2) )
          {
          // Find empty
          if (box_map[box[i][0]] == 0) box_map[box[i][0]] = computer;
          if (box_map[box[i][1]] == 0) box_map[box[i][1]] = computer;
          if (box_map[box[i][2]] == 0) box_map[box[i][2]] = computer;
       return( 1 );
       }
    }
return( 0 );
}

// check for a free space in corner
int check_corner(void)
{
int i;

if ( box_map[0] == 0)
       {
       box_map[0] = computer;
       i = 1;
       return( 1 );
    }

if ( box_map[2] == 0)
       {
       box_map[2] = computer;
       i = 1;
       return( 1 );
    }

if ( box_map[6] == 0)
       {
       box_map[6] = computer;
       i = 1;
       return( 1 );
    }

if ( box_map[8] == 0)
       {
       box_map[8] = computer;
       i = 1;
       return( 1 );
    }

return( 0 );
}
```

```
// Check for free space in row
int check_row(void)
{

if ( box_map[4] == 0)
       {
       box_map[4] = computer;
       return( 1 );
    }

if ( box_map[1] == 0)
       {
       box_map[1] = computer;
       return( 1 );
    }

if ( box_map[3] == 0)
       {
       box_map[3] = computer;
       return( 1 );
    }
if ( box_map[5] == 0)
       {
       box_map[5] = computer;
       return( 1 );
    }
if ( box_map[7] == 0)
       {
       box_map[7] = computer;
       return( 1 );
    }

return( 0 );
}

// logic for computer's turn
int computer_move()
{
if ( blocking_win() == 1) return( 1 );
if ( check_corner() == 1) return( 1 );
if ( check_row() == 1) return( 1);

return( 0 );
}

// I use this to put text on the screen
void Sprint( int x, int y, char *st)
{
       int l,i;

       l=strlen( st ); // see how many characters are in text string.
```

```
        glRasterPos2i( x, y); // location to start printing text
        for( i=0; i < l; i++)  // loop until i is greater then l
                {
                glutBitmapCharacter(GLUT_BITMAP_TIMES_ROMAN_24, st[i]); // Print a character on the
screen
        }

}


// This creates the spinning of the cube.
static void TimeEvent(int te)
{

    spin++;  // increase cube rotation by 1
        if (spin > 360) spin = 180; // if over 360 degress, start back at zero.
        glutPostRedisplay();  // Update screen with new rotation data
        glutTimerFunc( 8, TimeEvent, 1);  // Reset our timmer.
}


// Setup our Opengl world, called once at startup.
void init(void)
{
   glClearColor (0.6,0.6,0.4,0.0);  // When screen cleared, use black.
   glShadeModel (GL_SMOOTH);  // How the object color will be rendered smooth or flat
   glEnable(GL_DEPTH_TEST);   // Check depth when rendering
   // Lighting is added to scene
   glLightfv(GL_LIGHT1 ,GL_AMBIENT, LightAmbient);
     glLightfv(GL_LIGHT1 ,GL_DIFFUSE, LightDiffuse);
     glLightfv(GL_LIGHT1 ,GL_POSITION, LightPosition);
     glEnable(GL_LIGHTING);  // Turn on lighting
     glEnable(GL_LIGHT1);    // Turn on light 1

   start_game = 0;
   win = 0;

   // Create a new quadric
   Cylinder = gluNewQuadric();
   gluQuadricDrawStyle( Cylinder, GLU_FILL );
   gluQuadricNormals( Cylinder, GLU_SMOOTH );
   gluQuadricOrientation( Cylinder, GLU_OUTSIDE );
}


void Draw_O(int x, int y, int z, int a)
{

glPushMatrix();
glTranslatef(x, y, z);
glRotatef(a, 1, 0, 0);
glutSolidTorus(0.5, 2.0, 8, 16);
```

```
glPopMatrix();

}


void Draw_X(int x, int y, int z, int a)
{

glPushMatrix();
glTranslatef(x, y, z);
glPushMatrix();
glRotatef(a, 1, 0, 0);
glRotatef(90, 0, 1, 0);
glRotatef(45, 1, 0, 0);
glTranslatef( 0, 0, -3);
gluCylinder( Cylinder, 0.5, 0.5, 6, 16, 16);
//glutSolidCone( 2.5, 3.0, 16, 8 );
glPopMatrix();
glPushMatrix();
glRotatef(a, 1, 0, 0);
glRotatef(90, 0, 1, 0);
glRotatef(315, 1, 0, 0);
glTranslatef( 0, 0, -3);
gluCylinder( Cylinder, 0.5, 0.5, 6, 16, 16);
//glutSolidCone( 2.5, 3.0, 16, 8 );
glPopMatrix();
glPopMatrix();

}


// Draw our world
void display(void)
{
       if(abc==3)
       {
              //int mk=0;
              // glColor3f(0.0,1.0,0.0);
                     glClear (GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);  //Clear the screen
                       glColor3f(0.0,1.0,0.0);
                    glMatrixMode (GL_PROJECTION);  // Tell opengl that we are doing project matrix
work
                    glLoadIdentity();  // Clear the matrix
                    glOrtho(-9.0, 9.0, -9.0, 9.0, 0.0, 30.0);  // Setup an Ortho view
                    glMatrixMode(GL_MODELVIEW);  // Tell opengl that we are doing model matrix
work. (drawing)
                    glLoadIdentity(); // Clear the model matrix

                    glDisable(GL_COLOR_MATERIAL);
                    glDisable(GL_LIGHTING);
                    glColor3f(0.0, 0.0, 1.0);
```

```
                    Sprint(-2, 0, "Project by");
                      Sprint(-2, -1, "Gajanan and Nitin");
                      Sprint(-3, -2, "To Start press right button");
                      Sprint(-3, -3, "right button for X's");
                      Sprint(-3, -4, "and left for O's");
              glutSwapBuffers();
       }
       else if(abc==0)
{

       glClear (GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);  //Clear the screen

       glMatrixMode (GL_PROJECTION);  // Tell opengl that we are doing project matrix work
       glLoadIdentity();  // Clear the matrix
       glOrtho(-9.0, 9.0, -9.0, 9.0, 0.0, 30.0);  // Setup an Ortho view
       glMatrixMode(GL_MODELVIEW);  // Tell opengl that we are doing model matrix work. (drawing)
       glLoadIdentity(); // Clear the model matrix

       glDisable(GL_COLOR_MATERIAL);
       glDisable(GL_LIGHTING);
       glColor3f(0.0, 0.0, 1.0);
          Sprint(-4, 0, "Project by Gajanana G Bhat and Nitin Kulkarni");
          Sprint(-3, -1, "Right Click to Start the Game");
       glutSwapBuffers();

}
else
{
//char txt[30];
int ix, iy;
int i;
int j;

glClear (GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);  //Clear the screen

glMatrixMode (GL_PROJECTION);  // Tell opengl that we are doing project matrix work
glLoadIdentity();  // Clear the matrix
glOrtho(-9.0, 9.0, -9.0, 9.0, 0.0, 30.0);  // Setup an Ortho view
glMatrixMode(GL_MODELVIEW);  // Tell opengl that we are doing model matrix work. (drawing)
glLoadIdentity(); // Clear the model matrix

glDisable(GL_COLOR_MATERIAL);
glDisable(GL_LIGHTING);
glColor3f(1.0, 0.0, 0.0);

/*if ( start_game == 0 )
   {
   Sprint(-2, 0, "ggb and kittu");
   Sprint(-3, -1, "To Start press");
   Sprint(-3, -2, "right button for X's");
   Sprint(-3, -3, "and left for O's");
```

```
   }
*/
if (win == 1) Sprint( -2, 1, "congratulations you win");
if (win == -1) Sprint( -2, 1, "Computer win");
if (win == 2) Sprint( -2, 1, "Tie");

// Setup view, and print view state on screen
if (view_state == 1)
       {
    glColor3f( 0.0, 0.0, 1.0);
    Sprint(-3, 8, "Perspective view");
    glMatrixMode (GL_PROJECTION);
    glLoadIdentity();
    gluPerspective(60, 1, 1, 30);
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
    }else
       {
       glColor3f( 1.0, 0.0, 0.0);
    Sprint(-2, 8, "Ortho view");
    }

// Lighting on/off
if (light_state == 1)
       {
       glDisable(GL_LIGHTING);
    glDisable(GL_COLOR_MATERIAL);
    }else
       {
    glEnable(GL_LIGHTING);
    glEnable(GL_COLOR_MATERIAL);
    }


gluLookAt( 0, 0, 20, 0, 0, 0, 0, 1, 0);


// Draw Grid
for( ix = 0; ix < 4; ix++)
       {
       glPushMatrix();
       glColor3f(1,1,1);
          glBegin(GL_LINES);
          glVertex2i(-9 , -9 + ix * 6);
          glVertex2i(9 , -9 + ix * 6 );
          glEnd();
          glPopMatrix();
       }
       for( iy = 0; iy < 4; iy++ )
          {
       glPushMatrix();
       glColor3f(1,1,1);
```

```
            glBegin(GL_LINES);
            glVertex2i(-9 + iy * 6, 9 );
            glVertex2i(-9 + iy * 6, -9 );
            glEnd();
            glPopMatrix();
            }

glColorMaterial(GL_FRONT, GL_AMBIENT);
glColor4f(0.5, 0.5, 0.5, 1.0);
glColorMaterial(GL_FRONT, GL_EMISSION);
glColor4f(0.0, 0.0, 0.0, 1.0 );
glColorMaterial(GL_FRONT, GL_SPECULAR);
glColor4f(0.35, 0.35, 0.35, 1.0);
glColorMaterial(GL_FRONT, GL_DIFFUSE);
glColor4f(0.69, 0.69, 0.69, 1.0);
//glDisable(GL_COLOR_MATERIAL);
glColor3f( 0.0, 0.0, 0.0);  // Cube color
//glEnable(GL_COLOR_MATERIAL);
// Draw object in box's

for( i = 0; i < 9; i++)
   {
   j = 0;
   if (abs( win ) == 1 )
      {
      if ( (i == box[spinboxes][0]) || (i == box[spinboxes][1]) || (i == box[spinboxes][2]))
         {
          j = spin;
          }else j = 0;
       }
   if(box_map[i] == 1) Draw_X( object_map[i][0], object_map[i][1], -1, j);

   if(box_map[i] == -1) Draw_O( object_map[i][0], object_map[i][1], -1, j);
   }

//glDisable(GL_COLOR_MATERIAL);

glutSwapBuffers();
}
}

// This is called when the window has been resized.
void reshape (int w, int h)
{
   Win_x = w;
   Win_y = h;
   glViewport (0, 0, (GLsizei) w, (GLsizei) h);
   glMatrixMode (GL_PROJECTION);
   glLoadIdentity ();
}
```

```
// Read the keyboard
void keyboard (unsigned char key, int x, int y)
{
   switch (key)
   {

      case 'v':
         case 'V':
                view_state = abs(view_state -1);
                break;
         case 'b':
         case 'B':
                light_state = abs(light_state -1);
                break;
         case 27:
         exit(0); // exit program when [ESC] key presseed
         break;
      default:
         break;
   }


}


void mouse(int button, int state, int x, int y)
{
// We convert windows mouse coords to out openGL coords
mouse_x =  (18 * (float) ((float)x/(float)Win_x))/6;
mouse_y =  (18 * (float) ((float)y/(float)Win_y))/6;

// What square have they clicked in?
object_select = mouse_x + mouse_y * 3;

if ( start_game == 0)
       {
    if ((button == GLUT_RIGHT_BUTTON) && (state == GLUT_DOWN))
      {
       player = 1;
          computer = -1;
          init_game();
          computer_move();
          return;
      }

    if ((button == GLUT_LEFT_BUTTON) && (state == GLUT_DOWN))
      {
         player = -1;
      computer = 1;
         init_game();

         return;
```

```
      }
   }

if ( start_game == 1)
       {
    if ((button == GLUT_LEFT_BUTTON) && (state == GLUT_DOWN))
      {
      if (win == 0)
         {
         if (box_map[ object_select ] == 0)
            {
            box_map[ object_select ] = player;
               win = check_move();
               if (win == 1)
                  {
                  start_game = 0;
               return;
               }
              computer_move();
                 win = check_move();
                 if (win == 1)
                    {
                 win = -1;
                 start_game = 0;
                 }
              }
            }
         }
   }

if ( win == 2 )start_game = 0;

}
void menu(int choice)
{

              switch(choice)
              {
              case 1: abc=1;
                           glutMouseFunc(mouse);
                           break;

              case 2:
                     view_state = abs(view_state -1);
                     break;
              case 3: abc=3;
                glutMouseFunc(mouse);
              break;
              case 4:
                           exit(0);
                           break;
```

```
            }
}
// Main program
int main(int argc, char** argv)
{
   glutInit(&argc, argv);
   glutInitDisplayMode (GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH);

   glutInitWindowSize (850,600);
   glutInitWindowPosition (10, 10);
   glutCreateWindow (argv[0]);
   glutSetWindowTitle("X's and O's 3D");
   init ();
   glutCreateMenu(menu);
   glutAddMenuEntry("start game",1);
   glutAddMenuEntry("prespective view",2);
   glutAddMenuEntry("help",3);
   glutAddMenuEntry("Quit",4);

   glutAttachMenu(GLUT_RIGHT_BUTTON);

   glutDisplayFunc(display);
   glutReshapeFunc(reshape);
   glutKeyboardFunc(keyboard);
   //glutMouseFunc(mouse);
   glutTimerFunc( 50, TimeEvent, 1);
   glutMainLoop();
   return 0;

}
```