3D Tic-Tac-Toe Using OpenGL

CSE-420 Project by Christopher Huber and Jennifer Waterson

Objective

The objective of his project was to create a simple tic-tac-toe game rendered in 3D using opengl, and try to gain a better understanding of the language and how it can be applied to a gaming experience.

Abstract

3D Tic-Tac-Toe is a simple opengl based game that accepts keyboard input to select an ‘X’ or an ‘O’ and translate it into a user designated space on the rendered grid.

Introduction

This was an entirely new experience, and we found it challenging. The idea was to create the game using opengl, which would give us a better understanding of how it might be applied to a game.

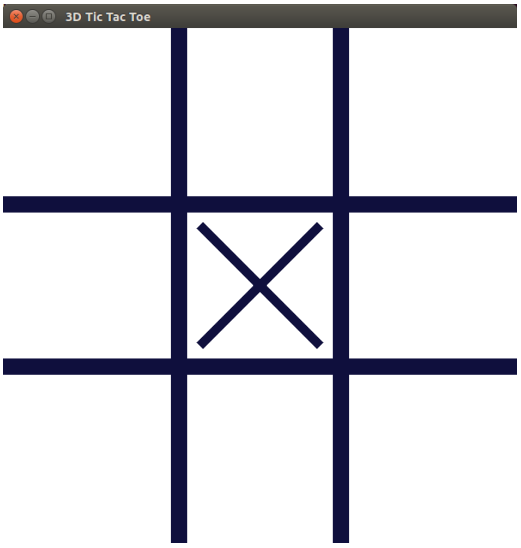
Software Package

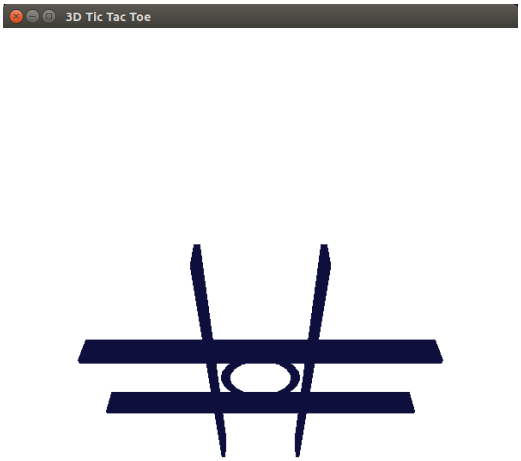
The program is made up on only two files: a main .cpp file and a makefile to compile it. The main file first draws a simple 3x3 grid made up of 3-dimentional blocks. Then it draws the player’s ‘X’ in the center space of the grid, which is also 3-dimensional, using solid cubes.

```
void gameBoard()
{
    glColor3f (0.0, 0.0, 0.0);
    glPushMatrix();

    glPushMatrix();
    glTranslatef (0, 1, 0);
    glScalef (7, .1, .5);
    glutSolidCube (1.0);
    glPopMatrix();

    glPushMatrix();
    glTranslatef (0, -1, 0);
    glScalef (7, .1, .5);
    glutSolidCube (1.0);
    glPopMatrix();

    glPushMatrix();
    glTranslatef (-1, 0, 0);
    glScalef (.1, 7, .5);
    glutSolidCube (1.0);
```

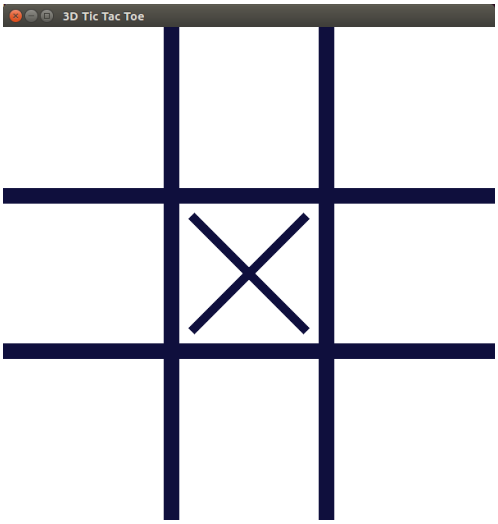
```
    glPopMatrix();

    glPushMatrix();
    glTranslatef (1, 0, 0);
    glScalef (.1, 7, .5);
    glutSolidCube (1.0);
    glPopMatrix();

    glPopMatrix();
}
```

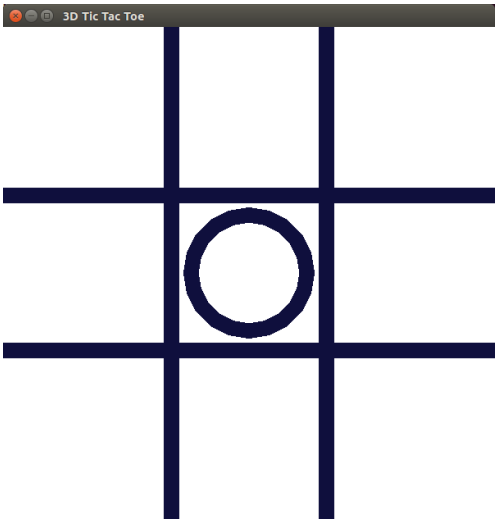
Here is the basic gameboard rendered both normally and with the camera angled to better snow the 3D blocks.





The ‘X’ can be toggled to an ‘O’, rendered with a solid torus ring, and back to an ‘X’.



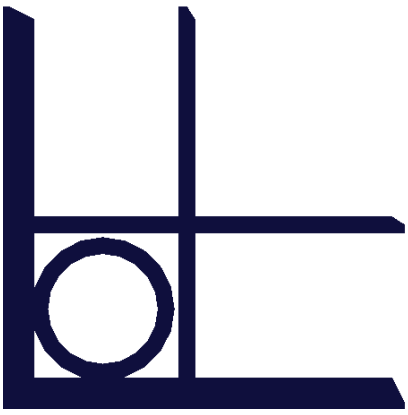


This function draws the ‘X’ and the ‘O’ and differentiate between which is displayed:

```
void drawPlayer()
{
    if (player == false)
    {
        glPushMatrix();
        glRotatef (45, 0, 0, 1);
        glScalef (2, .1, .5);
        glutSolidCube (1.0);
        glPopMatrix();

        glPushMatrix();
        glRotatef (-45, 0, 0, 1);
        glScalef (2, .1, .5);
        glutSolidCube (1.0);
        glPopMatrix();

        glPopMatrix();
     }
     else if (player == true)
     {
        glPushMatrix();
        glutSolidTorus (0.1, 0.75, 2, 20);
        glPopMatrix();
     }
}
```

The game can be played with two players, and makes use of an AI for single player mode, but wasn’t tested due to issues with other parts of the code. The program’s render window is divided into a 3x3 grid, and each space corresponds to a key on the num pad.

The idea was to translate the ‘X’ or ‘O’ to the specified space on the grid, however, when the translation is called, the entire game board is translated, as shown:



In order to avoid using the same space more than once, the game was set up using boolean checks with global variables.

```
static bool player = false;
static bool AI = false;
static bool one = false;
static bool two = false;
static bool three = false;
static bool four = false;
static bool five = false;
static bool six = false;
static bool seven = false;
static bool eight = false;
static bool nine = false;
static int x = 0;
static int y = 0;
static string answer;
```

When a key is pressed, a bool switch is switched to true. If the bool switch is true, nothing is rendered. The AI used a similar system, in that it would check is a bool switch was true, and if it was, it would move to a different space. Win conditions are also addressed with three separate functions that check for three of the same shape in a row.

```
void keyboard (unsigned char key, int
x, int y)
{
     if (player == false && AI == true)
     {
        switch (key)
        {
               case '1':
               if (one == false)
               {
                 one = true;
                 drawAI();
                 glutPostRedisplay();
                 break;
               }
               else if (one == true)
               {
                  break;
               }
               case '2':
               if (two == false)
               {
                two = true;
                drawAI();
                glutPostRedisplay();
                break;
               }
               else if (two == true)
               {
                break;
               }
               case '3':
               if (three == false)
               {
                three = true;
                drawAI();
                glutPostRedisplay();
                break;
               }
               else if (three == true)
               {
                break;
               }
               case '4':
               if (four == false)
               {
                four = true;
                drawAI();
                glutPostRedisplay();
                break;
               }
               else if (four == true)
               {
                break;
               }
               case '5':
               if (five == false)
               {
                five = true;
                drawAI();
                glutPostRedisplay();
                break;
               }
               else if (five == true)
               {
                break;
               }
               case '6':
```

```
                if (six == false)
                {
                 six = true;
                 drawAI();
                 glutPostRedisplay();
                 break;
                }
                else if (six == true)
                {
                 break;
                }
                case '7':
                if (seven == false)
                {
                 seven = true;
                 drawAI();
                 glutPostRedisplay();
                 break;
                }
                else if (seven == true)
                {
                 break;
                }
                case '8':
                if (eight == false)
                {
                 eight = true;
                 drawAI();
                 glutPostRedisplay();
                 break;
                }
                else if (eight == true)
                {
                 break;
                }
                case '9':
                if (nine == false)
                {
                 nine = true;
                 drawAI();
                 glutPostRedisplay();
                 break;
                }
                else if (nine == true)
                {
                 break;
                }
                case 27:
                exit(0);
                break;
                default:
                break;
        }
     }
    else //if (player = false && AI ==
false)
    {
        switch (key)
        {
            case '1':
                //if (one == false)
                //{
                    //glTranslatef (-2,
-2, 0);
                    player = !player;
                    one = true;

glutPostRedisplay();
                    break;
                //}
                //else if (one == true)
                //{
                    //break;
                //}
            case '2':
                if (two == false)
                {
                    player = !player;
                    two = true;

glutPostRedisplay();
                    break;
                }
                else if (two == true)
                {
                    break;
                }
            case '3':
               if (three == false)
               {
                player = !player;
                three = true;
                glutPostRedisplay();
                break;
               }
               else if (three == true)
               {
                break;
               }
            case '4':
               if (four == false)
               {
                player = !player;
                four = true;
                glutPostRedisplay();
                break;
               }
               else if (four == true)
               {
                break;
               }
               case '5':
               if (five == false)
               {
                player = !player;
                five = true;
                glutPostRedisplay();
                break;
               }
               else if (five == true)
               {
                break;
               }
               case '6':
               if (six == false)
```

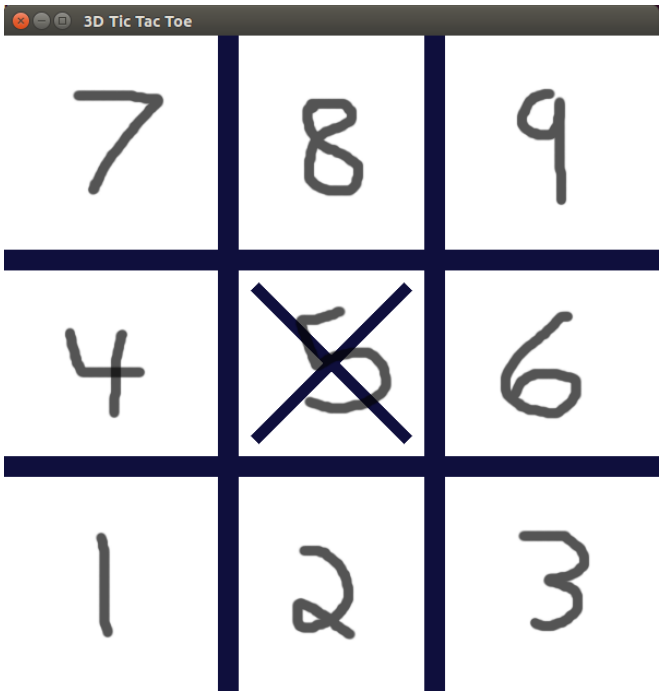
```
{
 player = !player;
 six = true;
 glutPostRedisplay();
 break;
}
else if (six == true)
{
 break;
}
case '7':
if (seven == false)
{
 player = !player;
 seven = true;
 glutPostRedisplay();
 break;
}
else if (seven == true)
{
 break;
}
case '8':
if (eight == false)
{
 player = !player;
 eight = true;
 glutPostRedisplay();
 break;
}
else if (eight == true)
{
 break;
}
case '9':
if (nine == false)
{
 player = !player;
 nine = true;
 glutPostRedisplay();
 break;
}
else if (nine == true)
{
 break;
}
case 27:
exit(0);
break;
default:
break;
}
}
}
```

The last part is the lighting. The game uses global variables to define the lighting and make the 3D aspects more visible:

```
GLfloat ambient[] = { 0.1, 0.1, 1.0 };
GLfloat diffuse[] = { 0.0, 0.0, 0.0, 1.0 };
GLfloat specular[] = { 1.0, 1.0, 1.0, 1.0 };
GLfloat shine[] = { 50.0 };
GLfloat low_shine[] = { 5.0 };
GLfloat position[] = { 2.0, 2.0, 1.0, 0.0 };

GLfloat mat_diffuse1[] = { 1.0, 0.0, 0.0, 1.0 };
GLfloat mat_diffuse2[] = { 0.0, 0.0, 1.0, 1.0 };
```

Interface

As stated before, the game runs off of boolean switches that check to see if a space has already been used. The spaces in the grid correspond to keys on the numpad. The user presses a key and the ‘X’ or ‘O’ appears in it’s designated space:



The win conditions of the game are to place three of the same shape in a straight row. The game includes several functions to check for these win conditions, one for player one, a second function for player two, and a third for the AI. All of them are similarly constructed.

```
void check()
{
    if (player == false)
    {
        if (one == true && two == true && three == true)
        {
            cout << "Player one wins" << endl;
        }
        else if (four == true && five == true && six == true)
        {
            cout << "Player one wins" << endl;
        }
        else if (seven == true && eight == true && nine == true)
        {
            cout << "Player one wins" << endl;
        }
```

```
    else if (one == true && four == true && seven == true)
    {
        cout << "Player one wins" << endl;
    }
    else if (two == true && five == true && eight == true)
    {
              cout << "Player one wins" << endl;
        }
        else if (three == true && six == true && nine == true)
        {
              cout << "Player one wins" << endl;
        }
        else if (three == true && five == true && seven == true)
        {
              cout << "Player one wins" << endl;
        }
        else if (one == true && five == true && nine == true)
        {
              cout << "Player one wins" << endl;
        }
}
```

Conclusion

The program was somewhat ambitious for our first opengl project. As such, we had trouble getting things to work the way they should've. The project was changed many times over during development as we tried to figure out what would work. We were very glad when we got the 'X' and 'O' to toggle back and forth, since that was tricky to figure out. The biggest obsticle was trying to keep the game board and the 'X' and 'O' pieces to translate seperately, since the entire board moves when the key is pressed, but time constraints prevented us from getting that aspect to function. If we had the time, we probably could've fixed it. But we are satisfied with the hours we put in.

References

Dr. Yu's lecture notes:
http://cse.csusb.edu/tongyu/courses/cs420/notes/index.php

The Book used for the class:
F.S. Hill, Jr. and Stephen M. Kelley, "Computer Graphics Using OpenGL", Latest Edition, Prentice Hall.