3D Graphics

Part 2

Billboards.

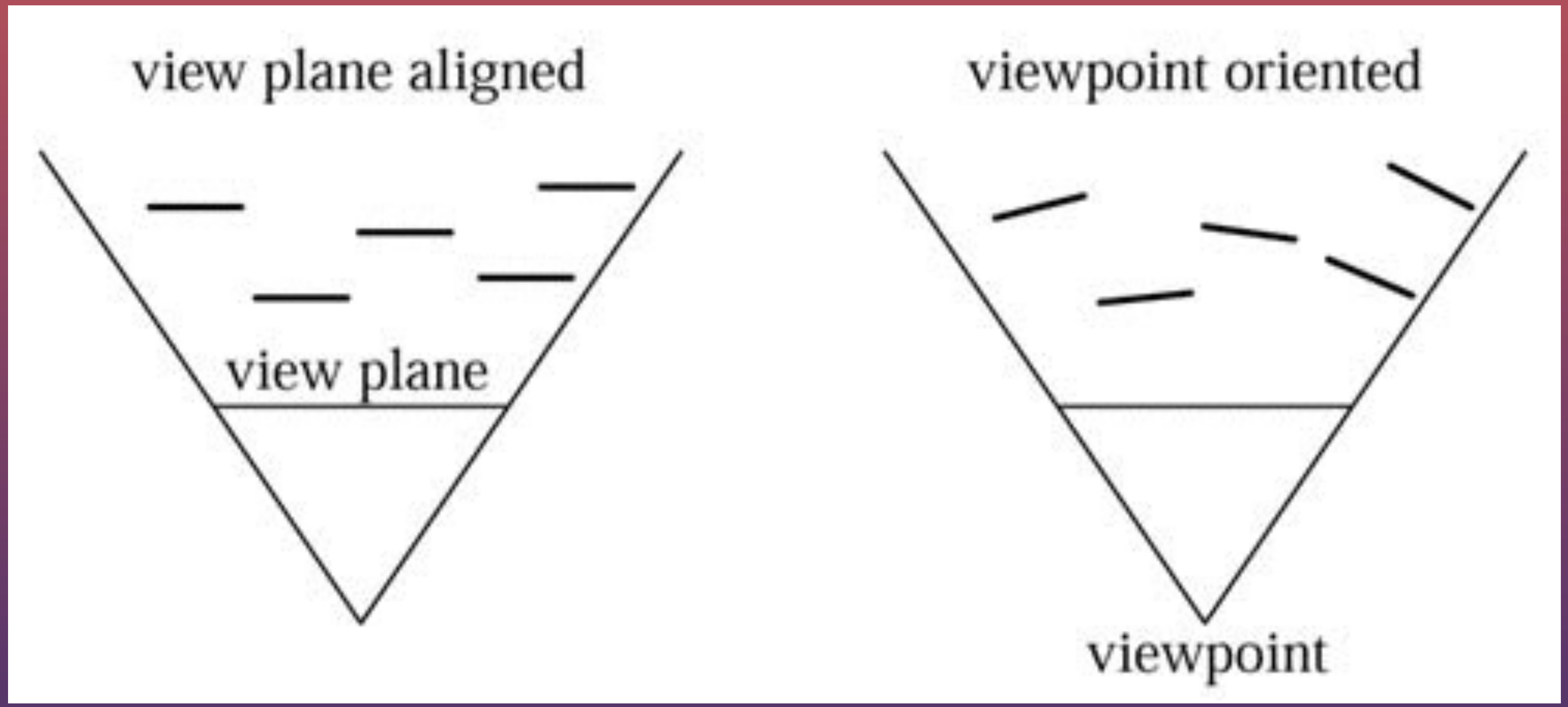

x 10
x0
x 16


PICKED UP 4 SHOTGUN SHELLS.
Xboxworld.nl
14
AMMO
59%
HEALTH
2 3 4
5 6 7
ARMS
0%
ARMOR
BULL 42 / 200
SHEL 14 / 50
ROKT 0 / 50
CELL 0 / 300

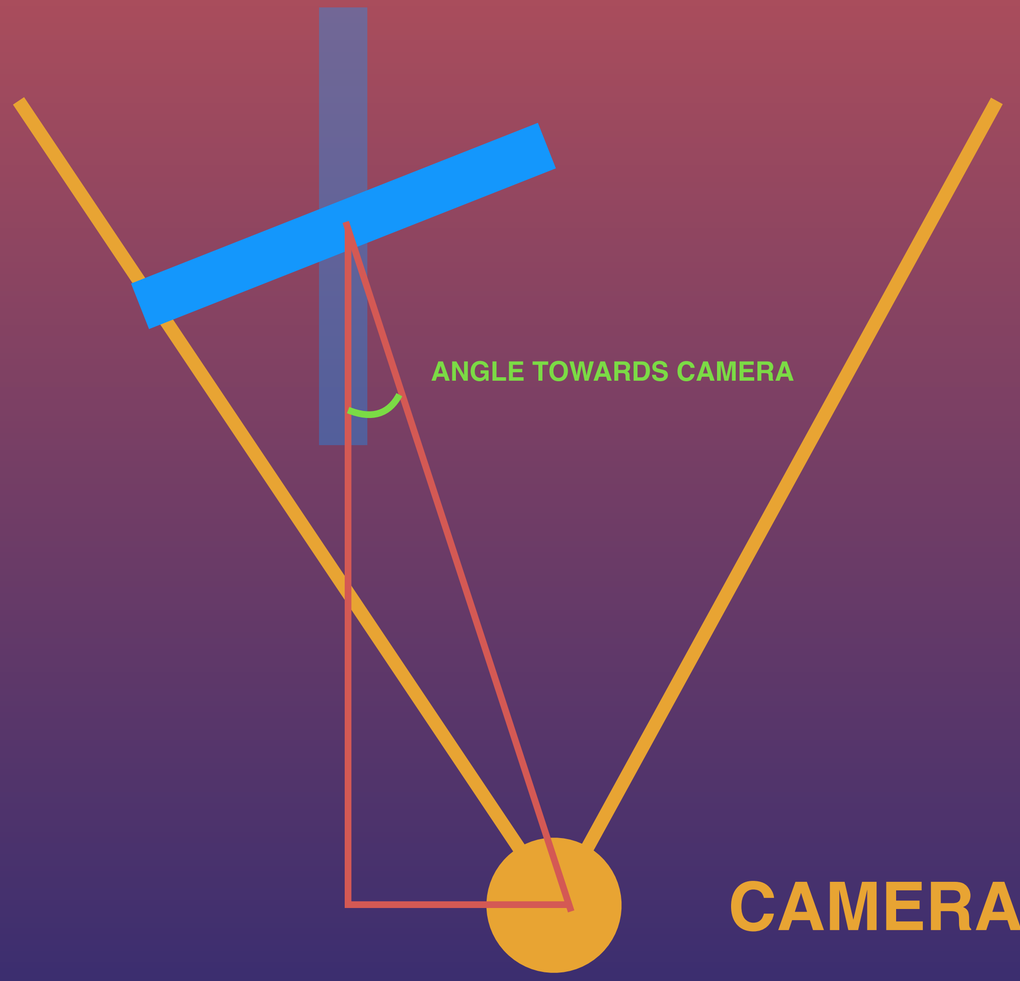
Billboards

=

Sprites that always face the camera.


view plane aligned
view plane
viewpoint oriented
viewpoint

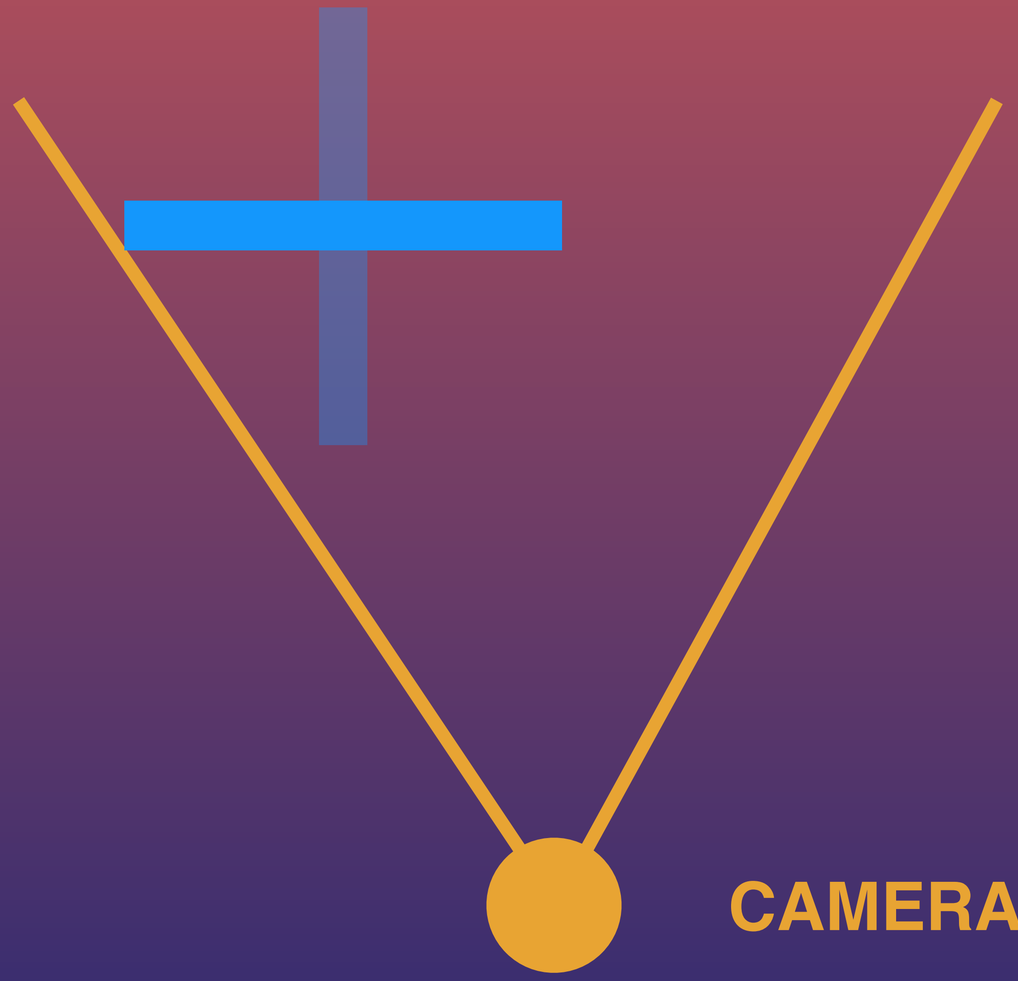
Method #1 - Rotate all billboards towards camera. (Viewpoint oriented)



Method #1 - Rotate all billboards towards camera. (Viewpoint oriented)

```
for(int i=0; i < entities.size(); i++) {

    if(entities[i]->billboard) {

        float directionX = entities[i]->position.x - camera->position.x;
        float directionZ = entities[i]->position.z - camera->position.z;

        entities[i]->rotation.y = atan2f(directionX, directionZ);
    }

    entities[i]->Render();
}
```

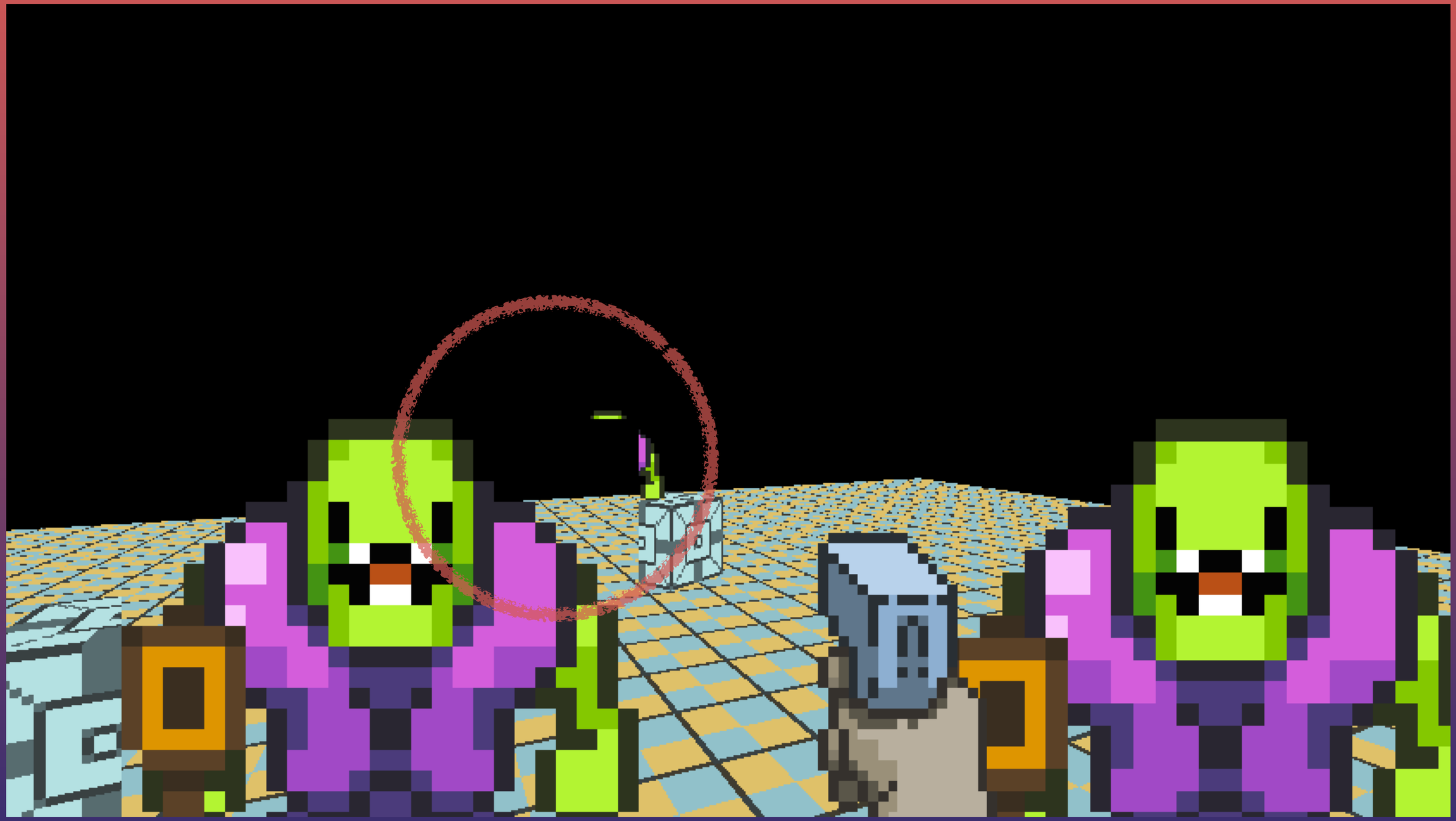
Method #2 - Remove all rotation from the modelview matrix. (Viewplane aligned)

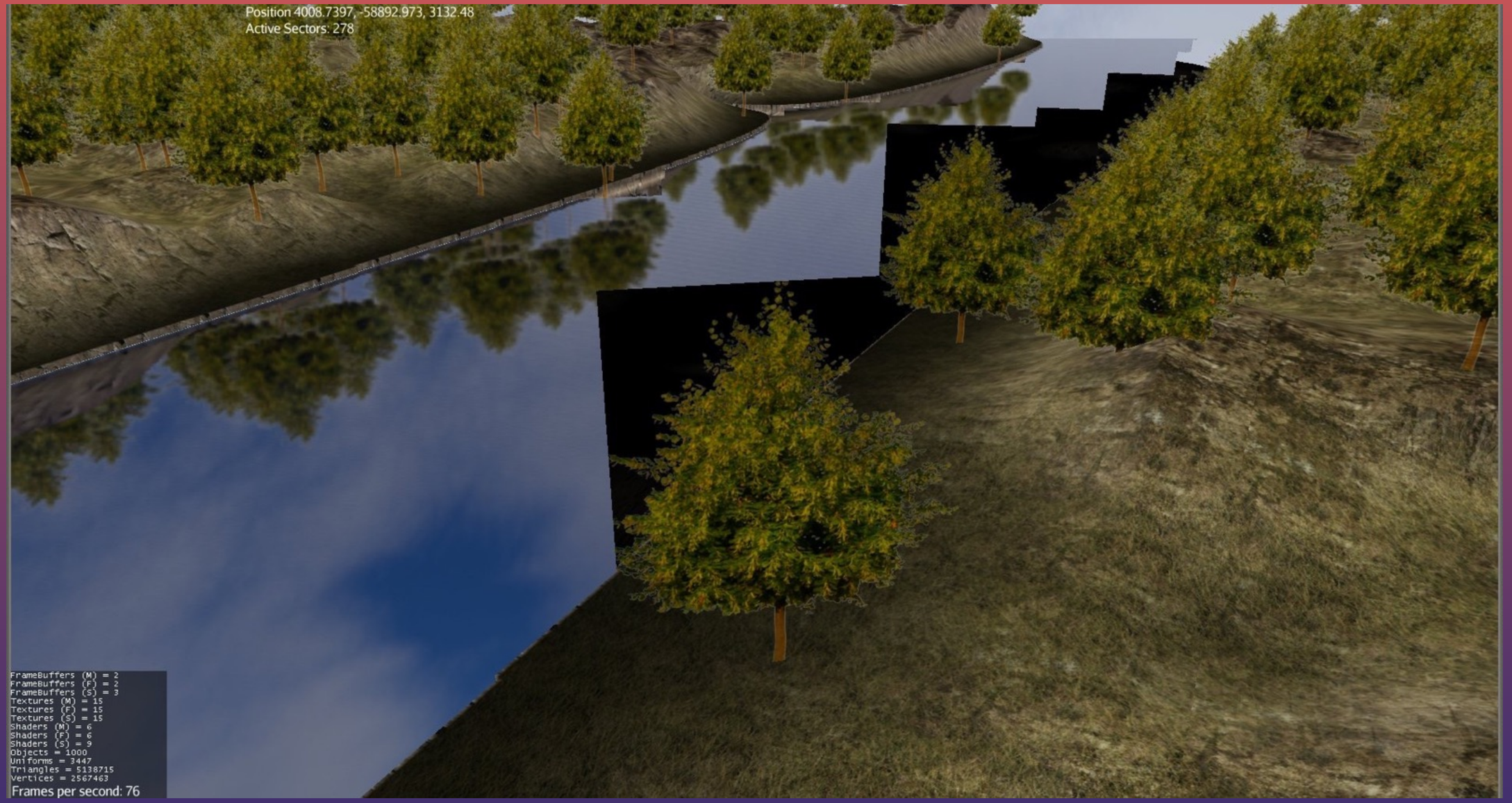


Method #2 - Remove all rotation from the modelview matrix. (Viewplane aligned)

```
if(billboard) {
    Matrix finalMatrix = matrix * camera->matrix.inverse();

    finalMatrix.m[0][0] = 1.0f;
    finalMatrix.m[0][1] = 0.0f;
    finalMatrix.m[0][2] = 0.0f;

    finalMatrix.m[1][0] = 0.0f;
    finalMatrix.m[1][1] = 1.0f;
    finalMatrix.m[1][2] = 0.0f;

    finalMatrix.m[2][0] = 0.0f;
    finalMatrix.m[2][1] = 0.0f;
    finalMatrix.m[2][2] = 1.0f;

    glLoadIdentity();
    glMultMatrixf(finalMatrix.ml);
} else {
    glMultMatrixf(matrix.ml);
}
```

Blending and the Z-Buffer




Position 4008.7397, -58892.973, 3132.48
Active Sectors: 278
FrameBuffers (M) = 2
FrameBuffers (F) = 2
FrameBuffers (S) = 3
Textures (M) = 15
Textures (F) = 15
Textures (S) = 15
Shaders (M) = 6
Shaders (F) = 6
Shaders (S) = 9
Objects = 1000
Uniforms = 3447
Triangles = 5138715
Vertices = 2567463
Frames per second: 76

```
void glEnable (GLenum capability);
void glDisable (GLenum capability);
```

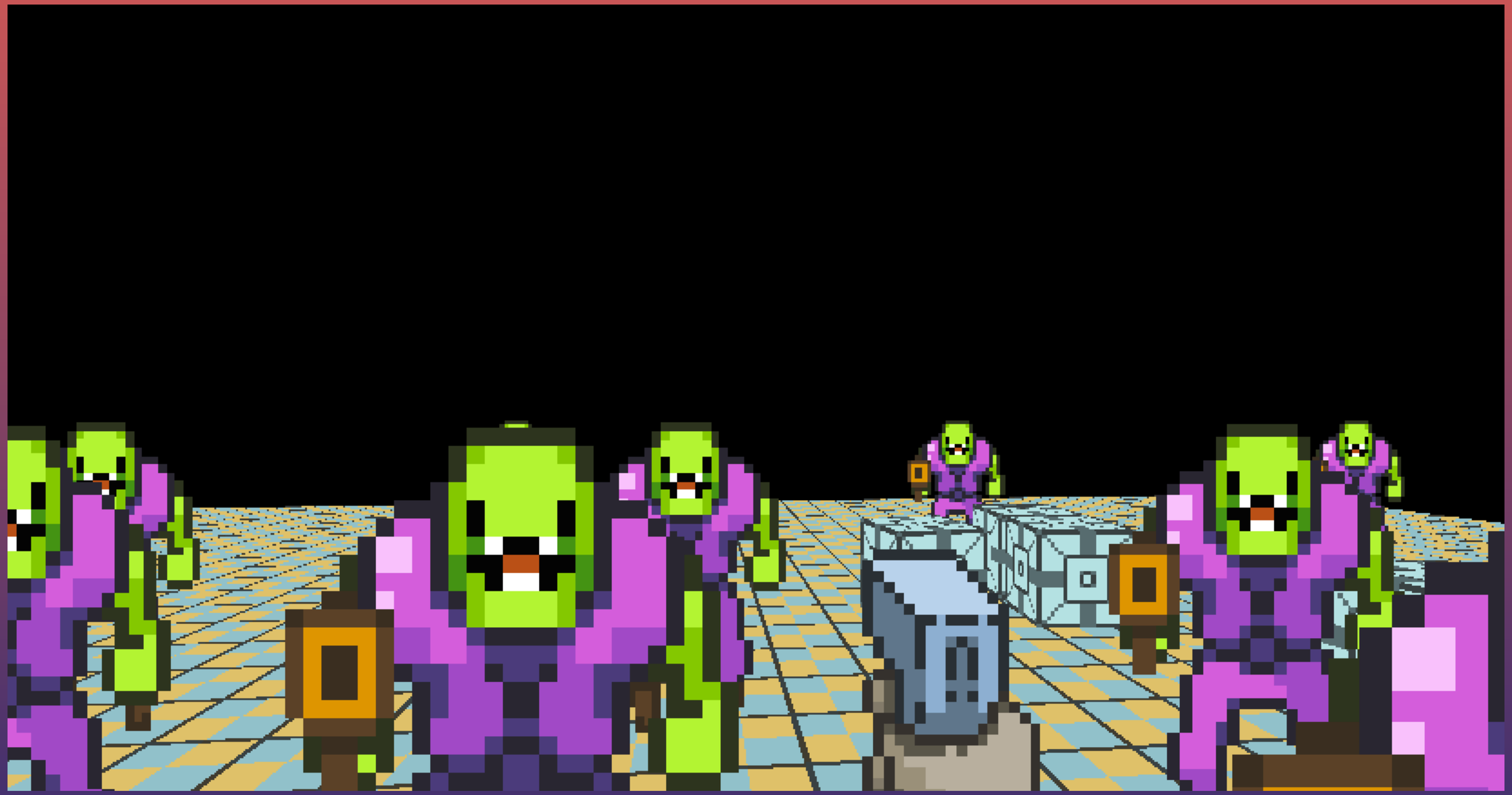
Enable or disable an OpenGL capability. Use GL_ALPHA_TEST to enable or disable alpha test capability.

```
glEnable(GL_ALPHA_TEST); // enable alpha testing
glDisable(GL_ALPHA_TEST); // disable alpha testing
```

```
void glAlphaFunc (GLenum func,
GLclampf ref);
```

Specifies the alpha comparison function and alpha value. Symbolic constants **GL_NEVER, GL_LESS, GL_EQUAL, GL_LEQUAL, GL_GREATER, GL_NOTEQUAL, GL_GEQUAL, and GL_ALWAYS** are accepted. The initial value is GL_ALWAYS.

```
glAlphaFunc ( GL_GREATER, 0.0f); // only render pixels above 0.0 alpha
```

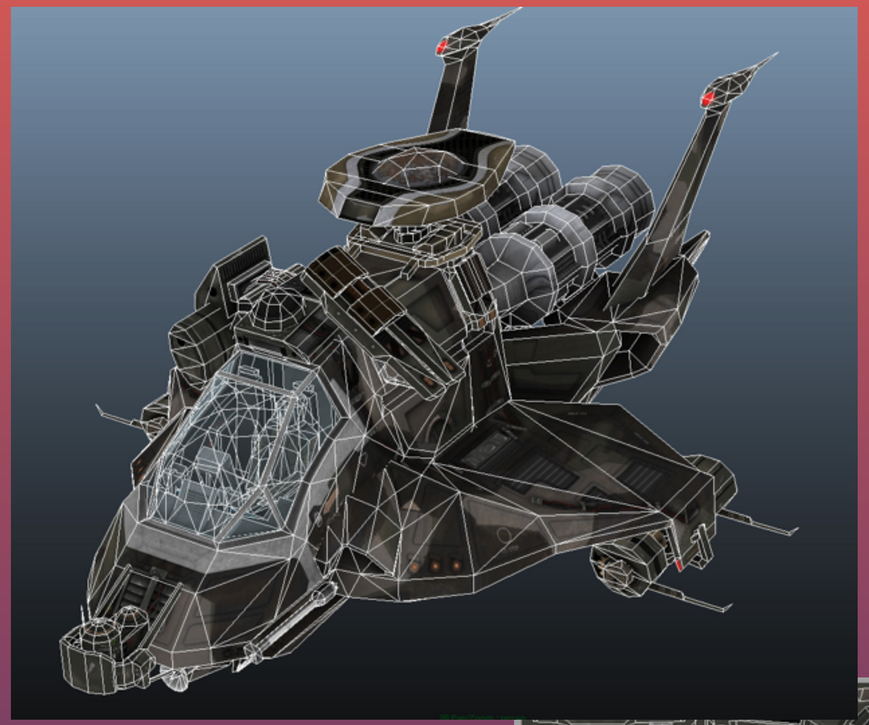


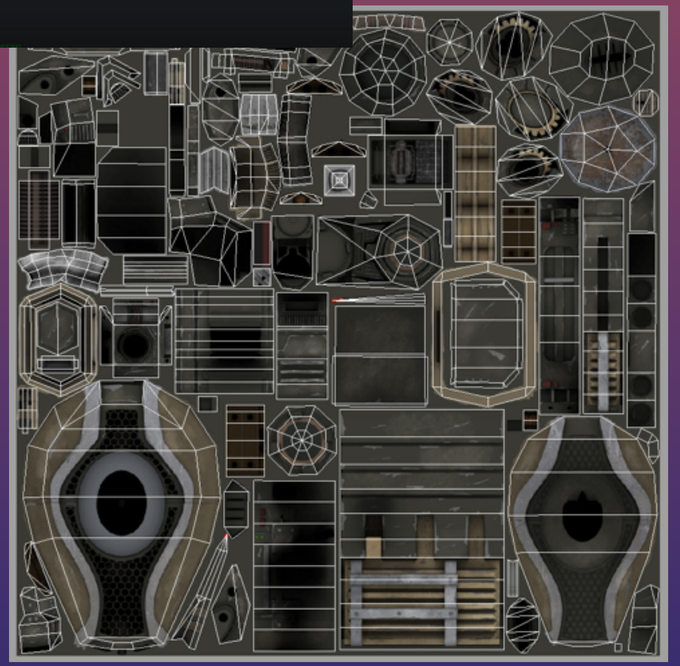
Or sort your drawing in such a way that transparent things get rendered last!

Storing 3D meshes.

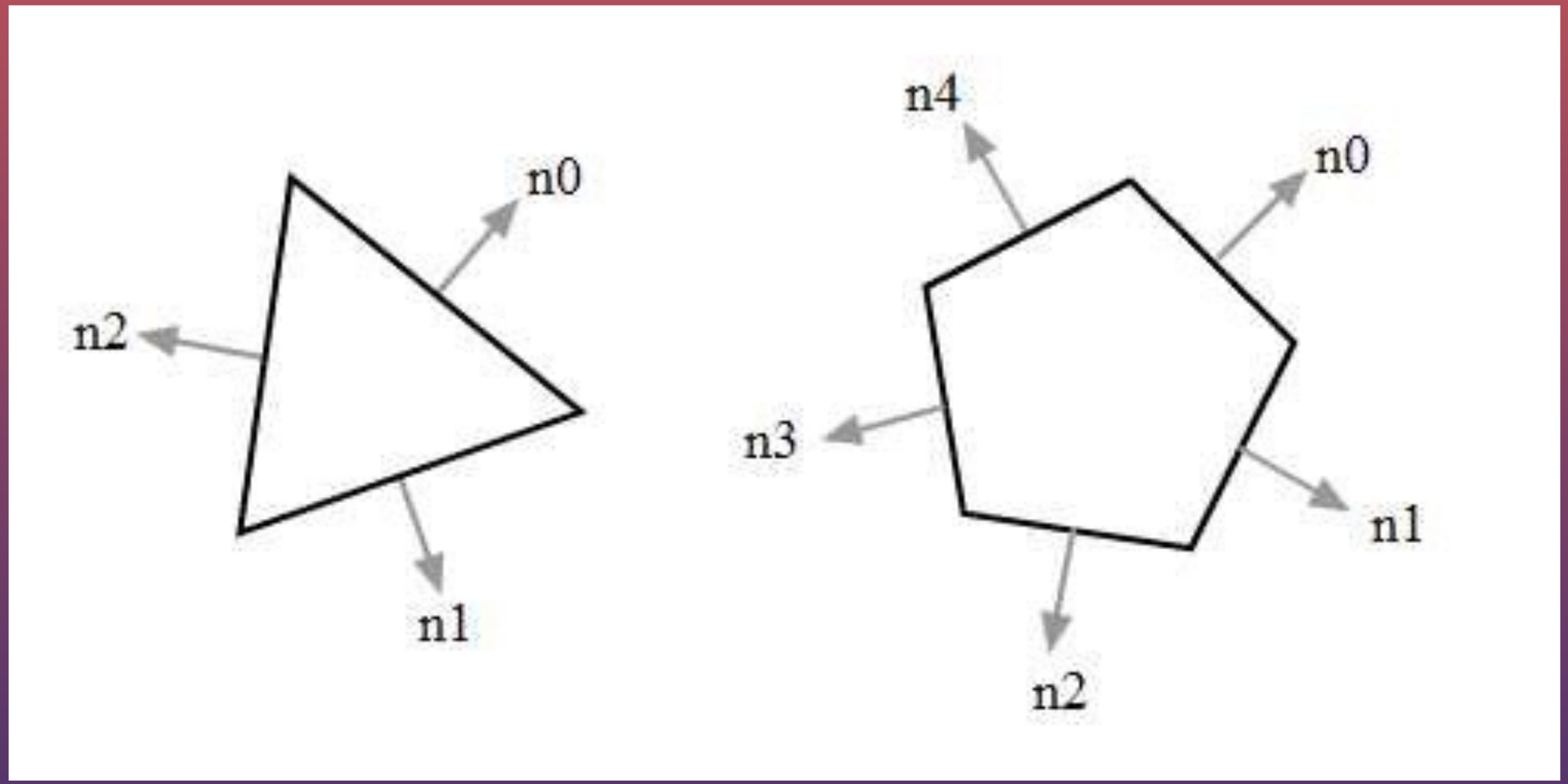
Texture coordinates.

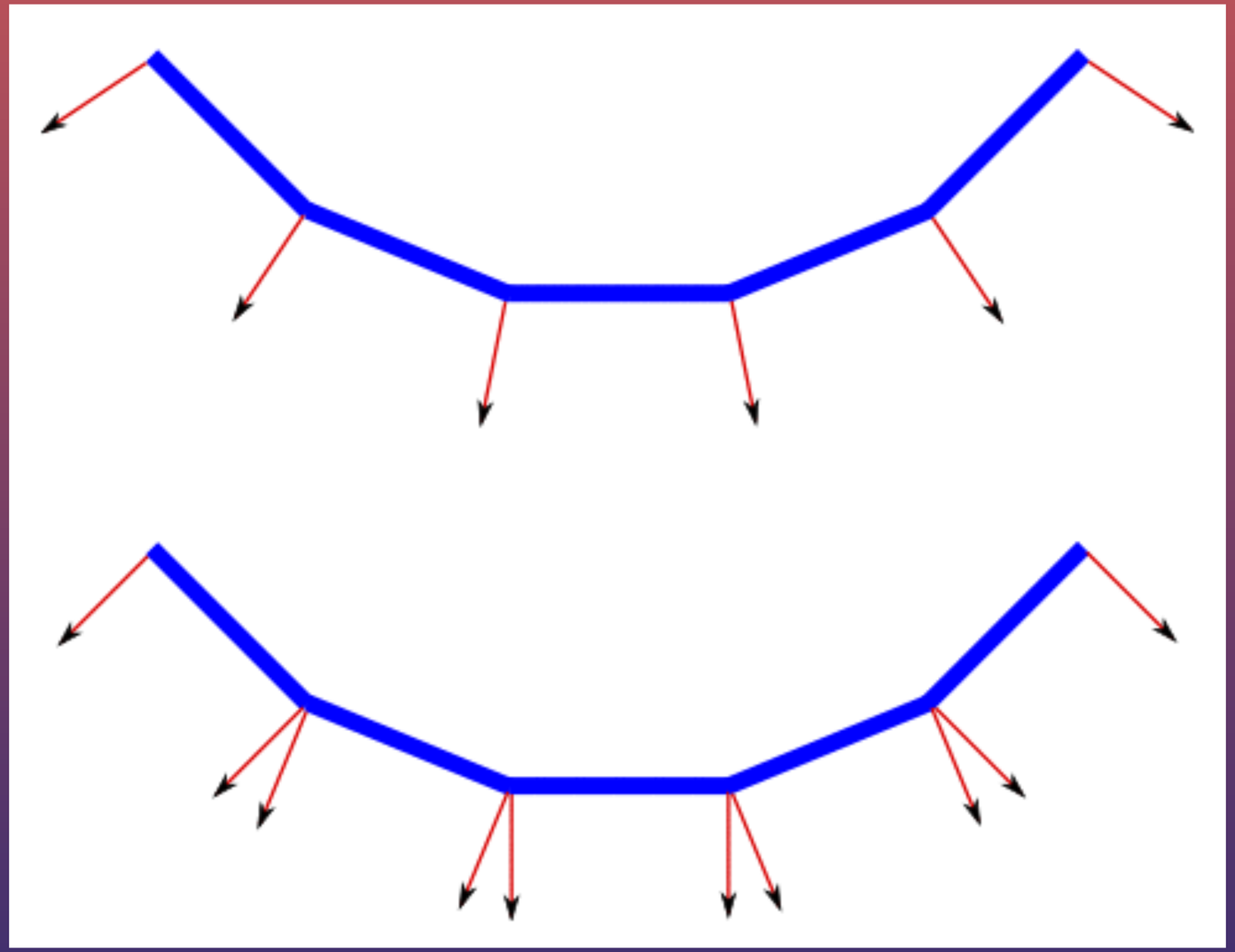

0,1
1,1
T (V)
0,0
S (U)
1,0

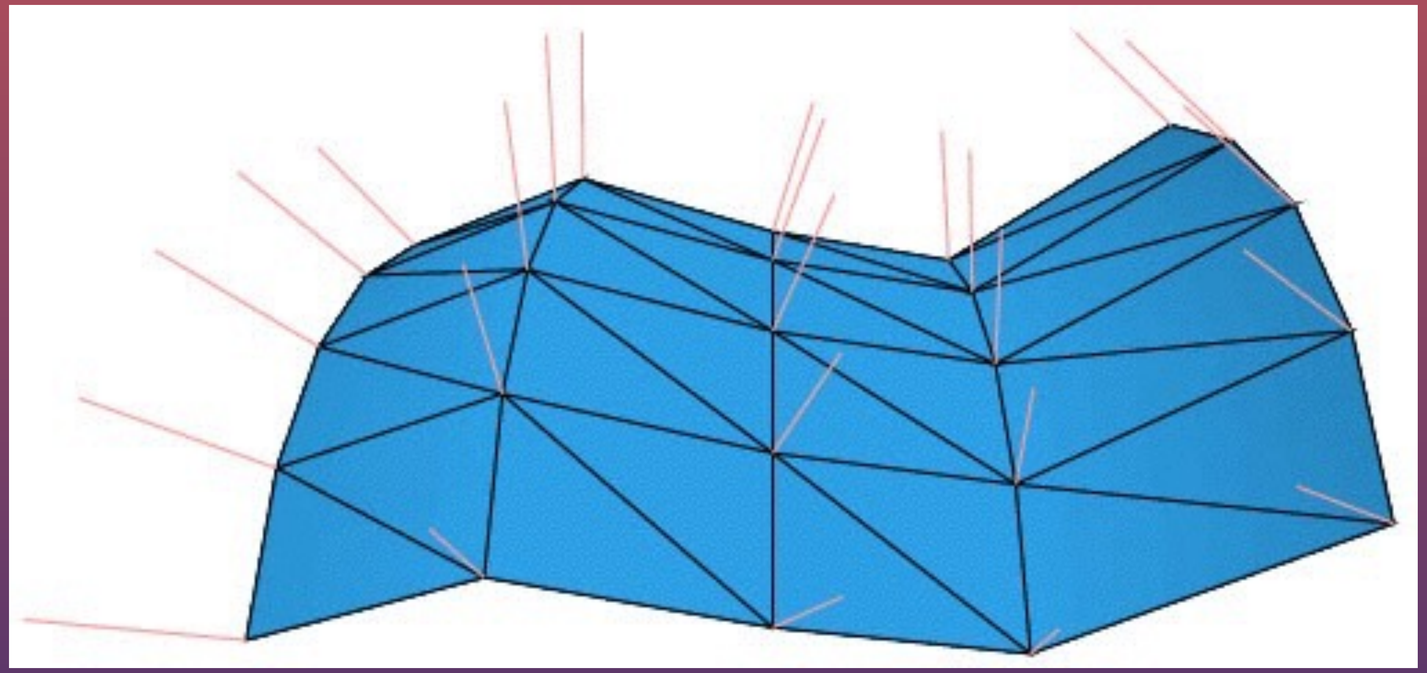


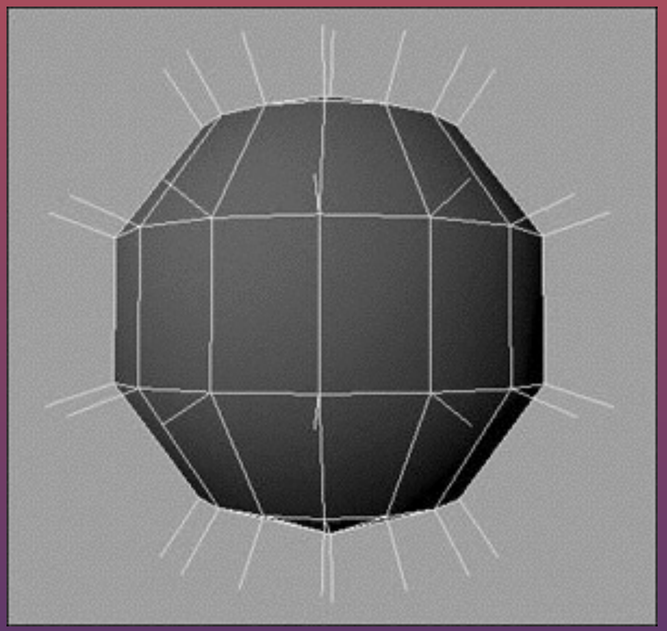


Vertex normals.


n0
n2
n1
n4
n0
n3
n1
n2







```
class Mesh {
    public:

        Mesh();
        void Render();

        std::vector<float> vertices;
        std::vector<float> uvs;
        std::vector<float> normals;
};


void Mesh::Render() {
    glVertexPointer(3, GL_FLOAT, 0, vertices.data());
    glEnableClientState(GL_VERTEX_ARRAY);

    glTexCoordPointer(2, GL_FLOAT, 0, uvs.data());
    glEnableClientState(GL_TEXTURE_COORD_ARRAY);

    glDrawArrays(GL_TRIANGLES, 0, vertices.size()/3);
}
```

Loading meshes from OBJ files.

Basic OBJ file structure

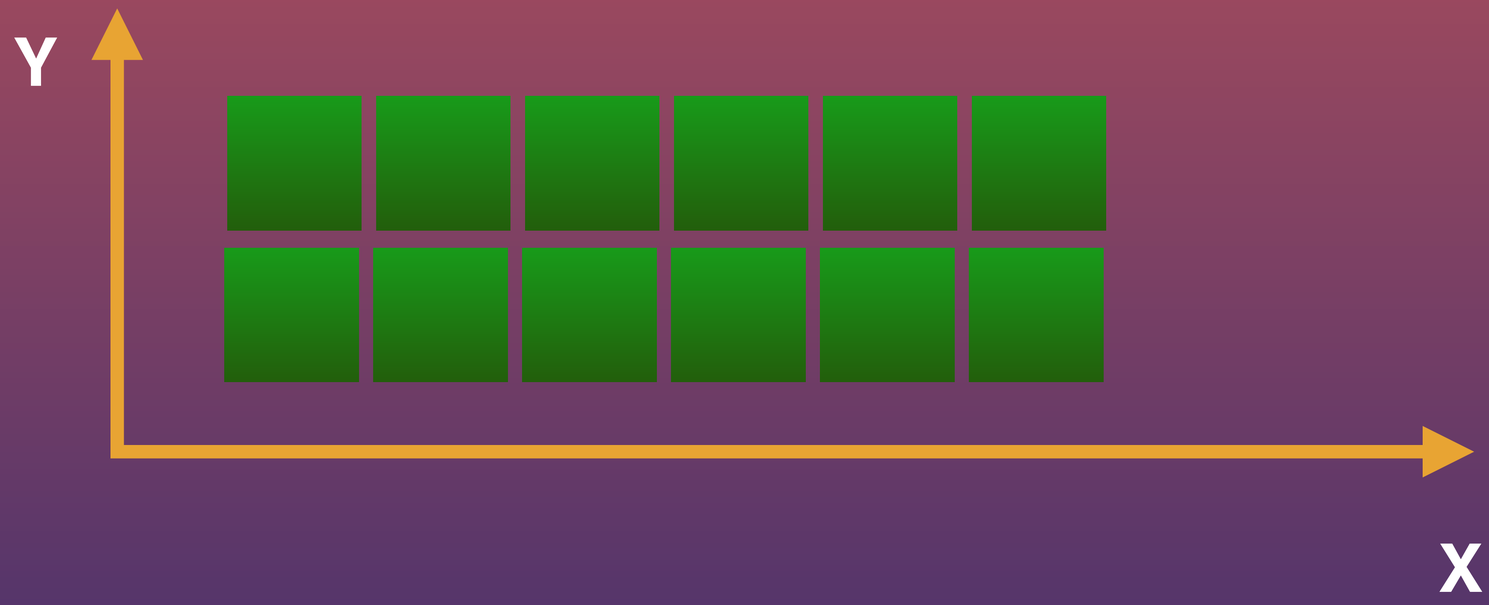
Indexes are 1-based, V is inverse in Texture Coordinates.

```
# comment

# vertex
v -0.436826 -0.208037 1.060532

# texture coordinate
vt 0.859375 0.500000

# vertex normal
vn 0.707 0.000 0.707

# indices
# can reference vertex
# or vertex / texture coordinate
# or vertex / texture coordinate / normal
f 21/10/3 16/11/5 1/7/2
```

```
void Mesh::loadOBJ(const char *fileName) {
    ifstream infile(fileName);
    string line;
    std::vector<float> fileVertices;
    std::vector<float> fileUVs;
    std::vector<float> fileNormals;
    while (getline(infile, line)) {
        istringstream sStream(line);
        string token;
        getline(sStream, token, ' ');
        if(token == "v") {
            while(getline(sStream, token, ' ')) {
                fileVertices.push_back(atof(token.c_str()));
            }
        } else if(token == "vn") {
            while(getline(sStream, token, ' ')) {
                fileNormals.push_back(atof(token.c_str()));
            }
        } else if(token == "vt") {
            while(getline(sStream, token, ' ')) {
                fileUVs.push_back(atof(token.c_str()));
            }
        } else if(token == "f") {
            while(getline(sStream, token, ' ')) {
                istringstream faceStream(token);
                string faceToken;
                int type = 0;
                while(getline(faceStream, faceToken, '/')) {
                    int index = atoi(faceToken.c_str())-1;
                    switch(type) {
                        case 0:
                            vertices.push_back(fileVertices[index*3]);
                            vertices.push_back(fileVertices[(index*3)+1]);
                            vertices.push_back(fileVertices[(index*3)+2]);
                        break;
                        case 1:
                            uvs.push_back(fileUVs[(index*2)]);
                            uvs.push_back(1.0f - fileUVs[(index*2)+1]);
                        break;
                        case 2:
                            normals.push_back(fileNormals[(index*3)]);
                            normals.push_back(fileNormals[(index*3)+1]);
                            normals.push_back(fileNormals[(index*3)+2]);
                        break;
                    }
                    type++;
                }
            }
        }
    }
}
```

Loading a 2D tilemap level in 3D

In 2D



In 3D