Van Dy

CS 4600

Hwk 3 3D Graphics

Link to my video for the assignment:

https://youtu.be/pSKRZ01zFFc

Code for computing normals. First we take the indices of each triangle, do the cross product with the vertexes and then plug them back into the normals array. At the end we simply do a normalize so that they are all unit length.

```
for (int i = 0; i < g_meshIndices.size(); i += 3)
{
    ax = g_meshVertices[g_meshIndices[i] * 3];
    ay = g_meshVertices[g_meshIndices[i] * 3 + 1];
    az = g_meshVertices[g_meshIndices[i] * 3 + 2];
    bx = g_meshVertices[g_meshIndices[i + 1] * 3];
    by = g_meshVertices[g_meshIndices[i + 1] * 3 + 1];
    bz = g_meshVertices[g_meshIndices[i + 1] * 3 + 2];
    cx = g_meshVertices[g_meshIndices[i + 2] * 3];
    cy = g_meshVertices[g_meshIndices[i + 2] * 3 + 1];
    cz = g_meshVertices[g_meshIndices[i + 2] * 3 + 2];

    //printf("index1 = %d, index2 = %d, index3 = %d\n", g_meshIndices[i],
    //printf("%f %f %f  %f %f %f  %f %f %f\n", ax, ay, az, bx, by, bz, cx

    xx = bx - ax; //vector a
    xy = by - ay;
    xz = bz - az;

    yx = cx - ax; //vector b
    yy = cy - ay;
    yz = cz - az;

    normi = xy * yz - yy * xz; //cross product
    normj = -(xx * yz - yx * xz);
    normk = xx * yy - yx * xy;

    g_meshNormals[g_meshIndices[i] * 3] += normi;
    g_meshNormals[g_meshIndices[i] * 3 + 1] += normj;
    g_meshNormals[g_meshIndices[i] * 3 + 2] += normk;
    g_meshNormals[g_meshIndices[i + 1] * 3] += normi;
    g_meshNormals[g_meshIndices[i + 1] * 3 + 1] += normj;
    g_meshNormals[g_meshIndices[i + 1] * 3 + 2] += normk;
    g_meshNormals[g_meshIndices[i + 2] * 3] += normi;
    g_meshNormals[g_meshIndices[i + 2] * 3 + 1] += normj;
    g_meshNormals[g_meshIndices[i + 2] * 3 + 2] += normk;
}

for (int v = 0; v < g_meshNormals.size(); v += 3)
{
    normalize(&g_meshNormals[v]);
}
```

Code for dolly zoom and orthogonal projection.

First we calculate the halfWidth value once and then lock it so it doesn't change. Then we do the equation in the enableDolly if statement below.

The code for glOrtho is the last line. The values there looked the best to me.

```
// Perspective Projection
if (enablePersp)
{
    if (halfWNotCalculated) {
        halfWidth = distance * tan(fov / 2); //calculates halfWidth
        halfWNotCalculated = false;
    }
    // Dolly zoom computation
    if (enableDolly) {

        // TASK 3
        // Your code for dolly zoom computation goes here
        // You can use getTime() to change fov over time
        // distance should be recalculated here using the Equation 2 in the description file

        if (distance > 3.0f) {
            fov = M_PI / (30.0f / getTime()); //change fov from A -> B
            distance = halfWidth / tanf(fov / 2); //distance calculation
        }
    }
    float fovInDegree = radianToDegree(fov);
    gluPerspective(fovInDegree, (GLfloat)g_windowWidth / (GLfloat)g_windowHeight, 1.0f, 40.f);
}
// Othogonal Projection
else
{
    // Scale down the object for a better view in orthographic projection
    glScalef(0.5, 0.5, 0.5);

    // TASK 3
    // Your code for orthogonal projection goes here
    // (Hint: you can use glOrtho() function in OpenGL)

    glOrtho(-1.5f, 1.5f, -1.0f, 1.0f, 1.0f, 40.f);
}
```