Introduction to

OpenGL|ES™

the open graphics library for embedded systems

PRESENTATION TOPICS

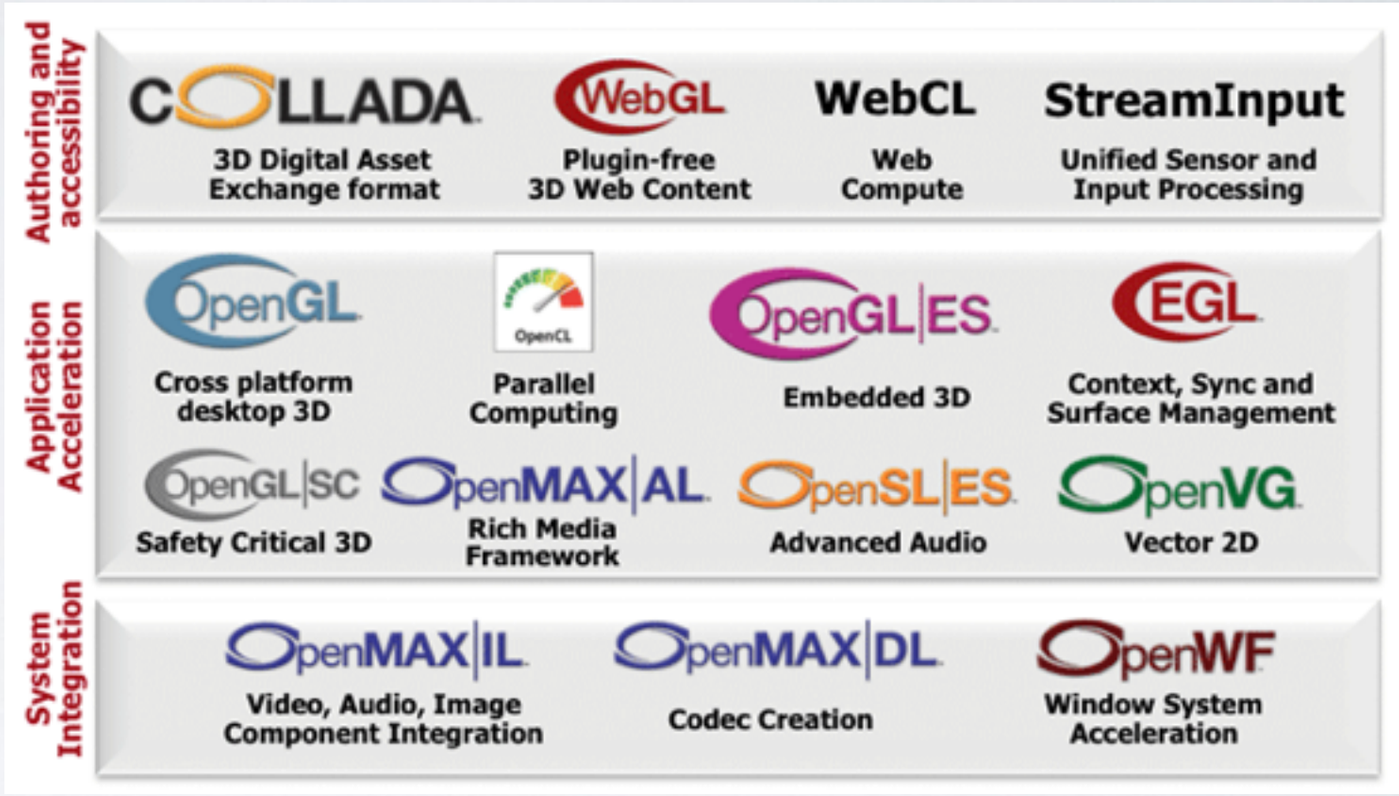
- Introduction to OpenGL ES
- Spaces and Transformations
- Drawing Geometry
- Colors
- Textures
- Android Fragmentation
- OpenGL ES 2.0

"OpenGL (Open Graphics Library) is a cross-language, multi-platform application programming interface (API) for rendering 2D and 3D vector graphics. The API is typically used to interact with a graphics processing unit (GPU), to achieve hardware-accelerated rendering."

– Le Wikipedia

OPENGL ES

- Developed by the Khronous Group



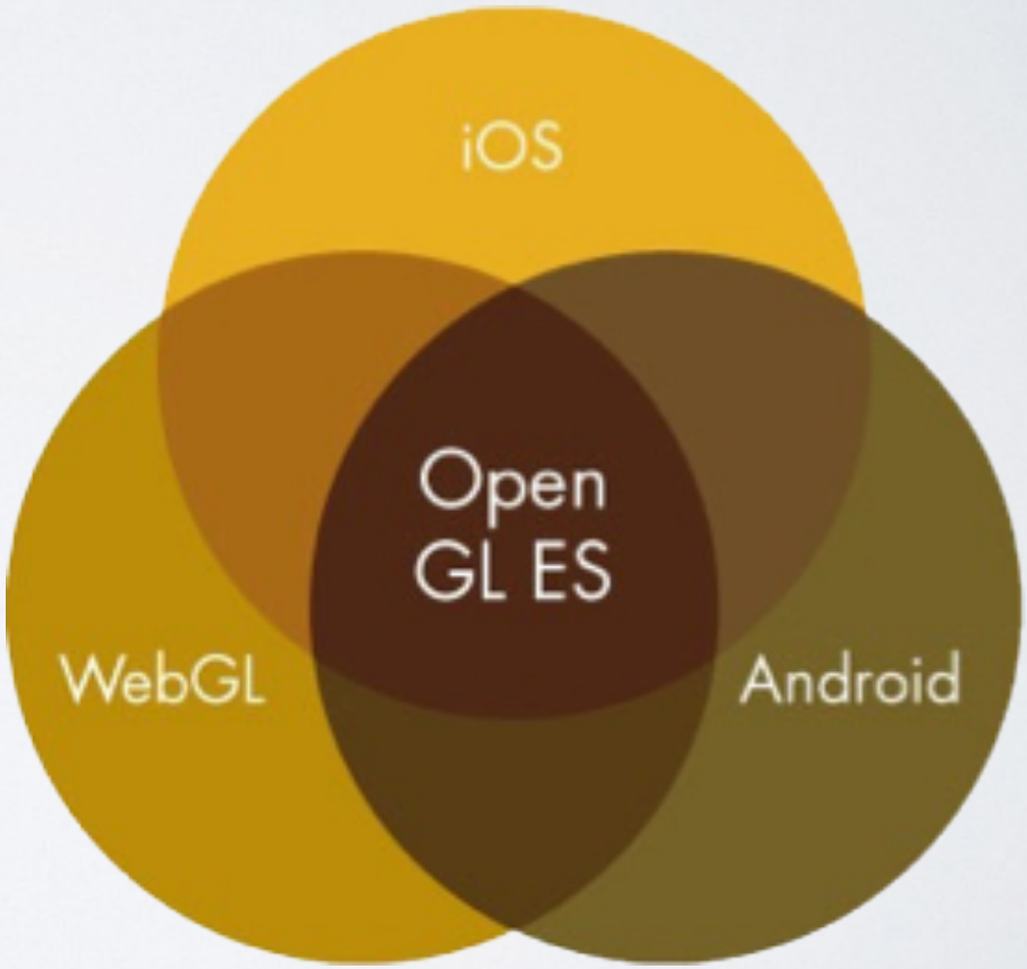
- Software interface for hardware

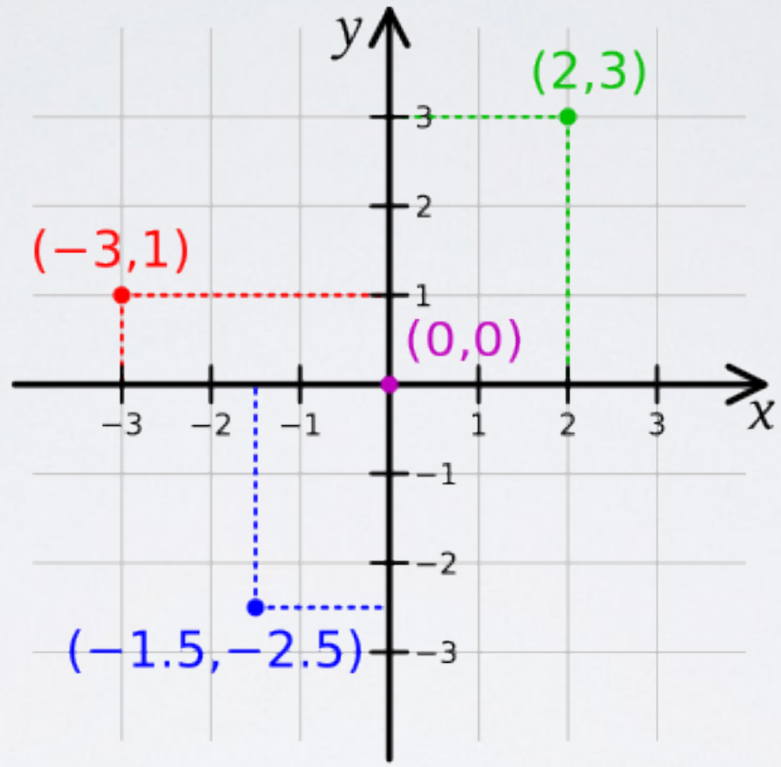
- Subset of the original OpenGL

- A lot of material on the internet

OPENGL ES

- A lot of fragmentation on android (we will cover it later)
- State Machine
- Errors are hard to get



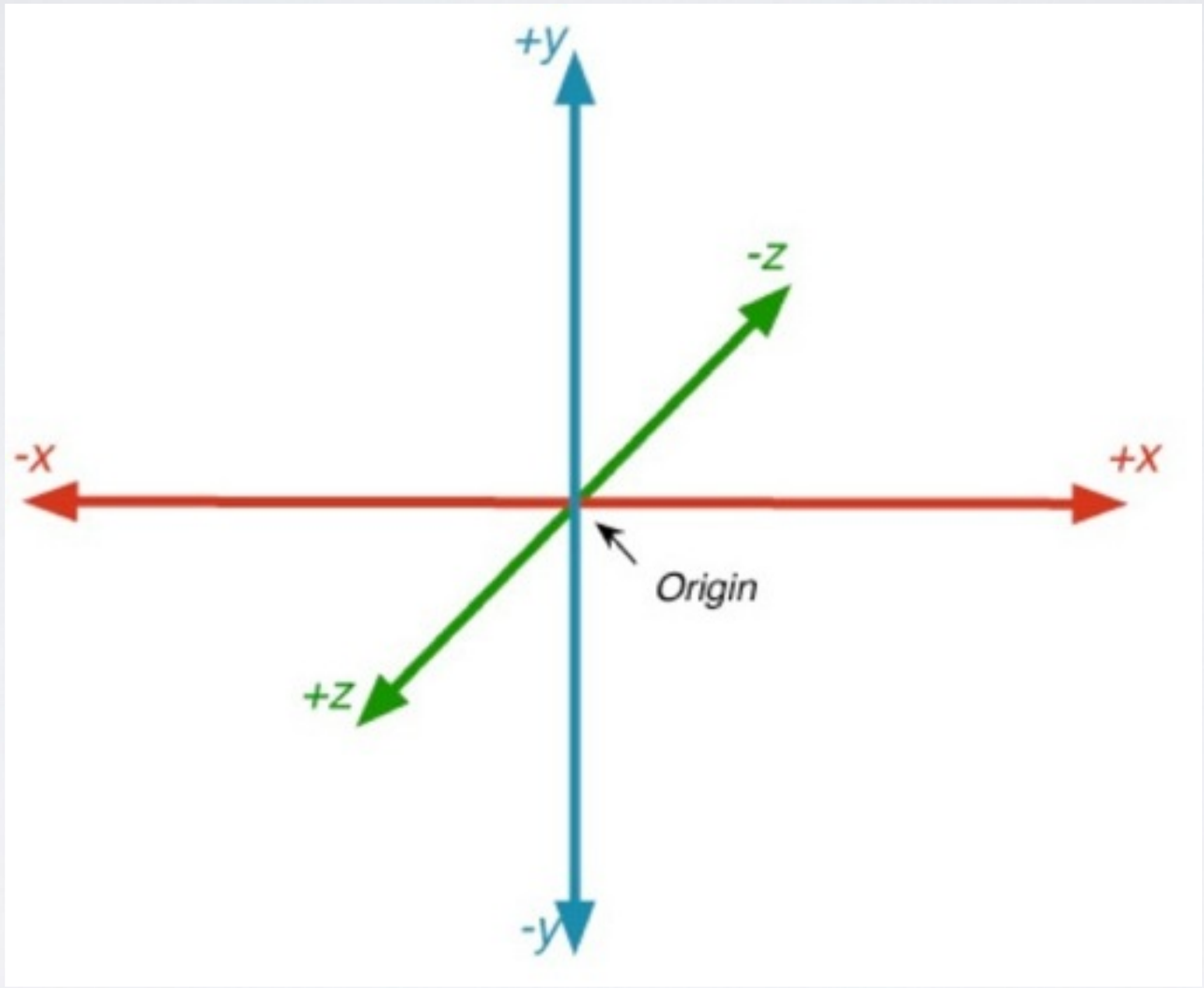


SPACES

SPACES (OR COORDINATES)

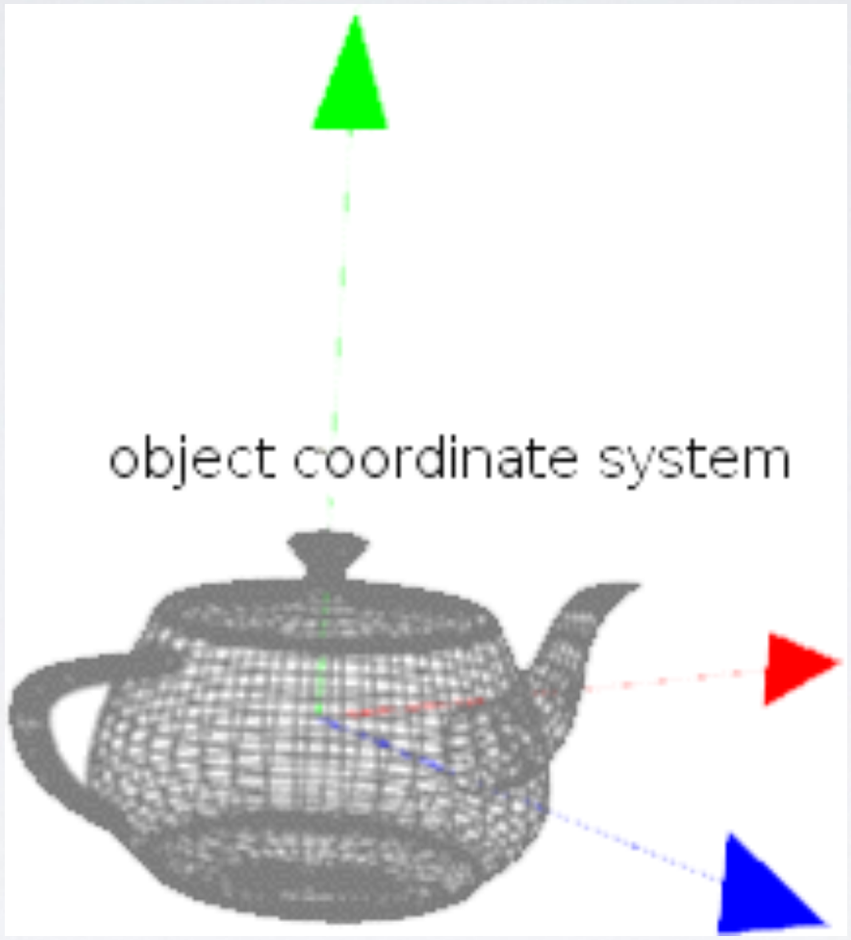
- Window Space
- Clip and Eye Space
- World Space
- Object Space

WORLD SPACE



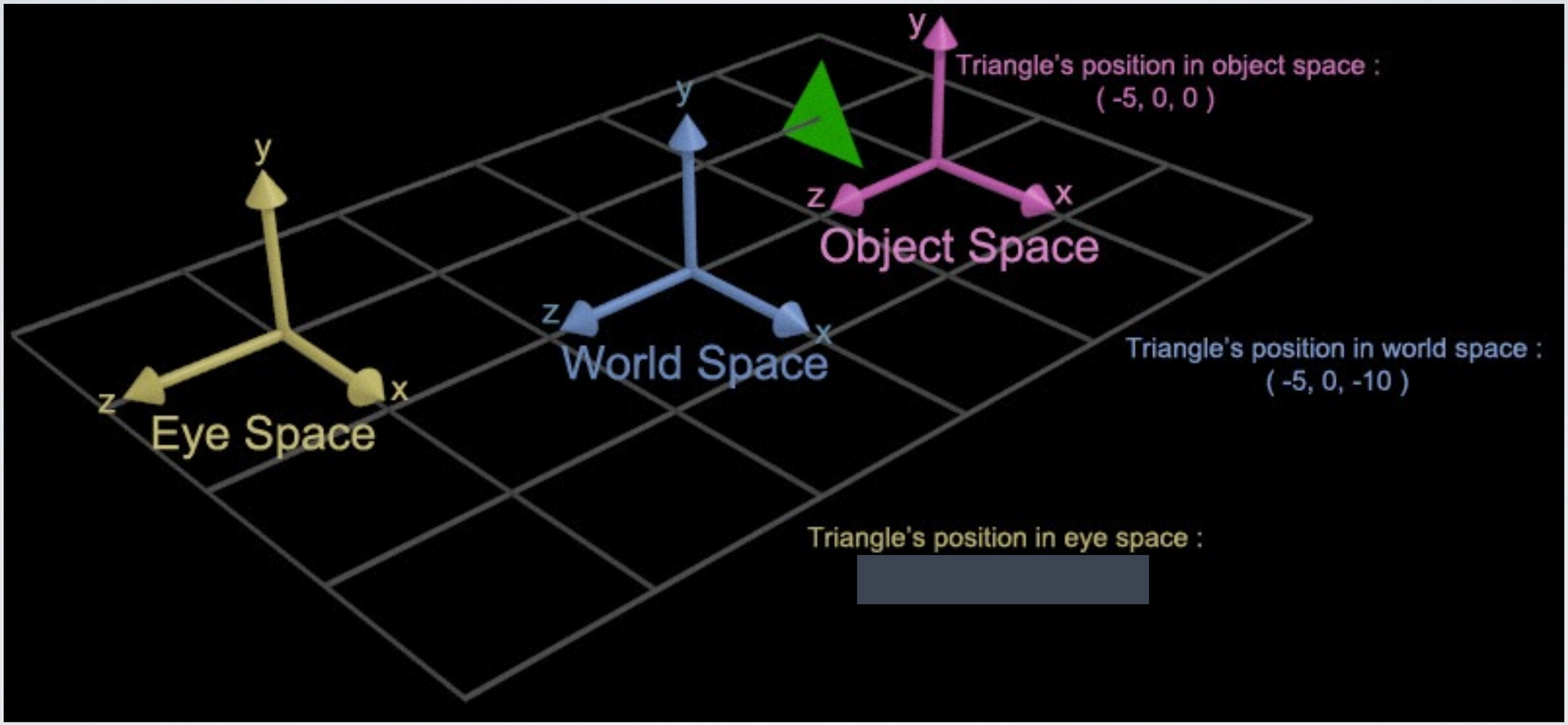
The world coordinates are the most used by developers to build the scene.
They are also called Model Space.

OBJECT SPACE



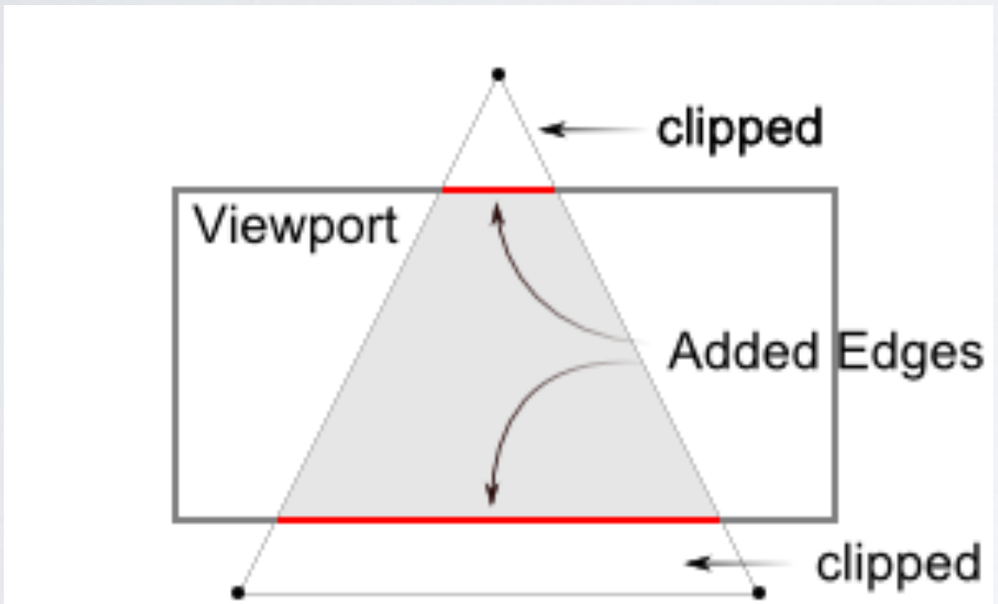
Object coordinates have the origin (0,0,0) at the object's world location.

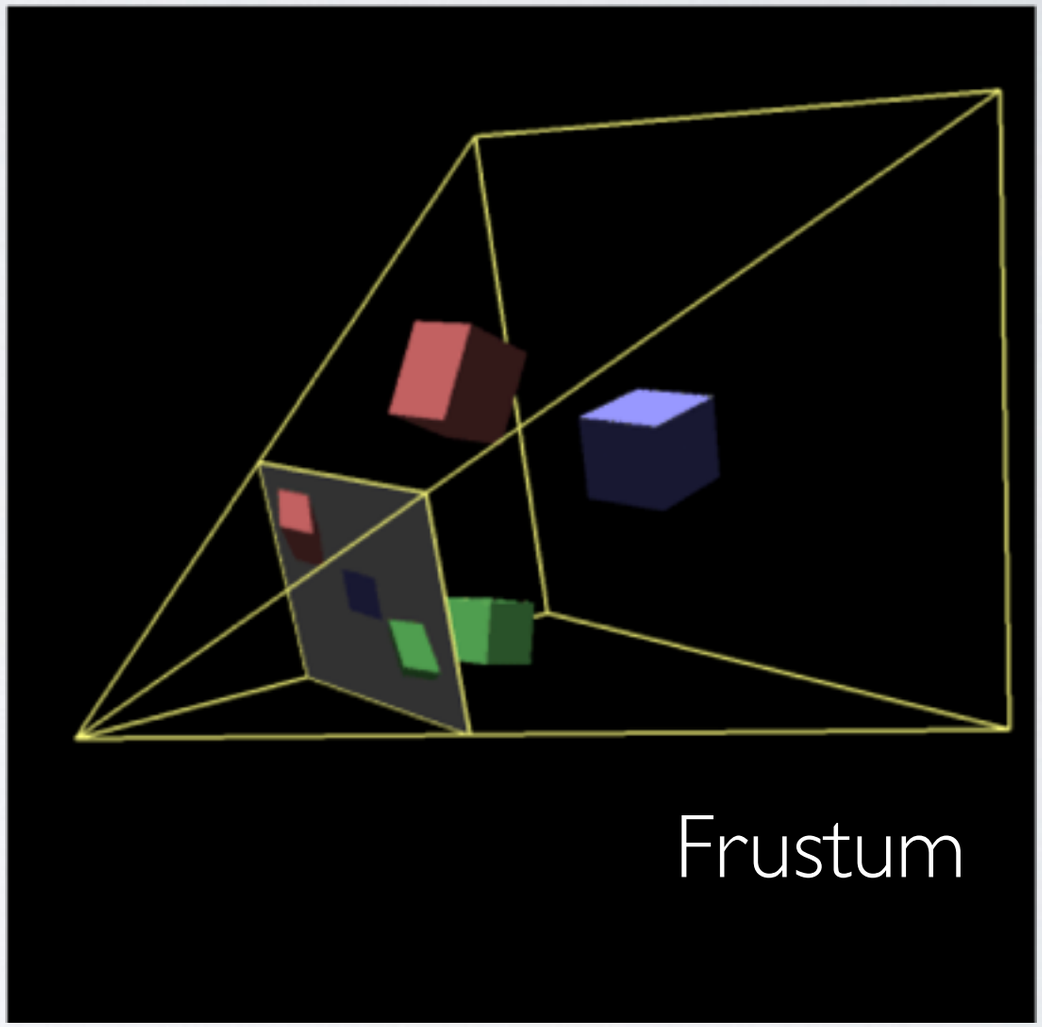
COORDINATES



The triangle's position relative to the object, world and eye spaces

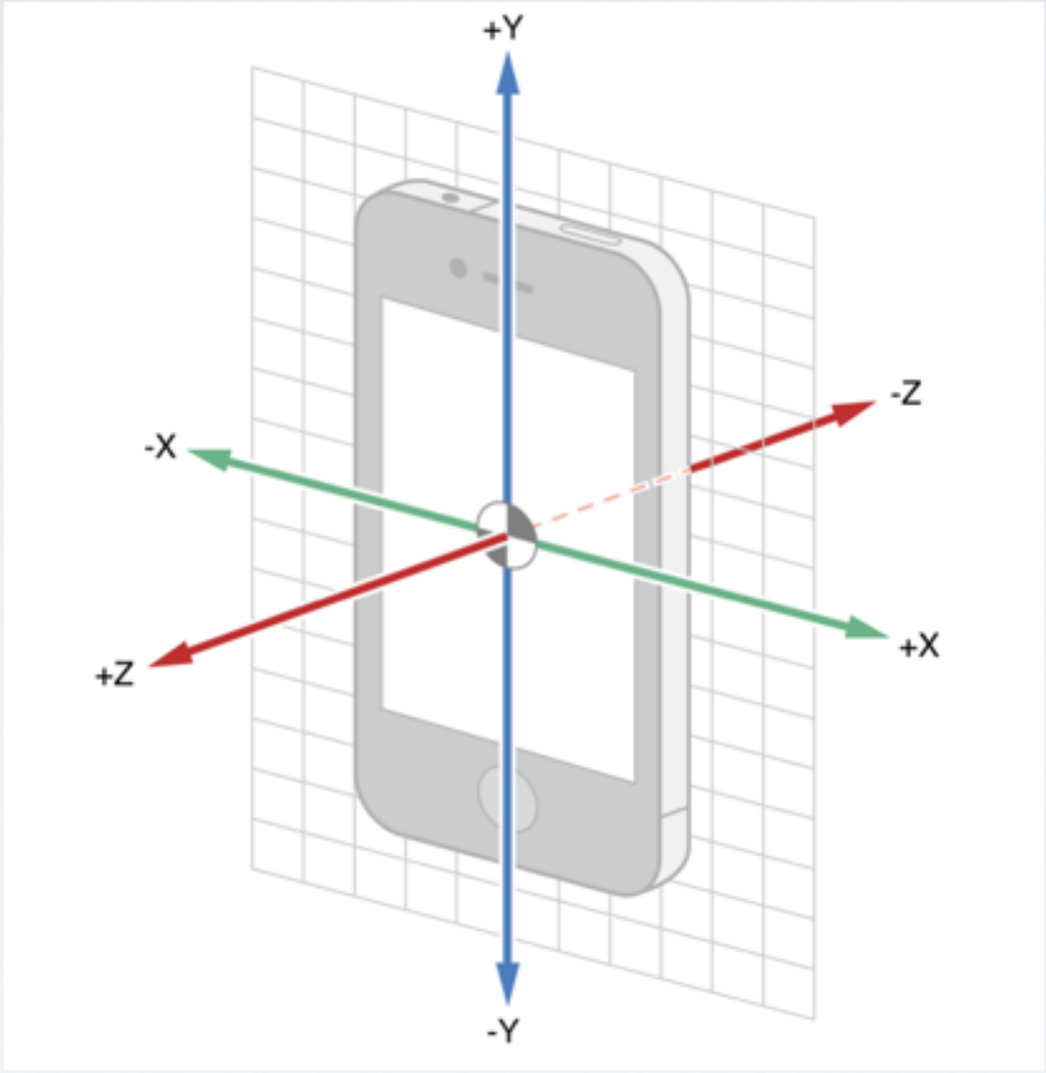
CLIP AND EYE SPACE





Eye coordinates are related to the camera. Clipping coordinates is the matrix that originates the view frustum.

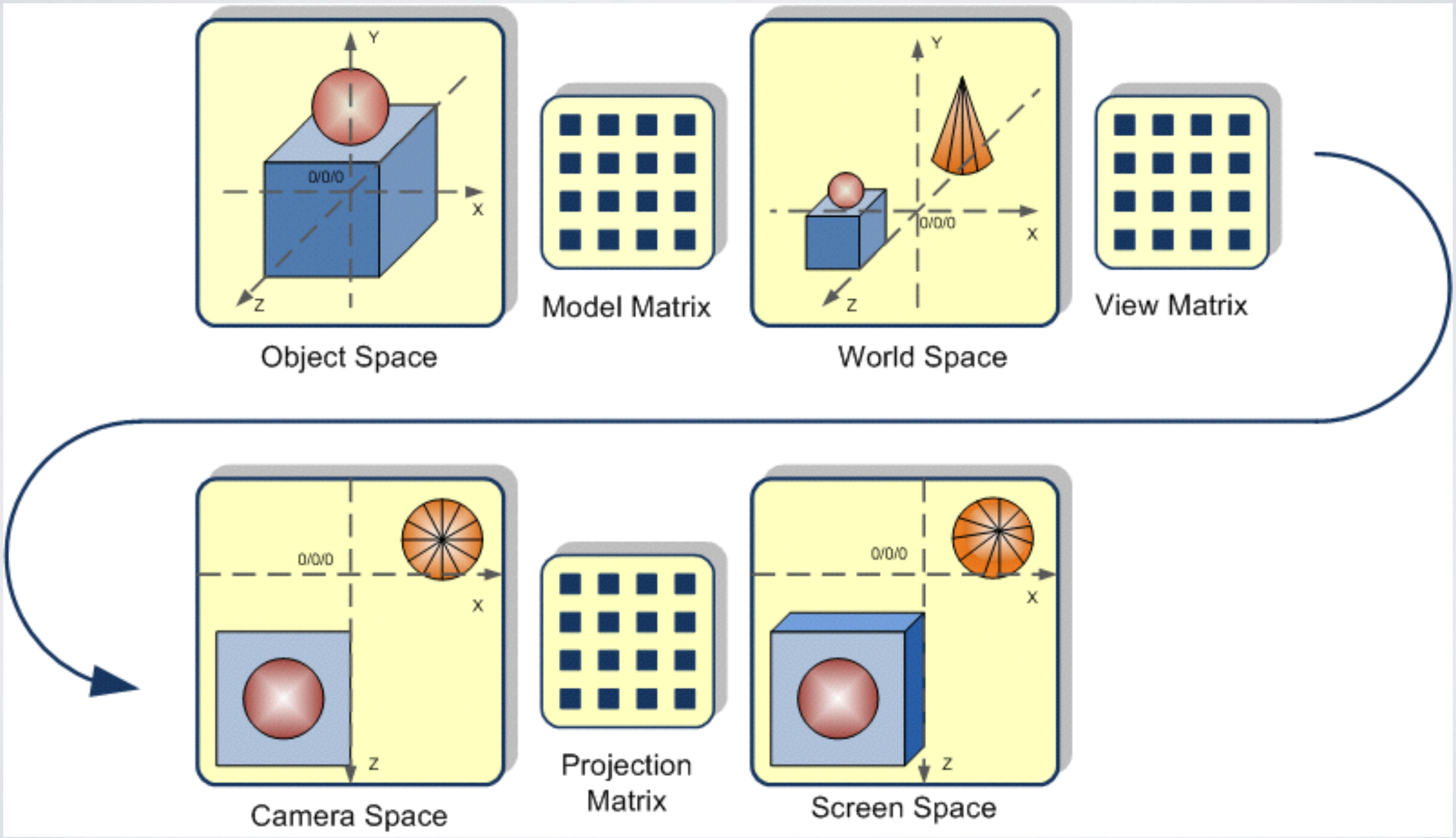
WINDOW SPACE



These are relative to device's screen and its origin is at the center of the device.

They can be normalized and range from -1 to +1 on every axis.

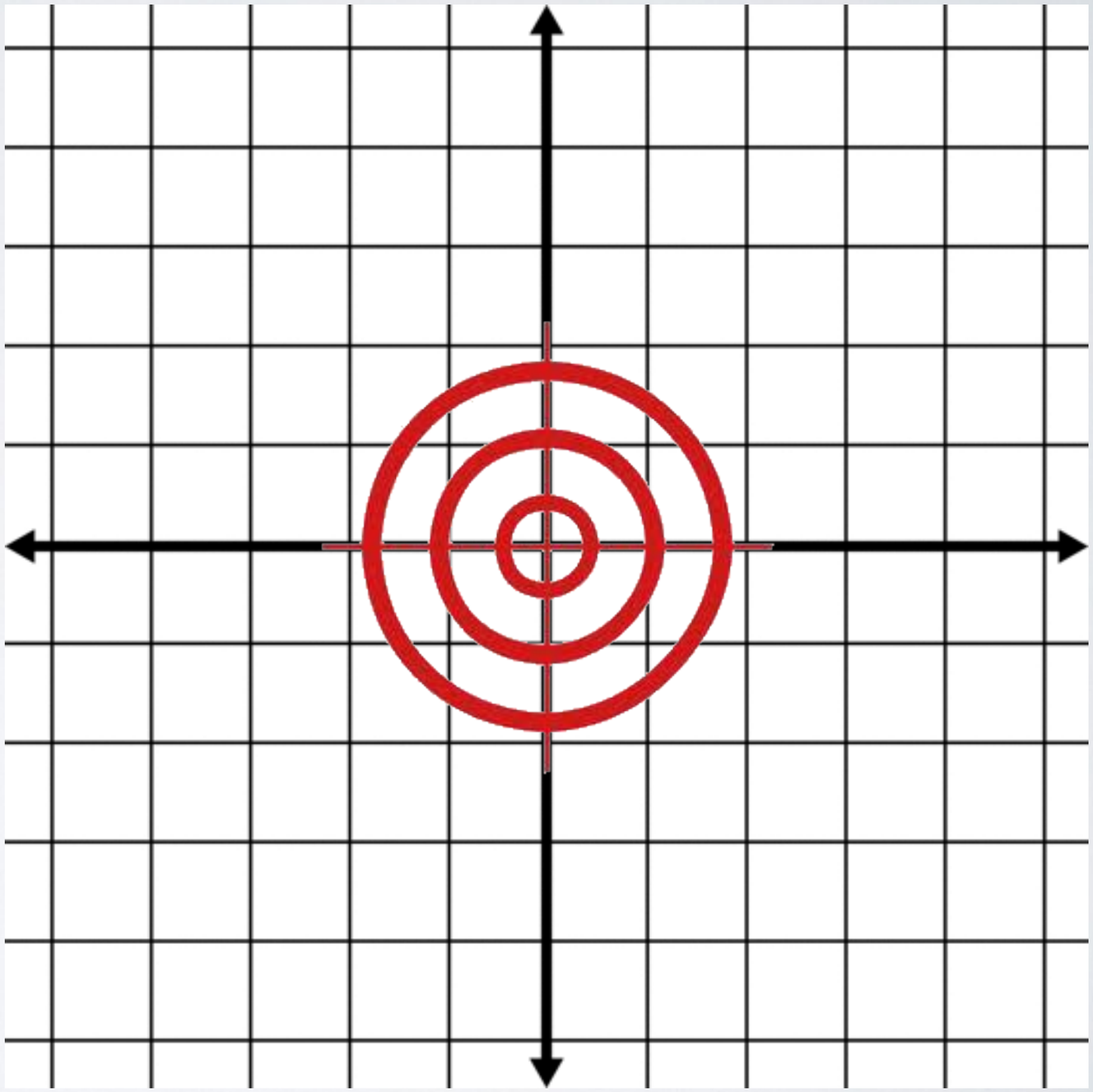
HOW IT FITS TOGETHER




OPENGL...
VERTICES EVERYWHERE
memegenerator.net

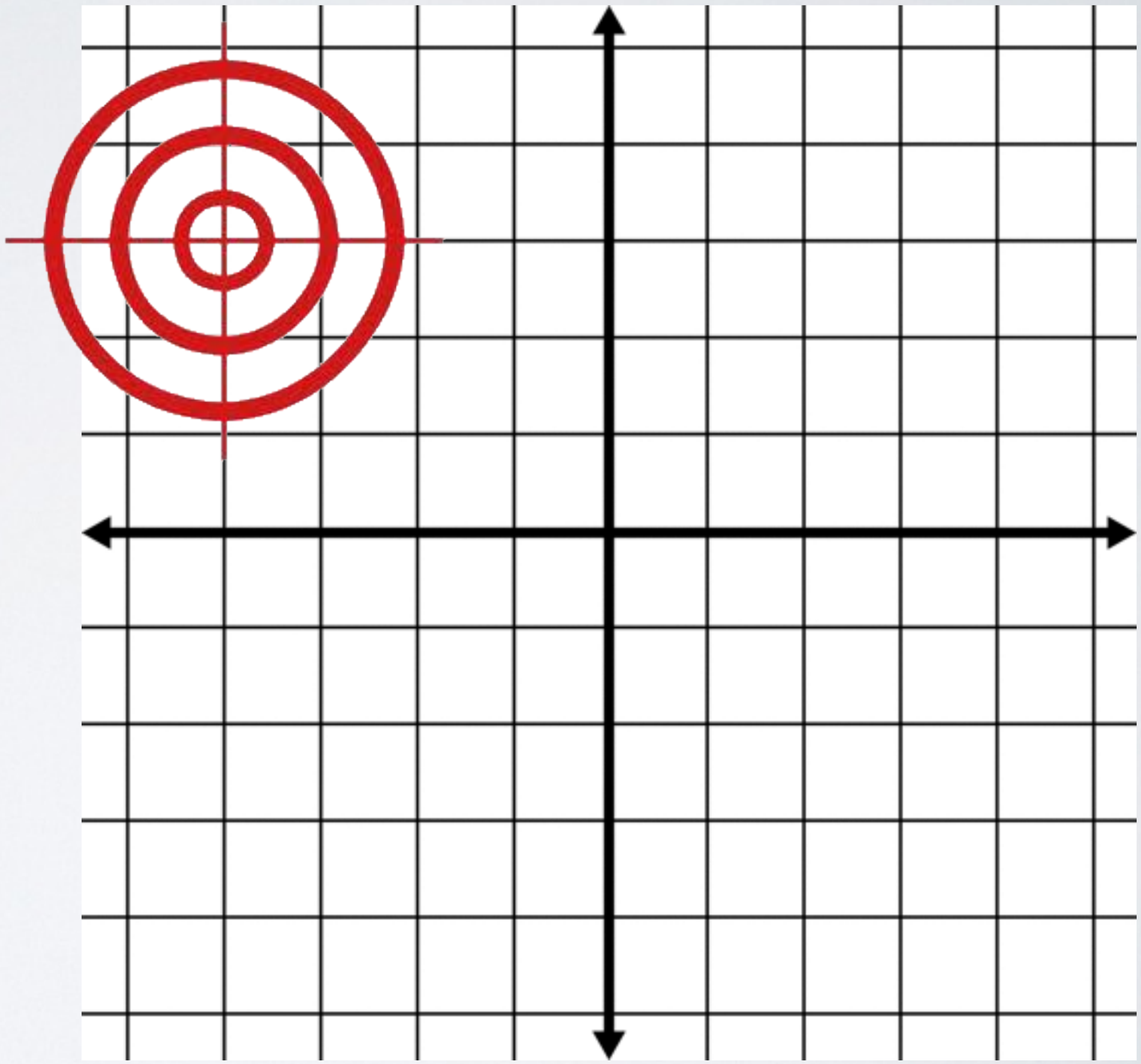
TRANSFORMATIONS

```
glLoadIdentity();
glPushMatrix();
```



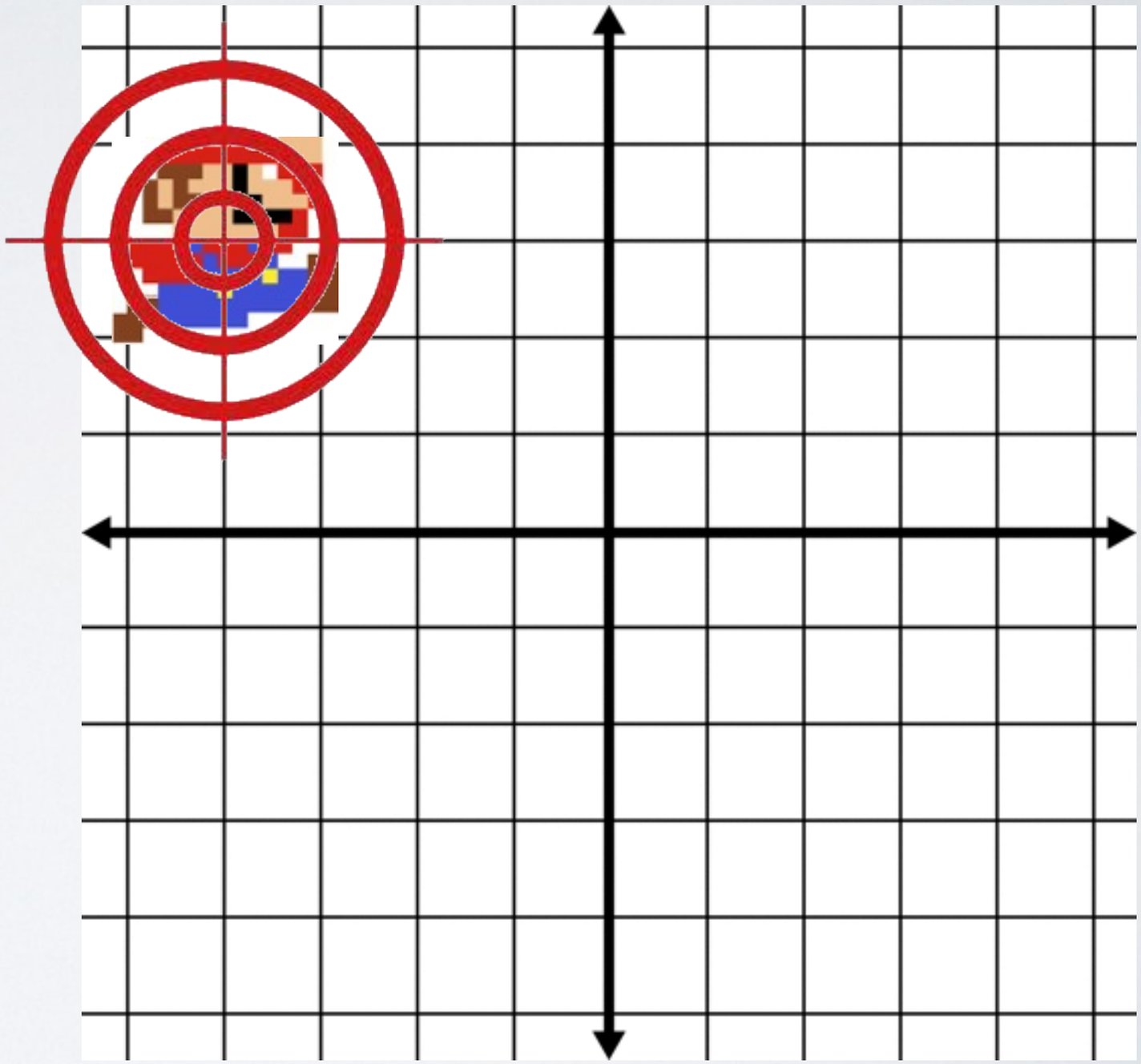
TRANSFORMATIONS

glLoadIdentity();
glPushMatrix();
glTranslate(-4,3);



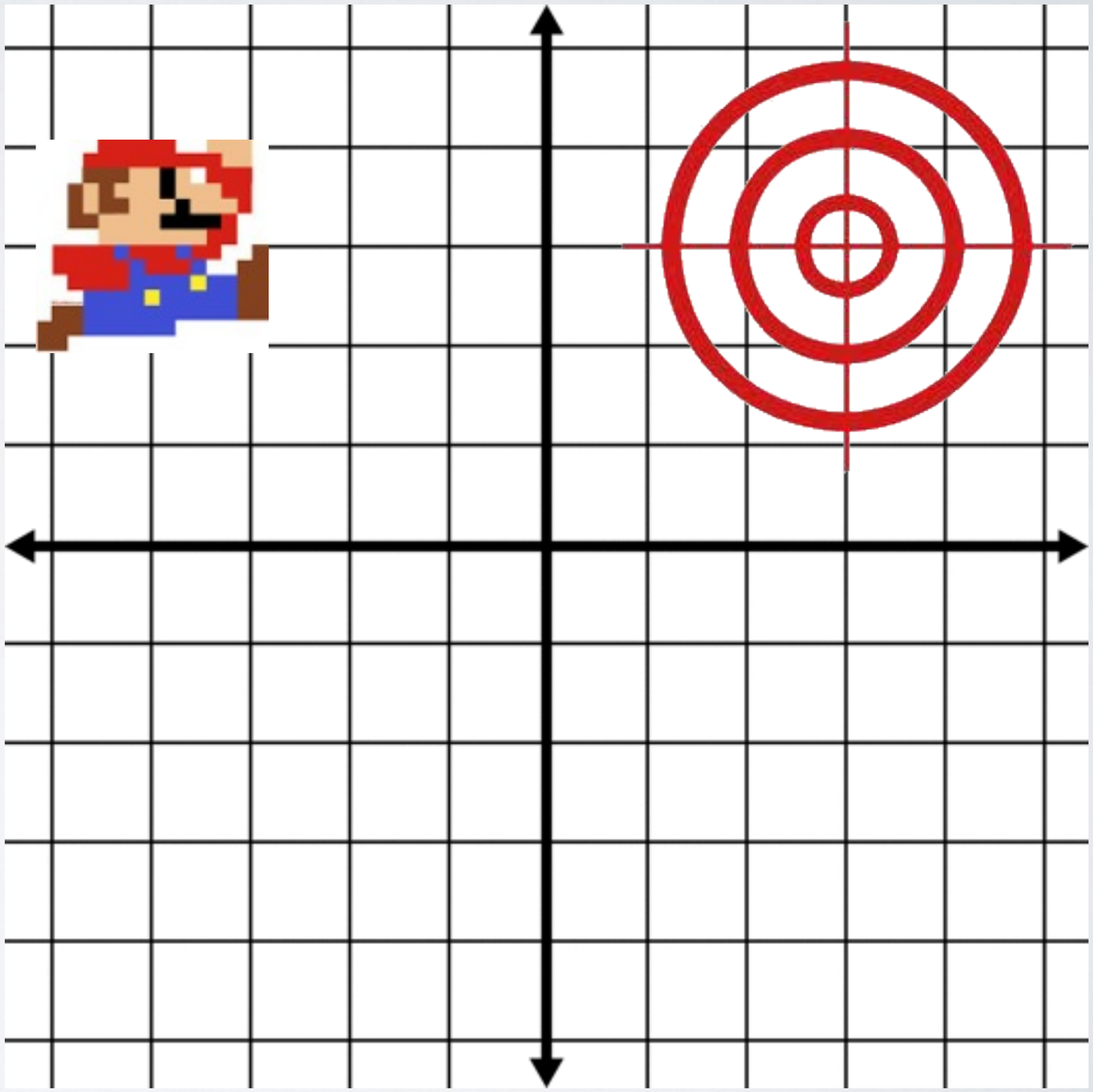
TRANSFORMATIONS

glLoadIdentity();
glPushMatrix();
glTranslate(-4,3);
drawMario();



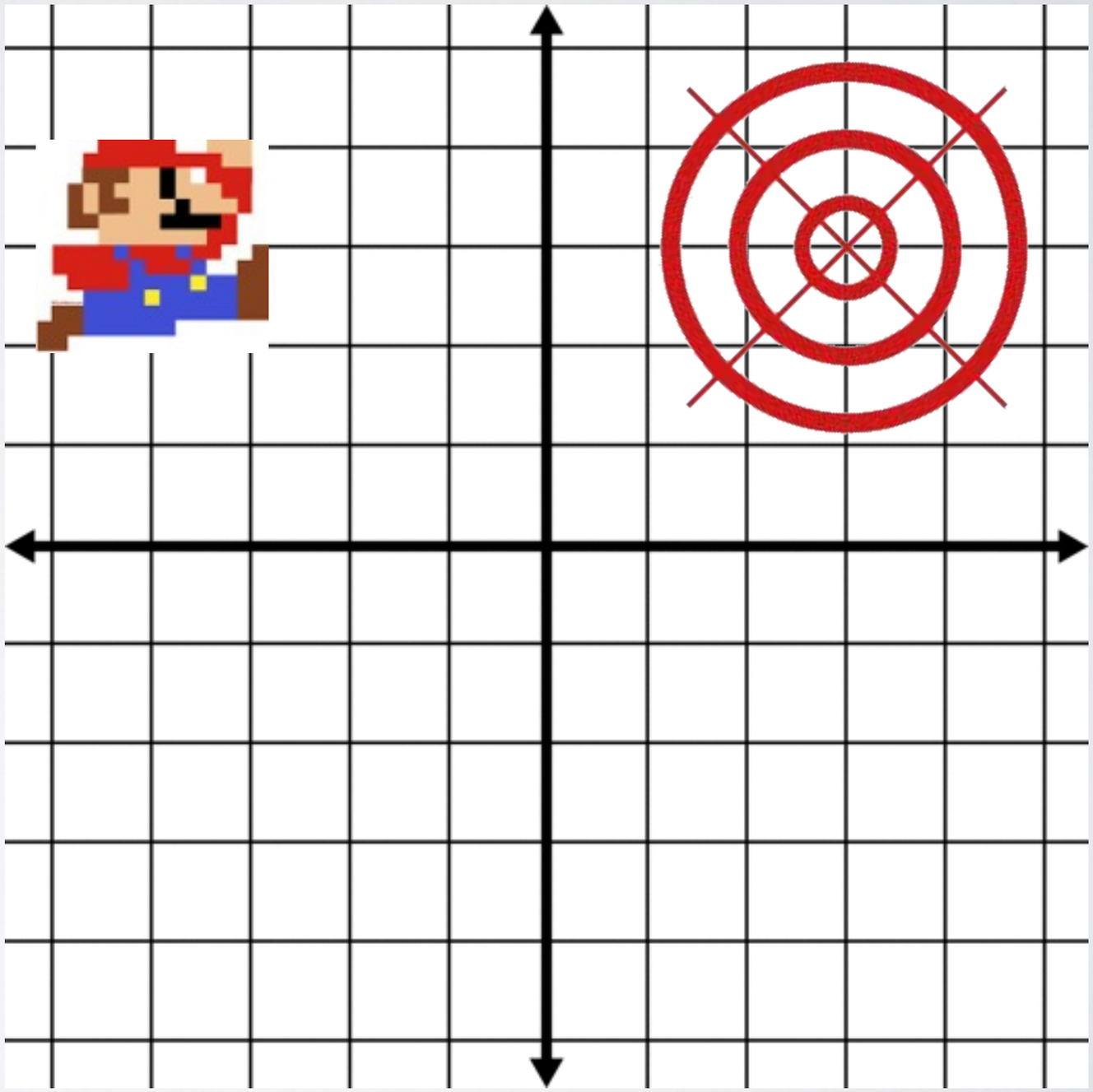
TRANSFORMATIONS

glLoadIdentity();
glPushMatrix();
glTranslate(-4,3);
drawMario();
glTranslate(7, 0);



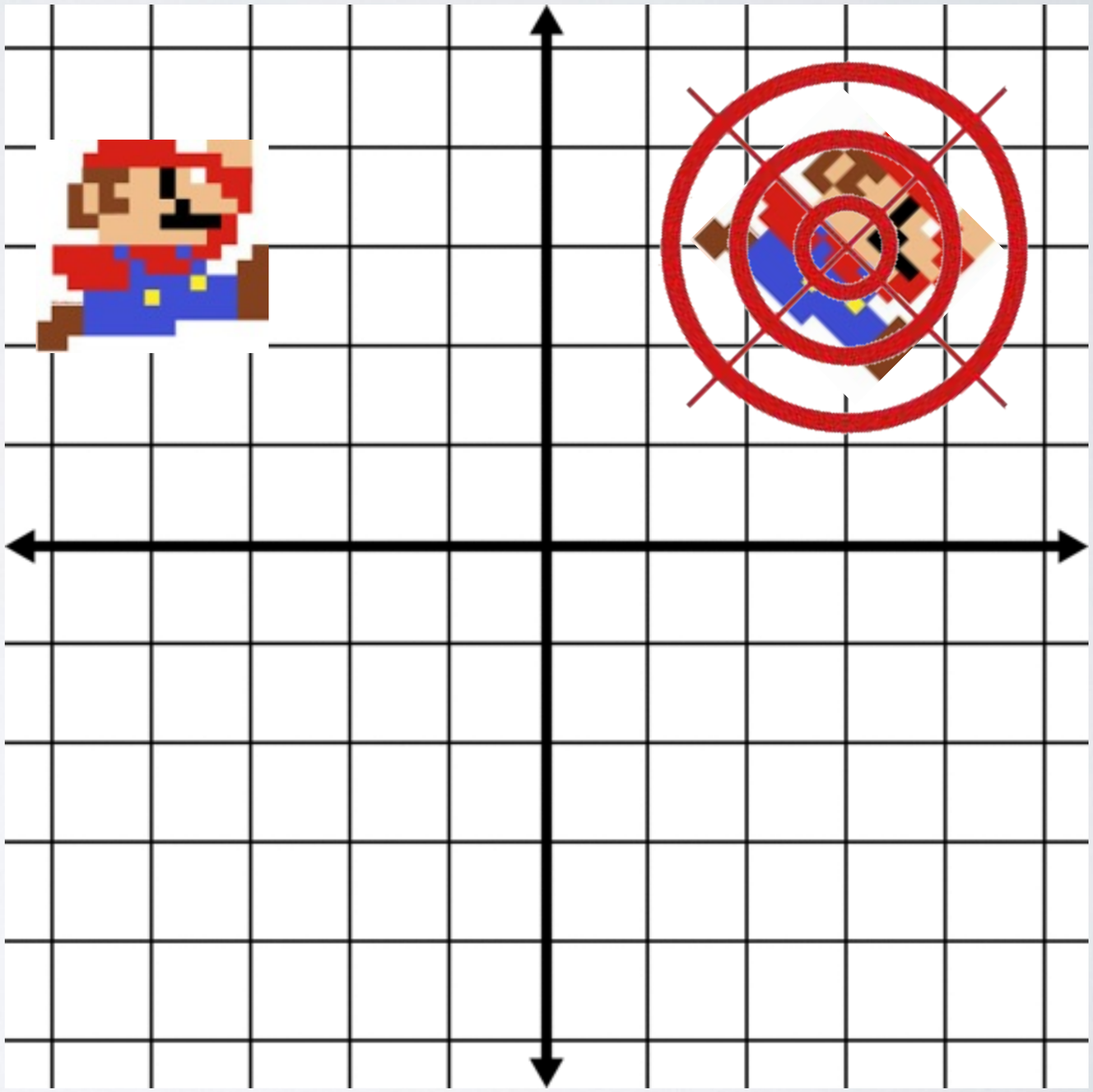
TRANSFORMATIONS

```
glLoadIdentity();
glPushMatrix();
glTranslate(-4,3);
drawMario();
glTranslate(7, 0);
glRotate(45, 0, 0, 1);
```



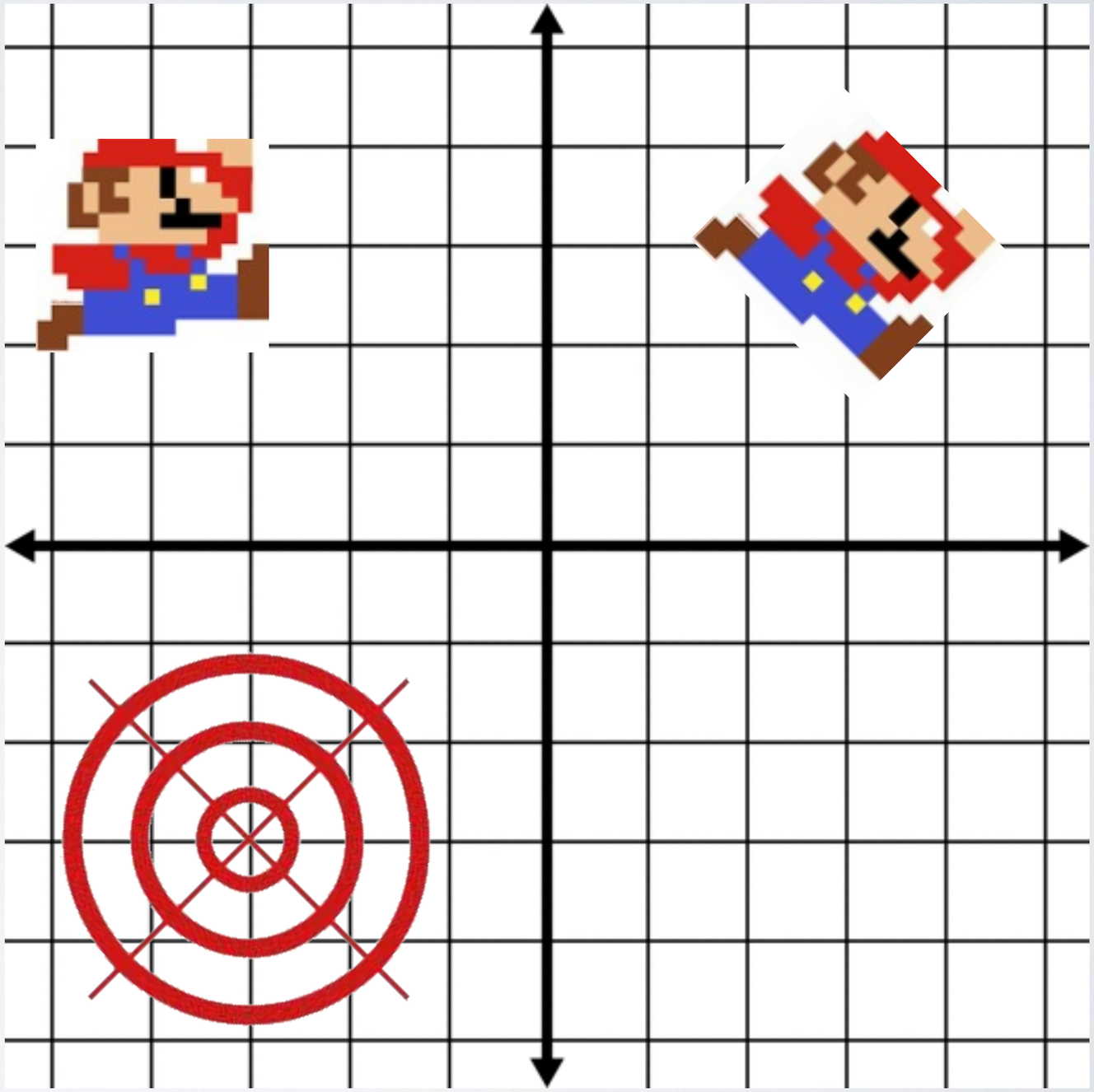
TRANSFORMATIONS

```
glLoadIdentity();
glPushMatrix();
glTranslate(-4,3);
drawMario();
glTranslate(7, 0);
glRotate(45, 0, 0, 0);
drawMario();
```



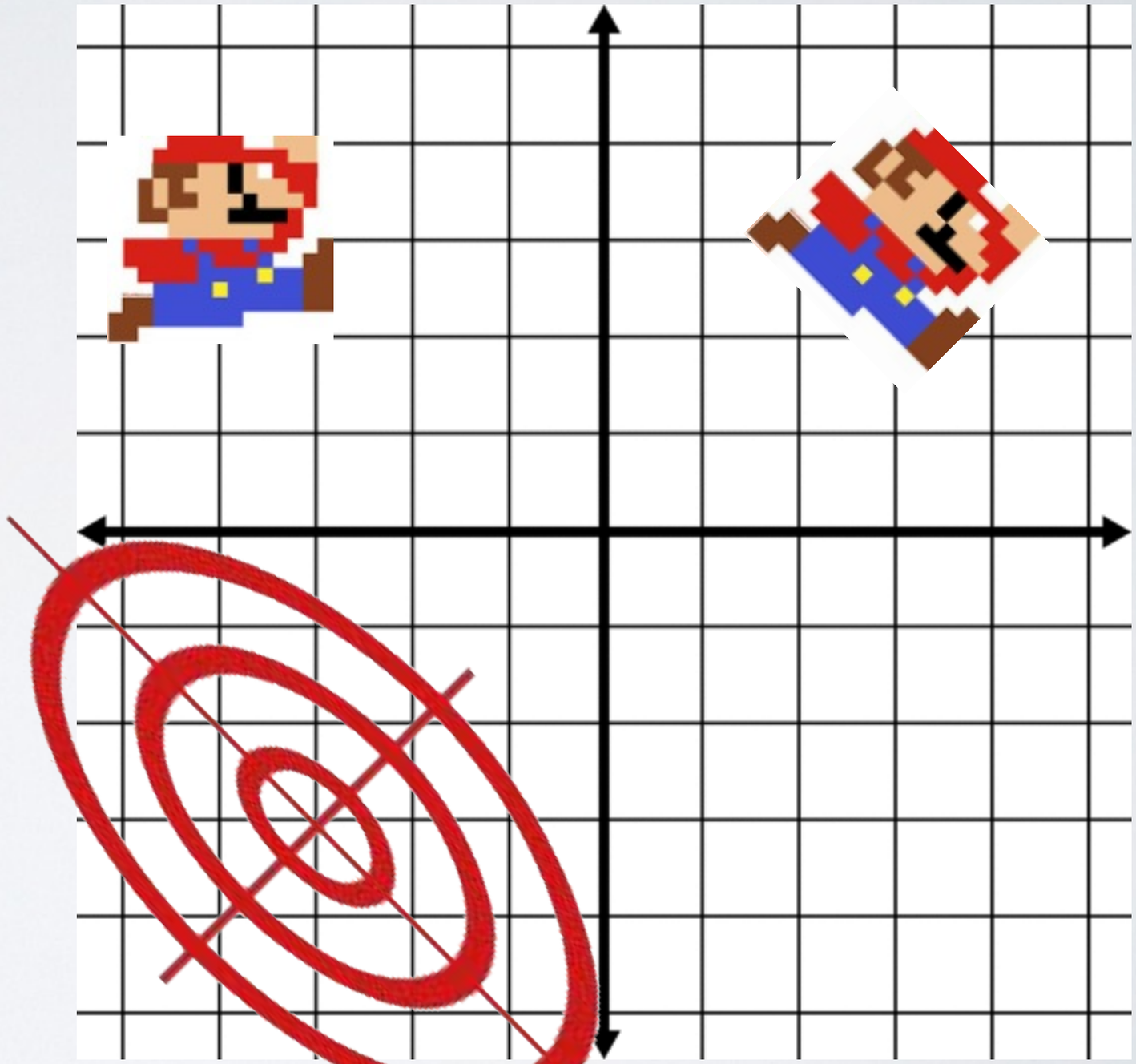
TRANSFORMATIONS

```
glLoadIdentity();
glPushMatrix();
glTranslate(-4,3);
drawMario();
glTranslate(7, 0);
glRotate(45, 0, 0, 0);
drawMario();
glTranslate(0, -6);
```



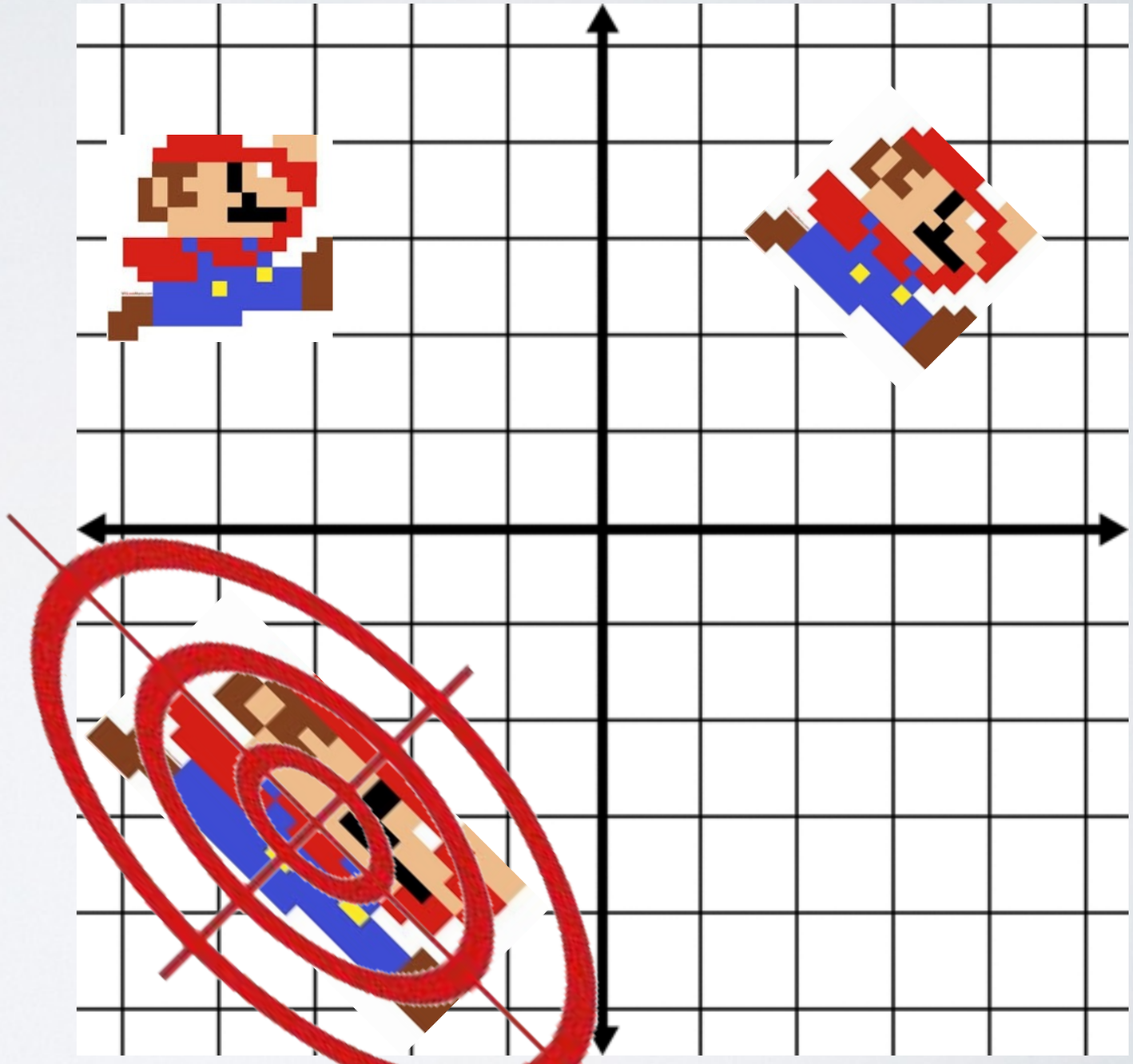
TRANSFORMATIONS

```
glLoadIdentity();
glPushMatrix();
glTranslate(-4,3);
drawMario();
glTranslate(7, 0);
glRotate(45, 0, 0, 0);
drawMario();
glTranslate(0, -6);
glScale(2,0);
```



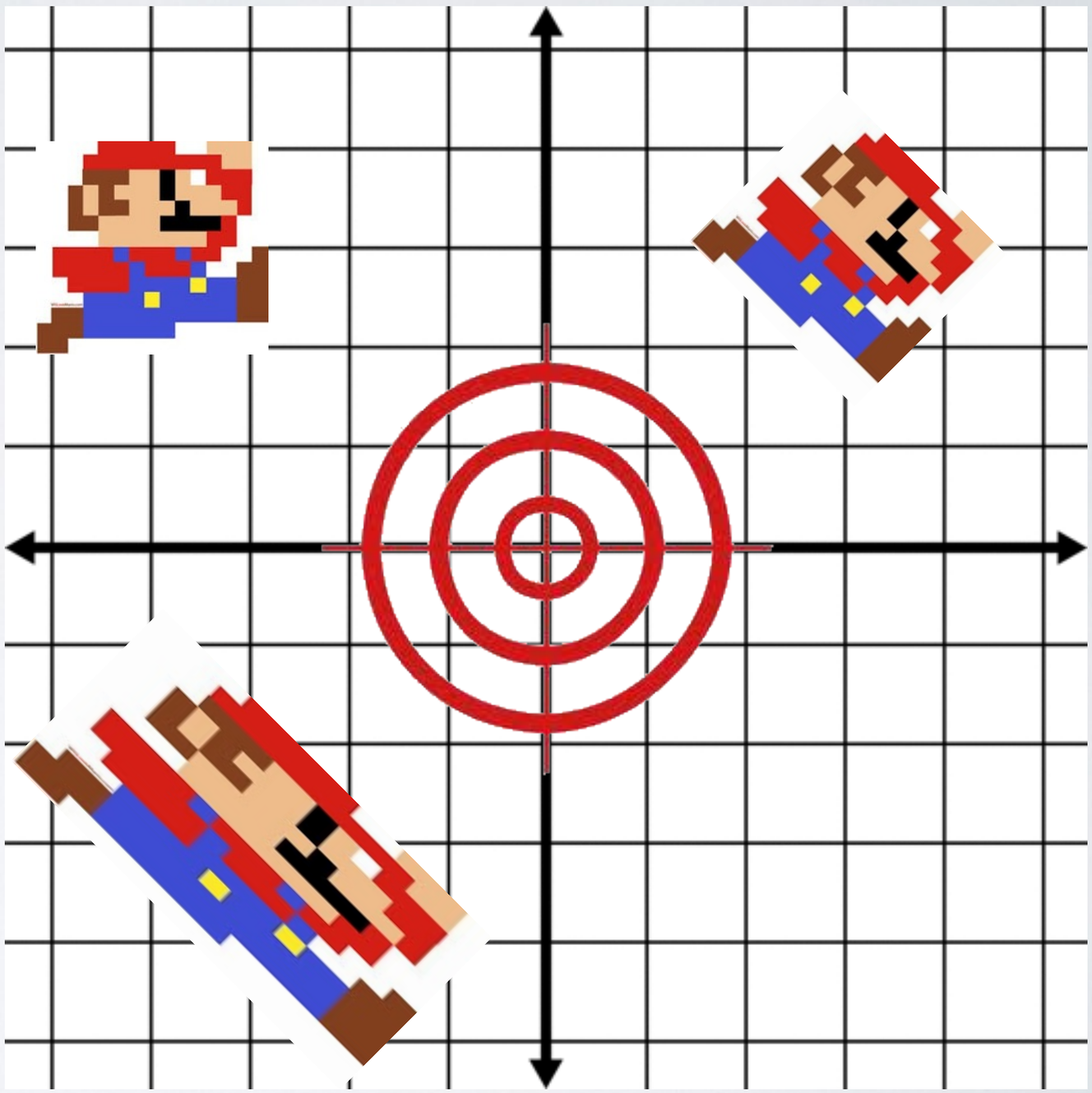
TRANSFORMATIONS

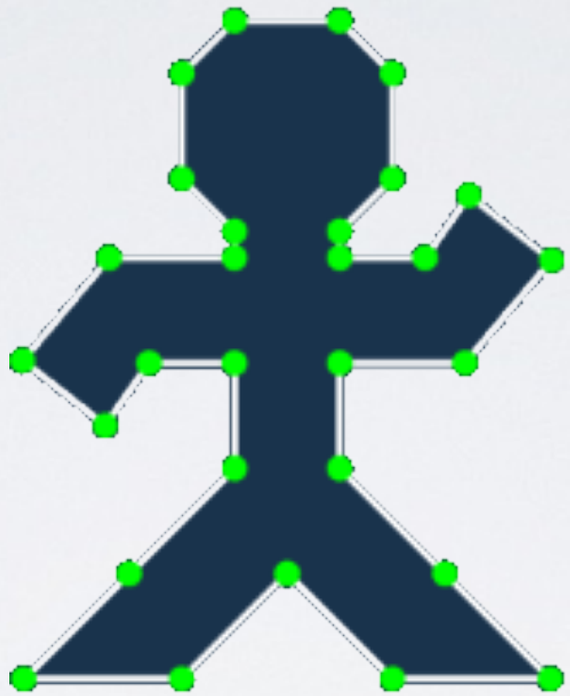
```
glLoadIdentity();
glPushMatrix();
glTranslate(-4,3);
drawMario();
glTranslate(7, 0);
glRotate(45, 0, 0, 0);
drawMario();
glTranslate(0, -6);
glScale(2,0);
drawMario();
```



TRANSFORMATIONS

```
glLoadIdentity();
glPushMatrix();
glTranslate(-4,3);
drawMario();
glTranslate(7, 0);
glRotate(45, 0, 0, 0);
drawMario();
glTranslate(0, -6);
glScale(2,0);
drawMario();
glPopMatrix();
```





GEOMETRY

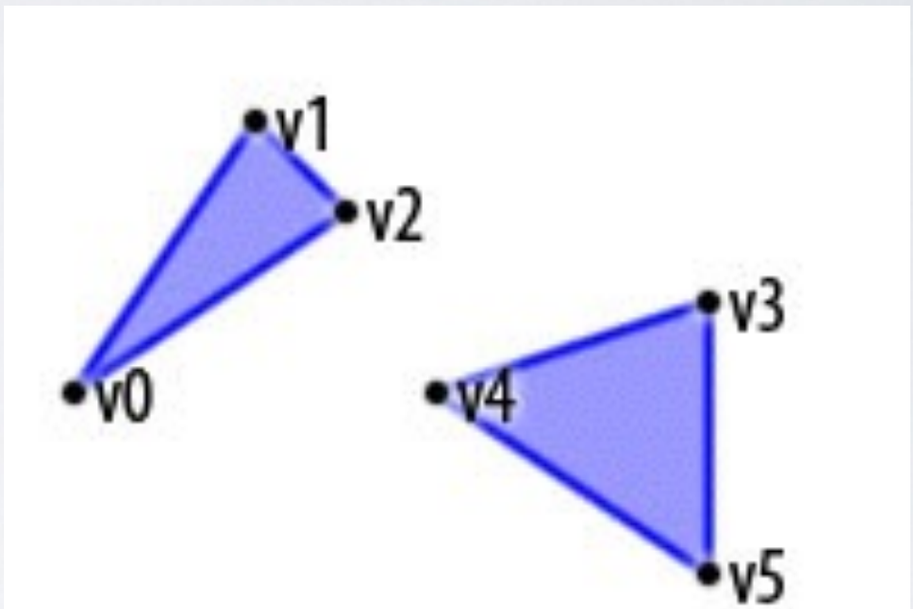
GEOMETRY

- In order to draw geometry, OpenGL receives a list of vertices.

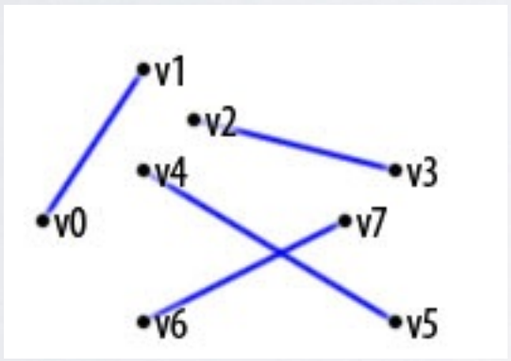
- But... what does it do with it ?

PRIMITIVES

GL_POINTS

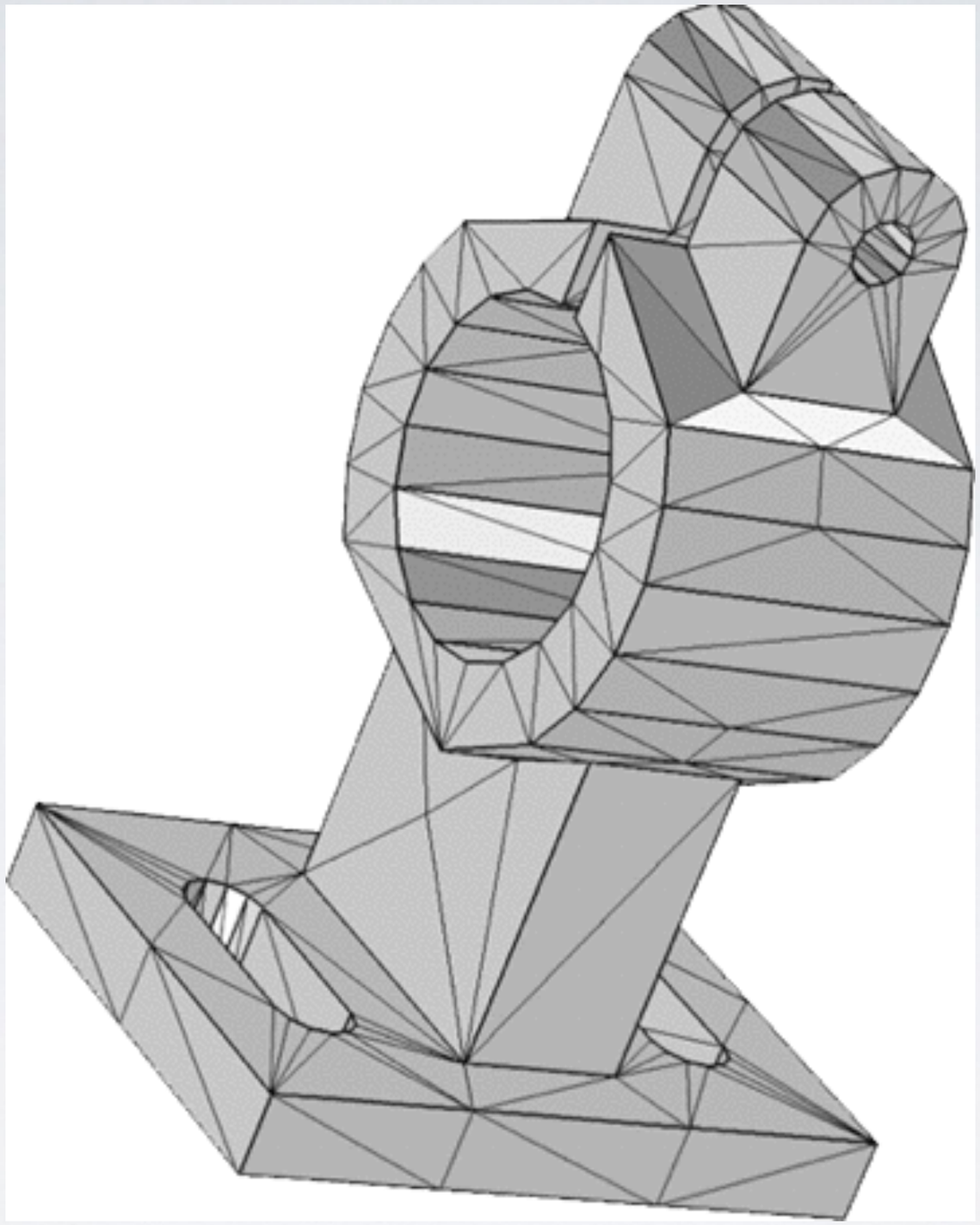


GL_TRIANGLES



GL_LINES

What can we draw with triangles?



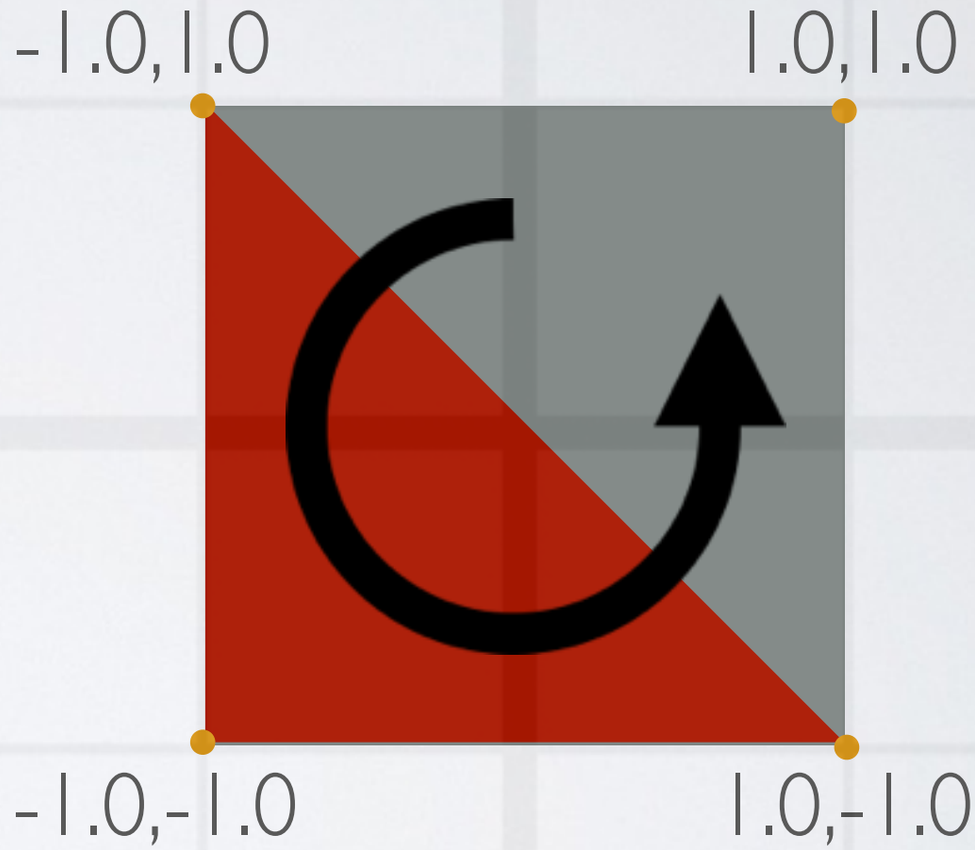
EVERYTHING

NVIDIA Headquarters - Santa Clara - California

HOW TO DRAW

Object Coords

```
// Our vertices
float[] vertices = {
      -1.0f, 1.0f, 0.0f,  // 0, Top Left
      -1.0f, -1.0f, 0.0f, // 1, Bottom Left
      1.0f, -1.0f, 0.0f,  // 2, Bottom Right
      1.0f, 1.0f, 0.0f,   // 3, Top Right
};

// The order we like to connect them.
private short[] indices = {
      0, 1, 2, // Red Triangle
      0, 2, 3 // Gray Triangle
};
```

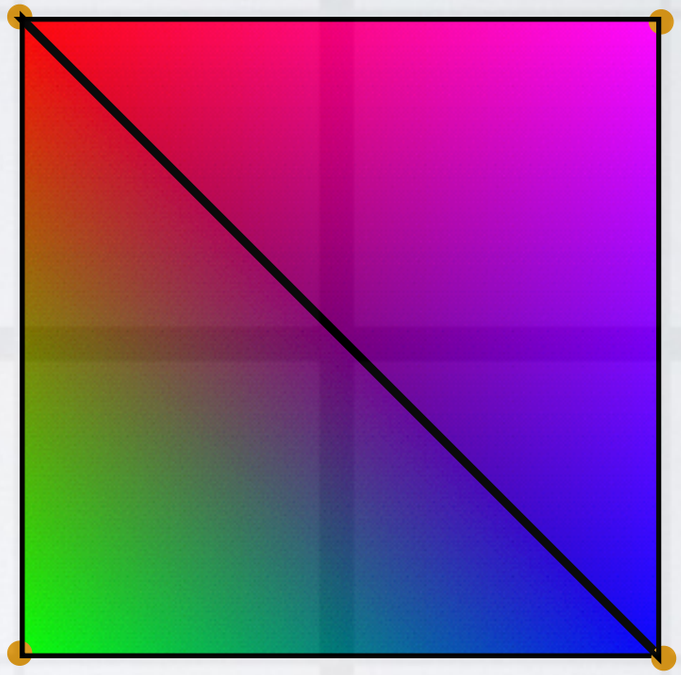


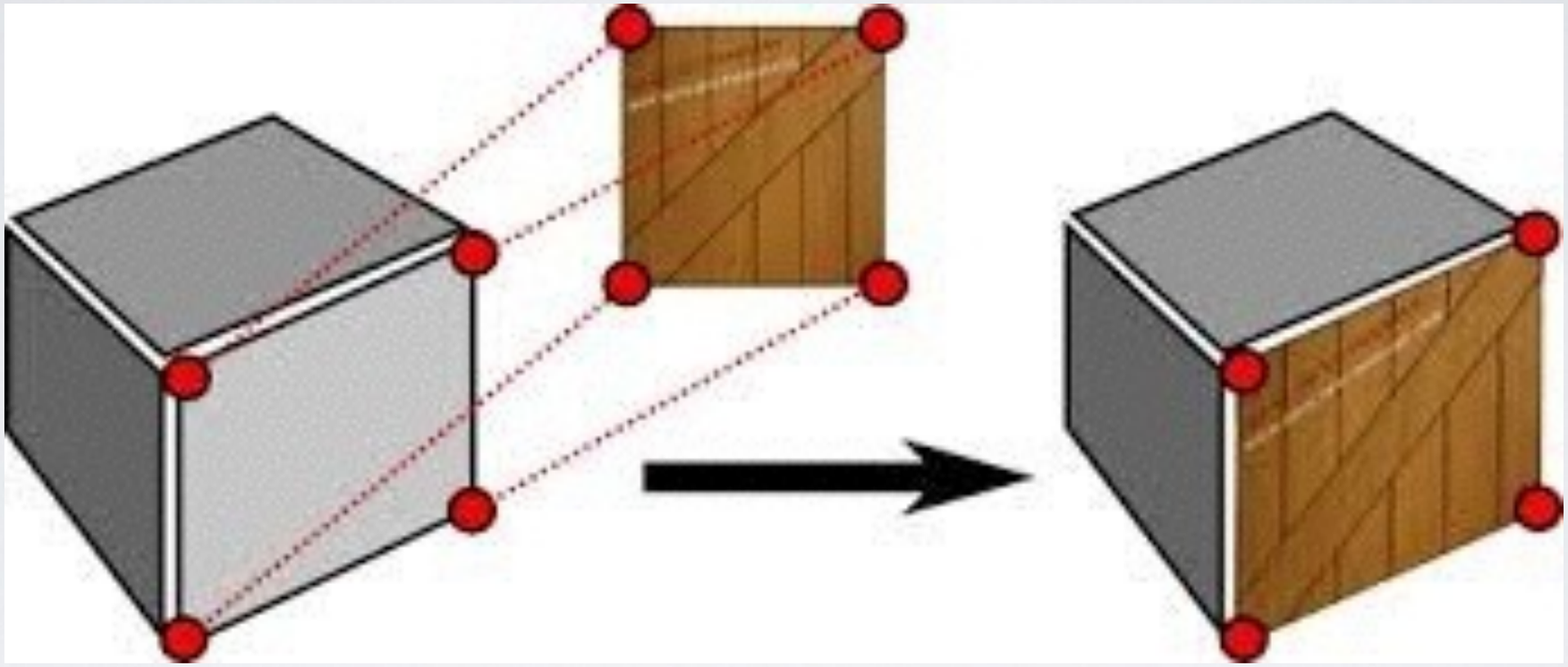
Order matters! Always use counter-clockwise ordering.

COLORING

COLORING

```
// The colors mapped to the vertices.
float[] colors = {
        1f, 0f, 0f, 1f, // point 0 red
        0f, 1f, 0f, 1f, // point 1 green
        0f, 0f, 1f, 1f, // point 2 blue
        1f, 0f, 1f, 1f, // point 3 pink (red + blue)
};
```

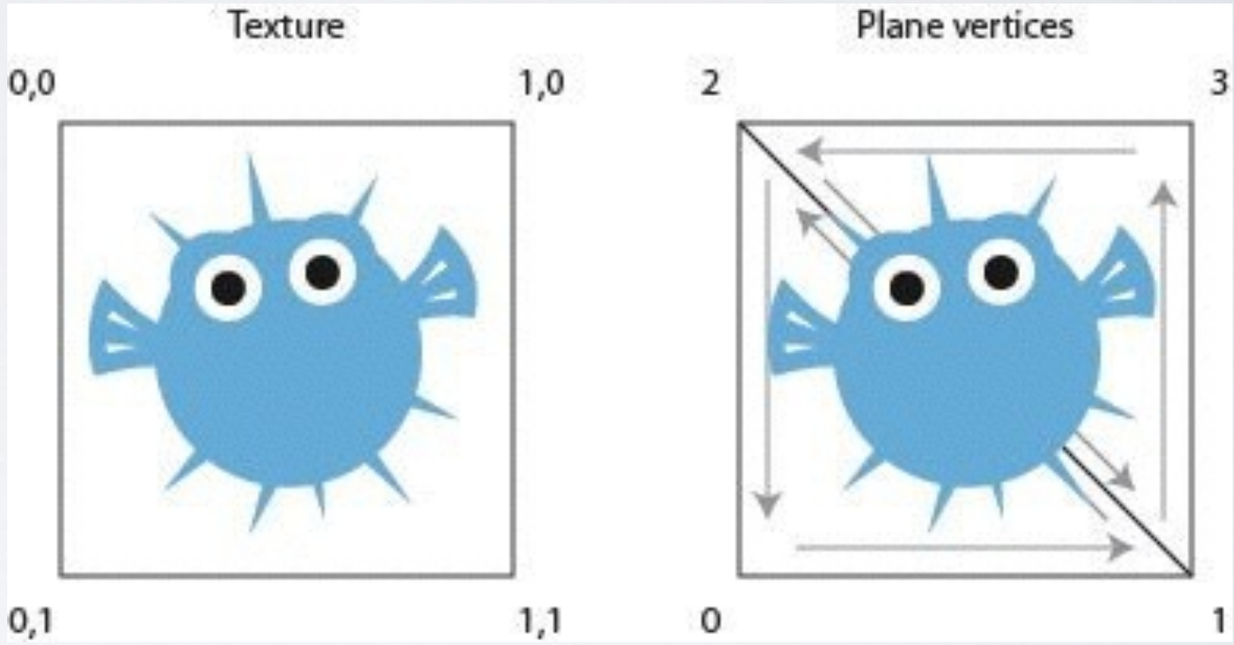




TEXTURES

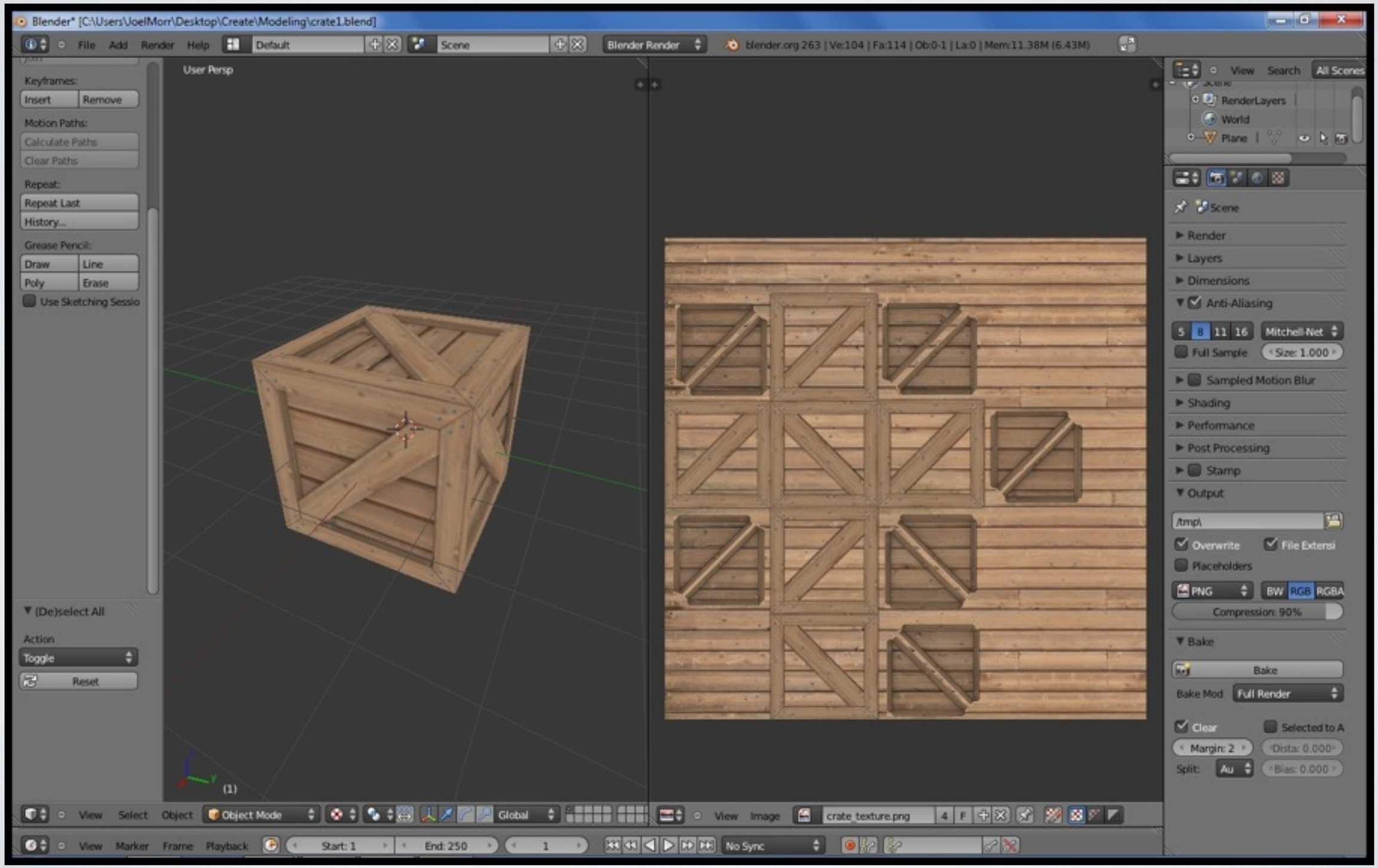
UV MAPPING

```
float[] texCoords = new float[] {
    0.0f, 0.0f,
    0.0f, 1.0f,
    1.0f, 1.0f,
    1.0f, 0.0f
});
```



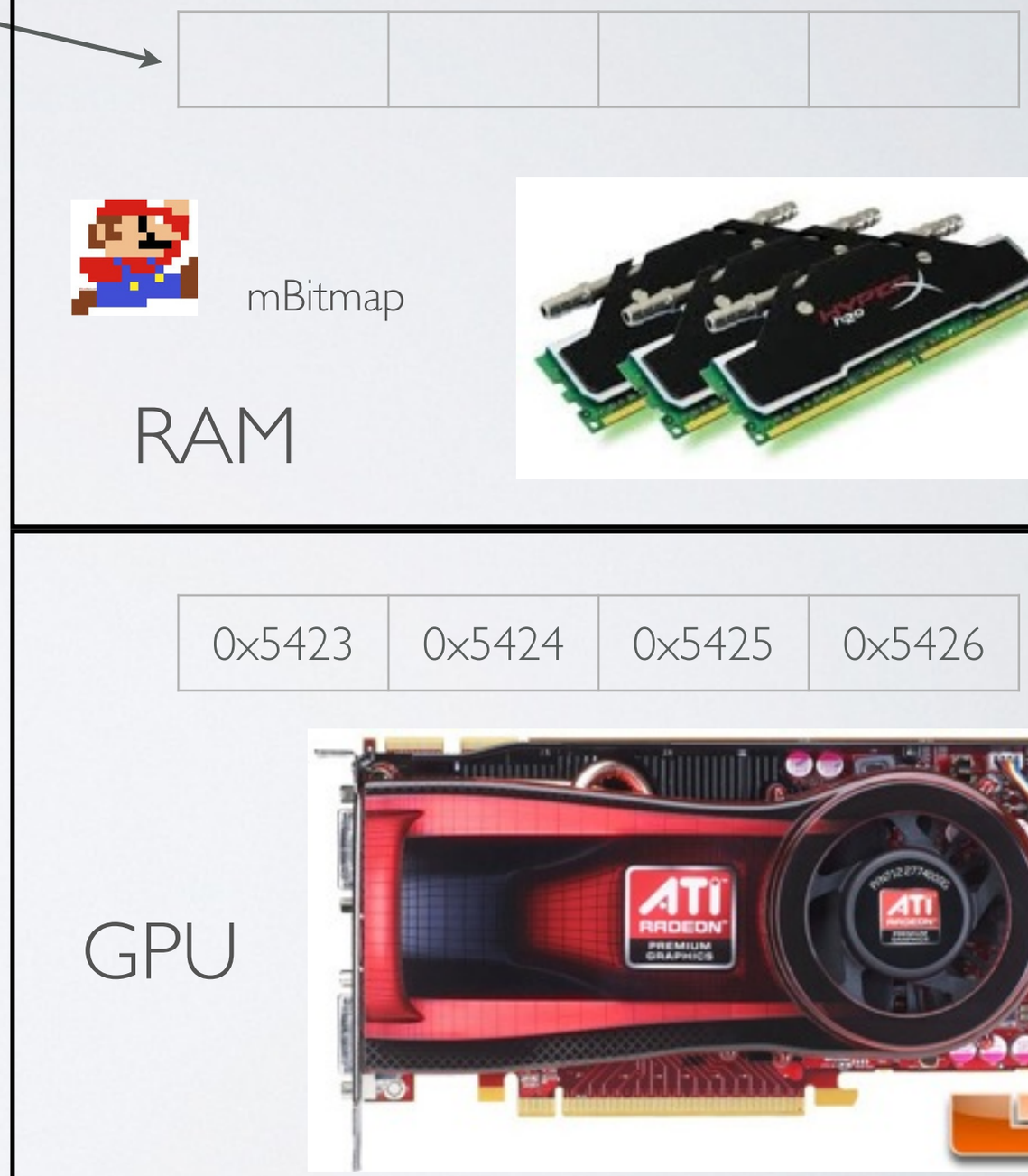
Specify the texture coordinate for each vertex of the geometry you defined

TOOLS TO HELP WITH UV MAPPING



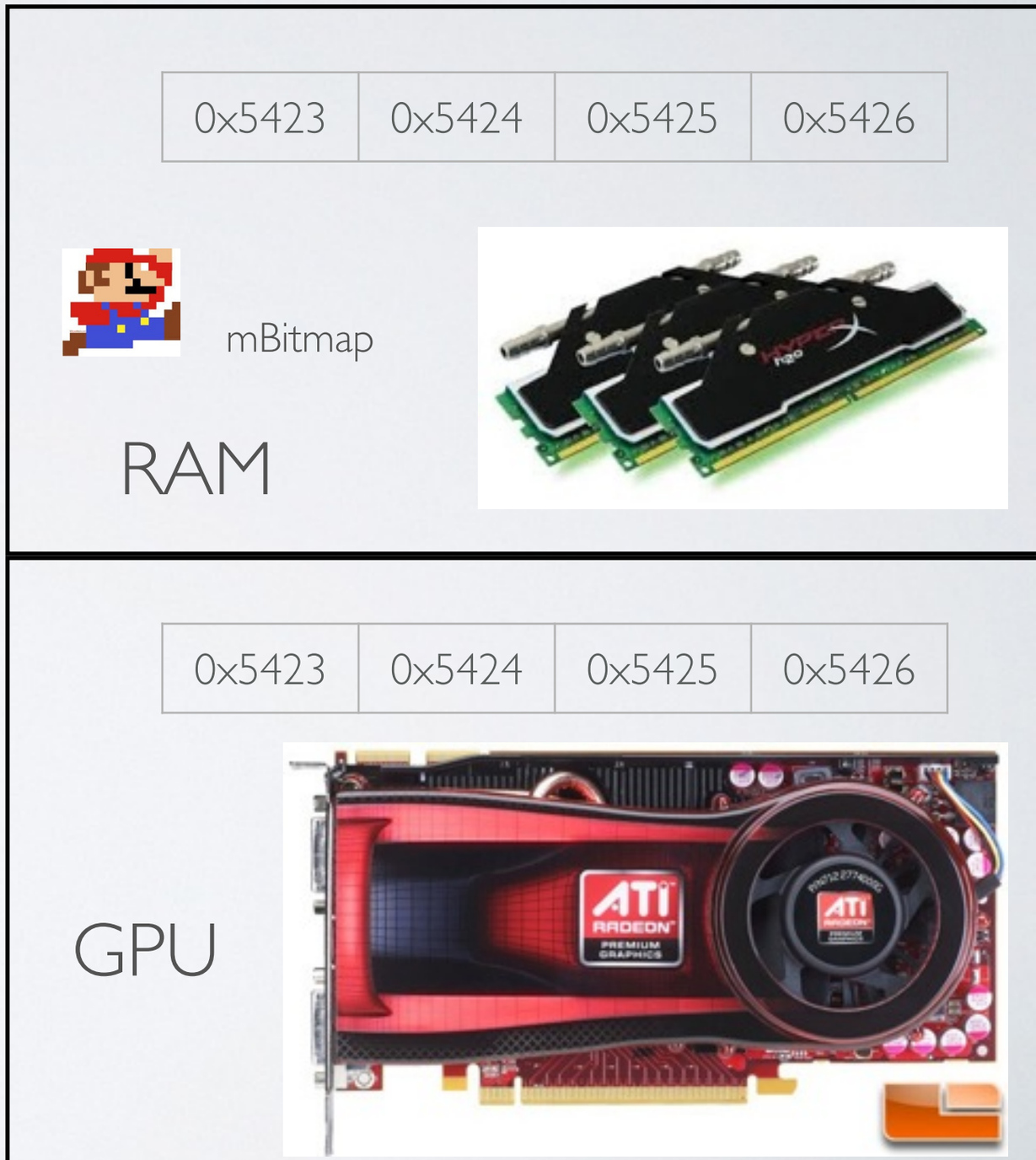
LOADING TEXTURES

```
int[] textures = new int[4];
```



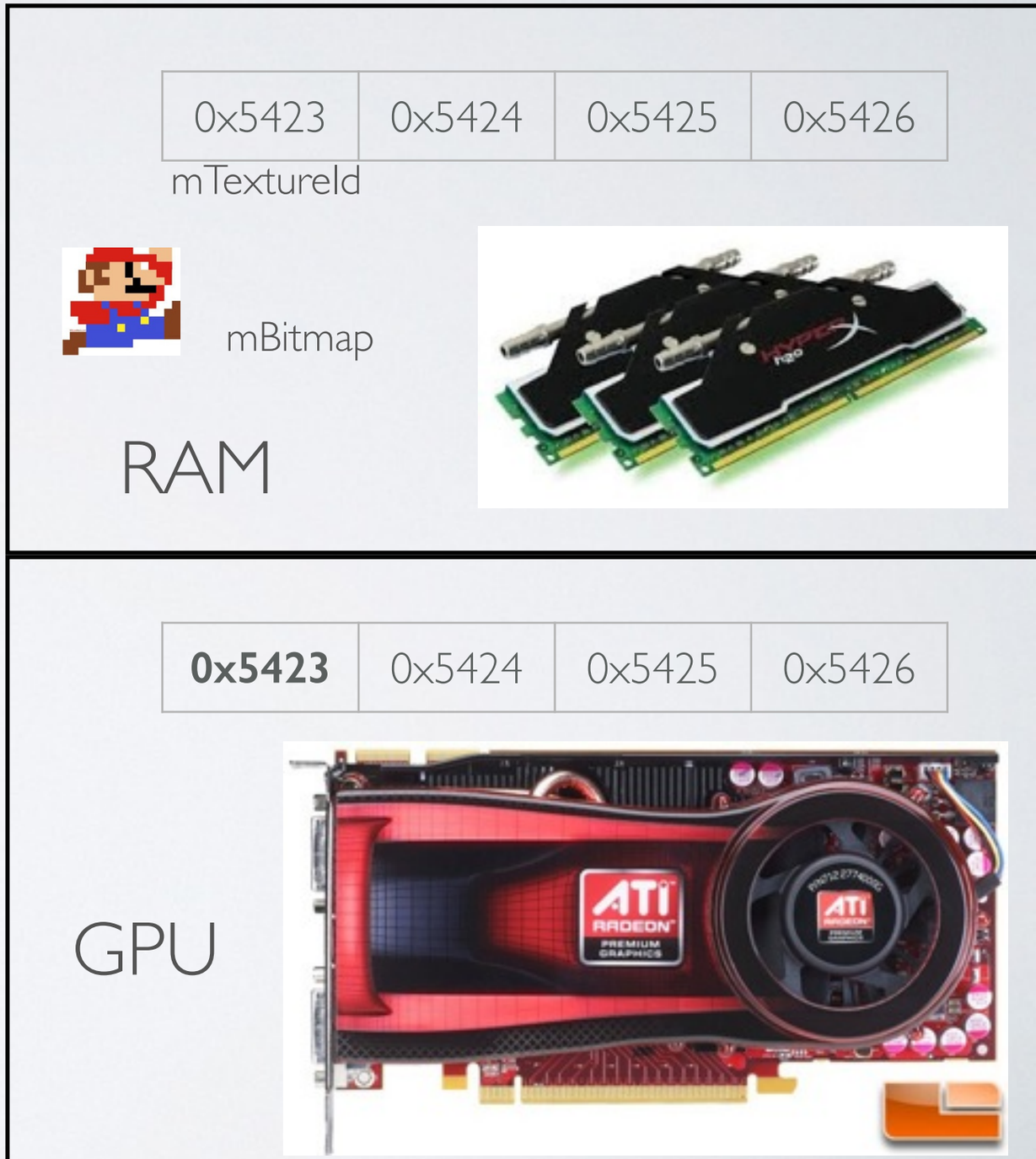
LOADING TEXTURES

```
int[] textures = new int[1];
gl.glGenTextures(4, textures, 0);
```



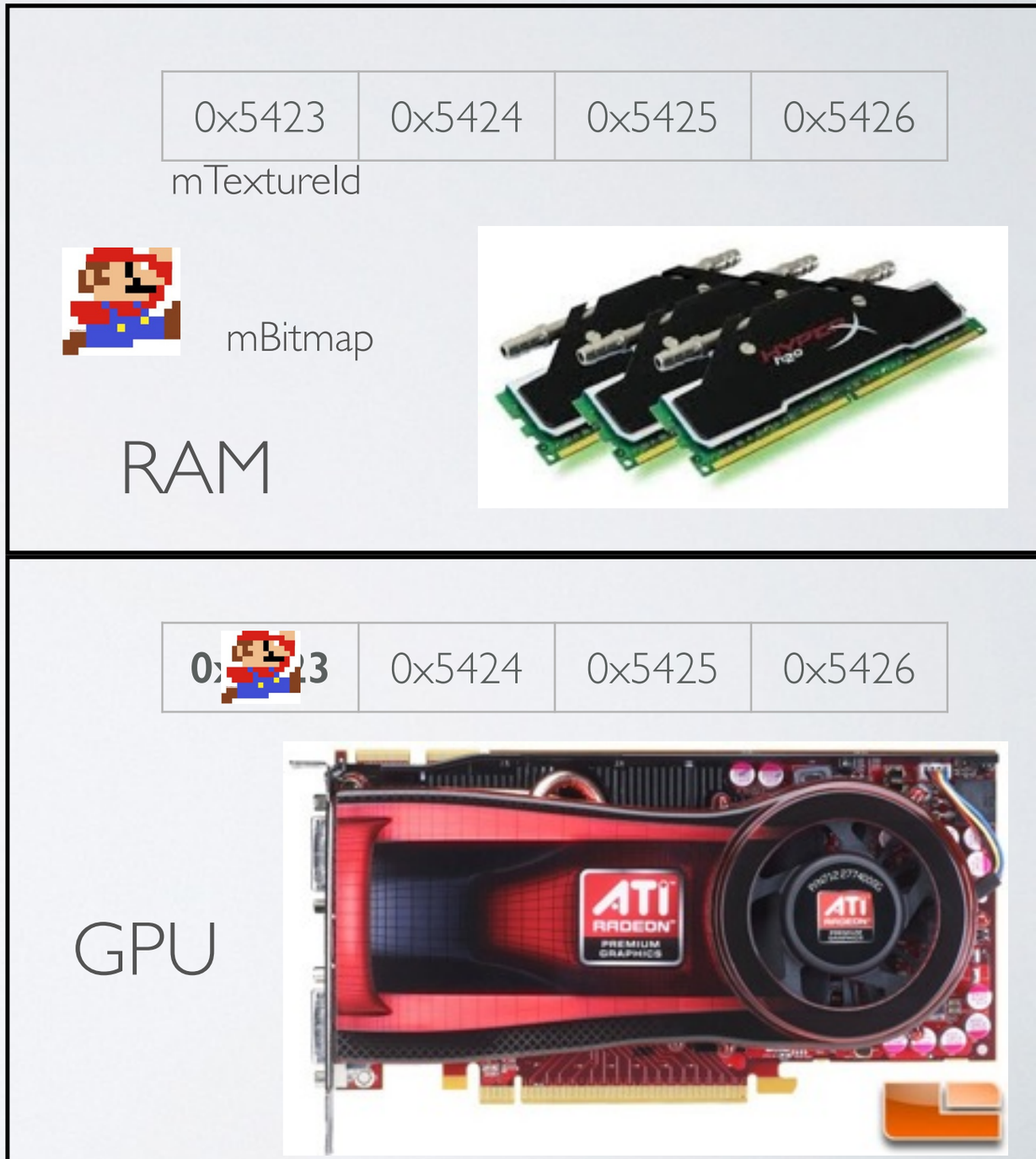
LOADING TEXTURES

```
int[] textures = new int[1];
gl.glGenTextures(4, textures, 0);

int mTextureId = textures[0];
gl.glBindTexture(GL10.GL_TEXTURE_2D,
      mTextureId);

/* Setup Configs */
/* ... */
```



LOADING TEXTURES

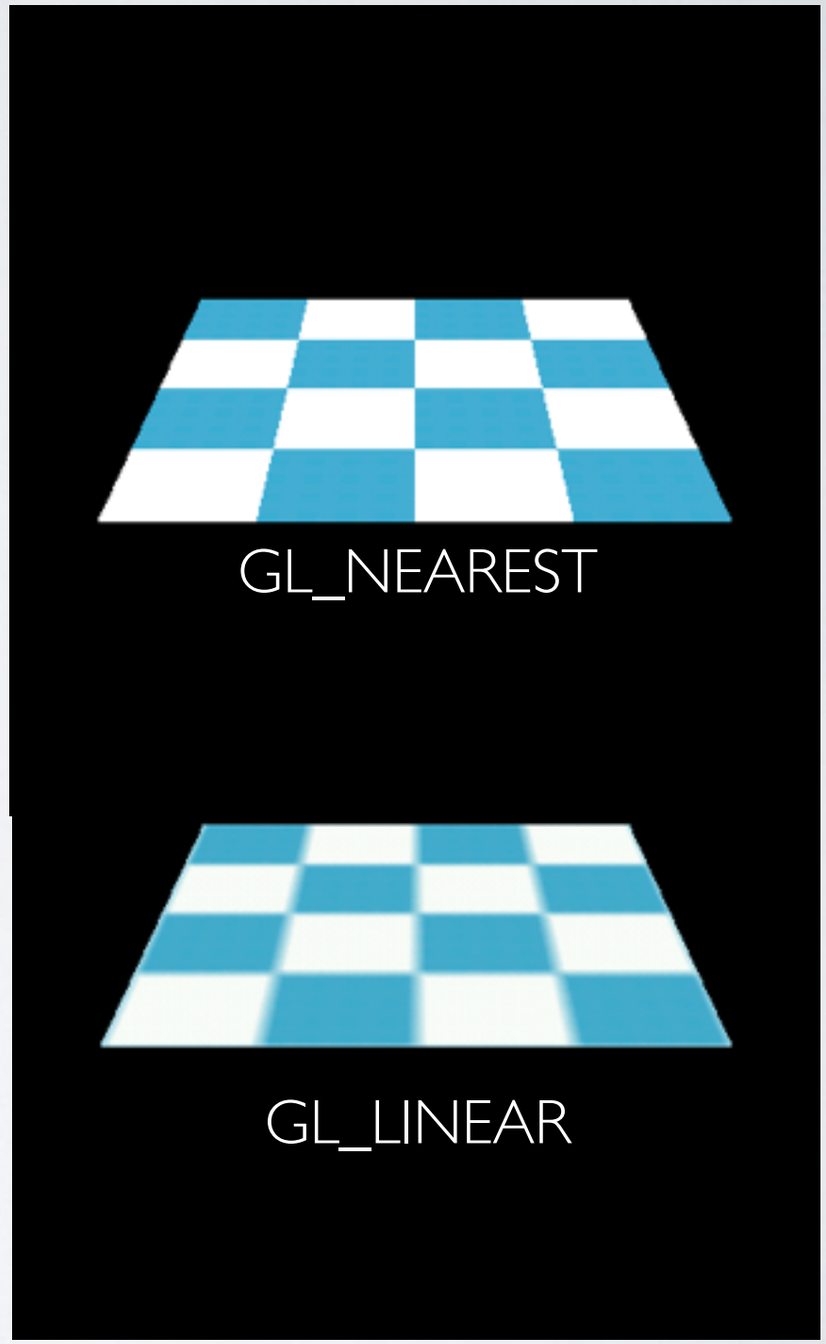
```
int[] textures = new int[1];
gl.glGenTextures(4, textures, 0);

int mTextureId = textures[0];
gl.glBindTexture(GL10.GL_TEXTURE_2D,
      mTextureId);

/* Setup Configs */
/* ... */

GLUtils.texImage2D(
      GL10.GL_TEXTURE_2D, 0,
      mBitmap, 0);
```



TEXTURE PARAMETERS

```
// Scale up if the texture if smaller.
gl.glTexParameterf(GL10.GL_TEXTURE_2D,
                   GL10.GL_TEXTURE_MAG_FILTER,
                   GL10.GL_LINEAR);
```

```
// scale linearly when image smalled than texture
gl.glTexParameterf(GL10.GL_TEXTURE_2D,
                   GL10.GL_TEXTURE_MIN_FILTER,
                   GL10.GL_NEAREST);
```



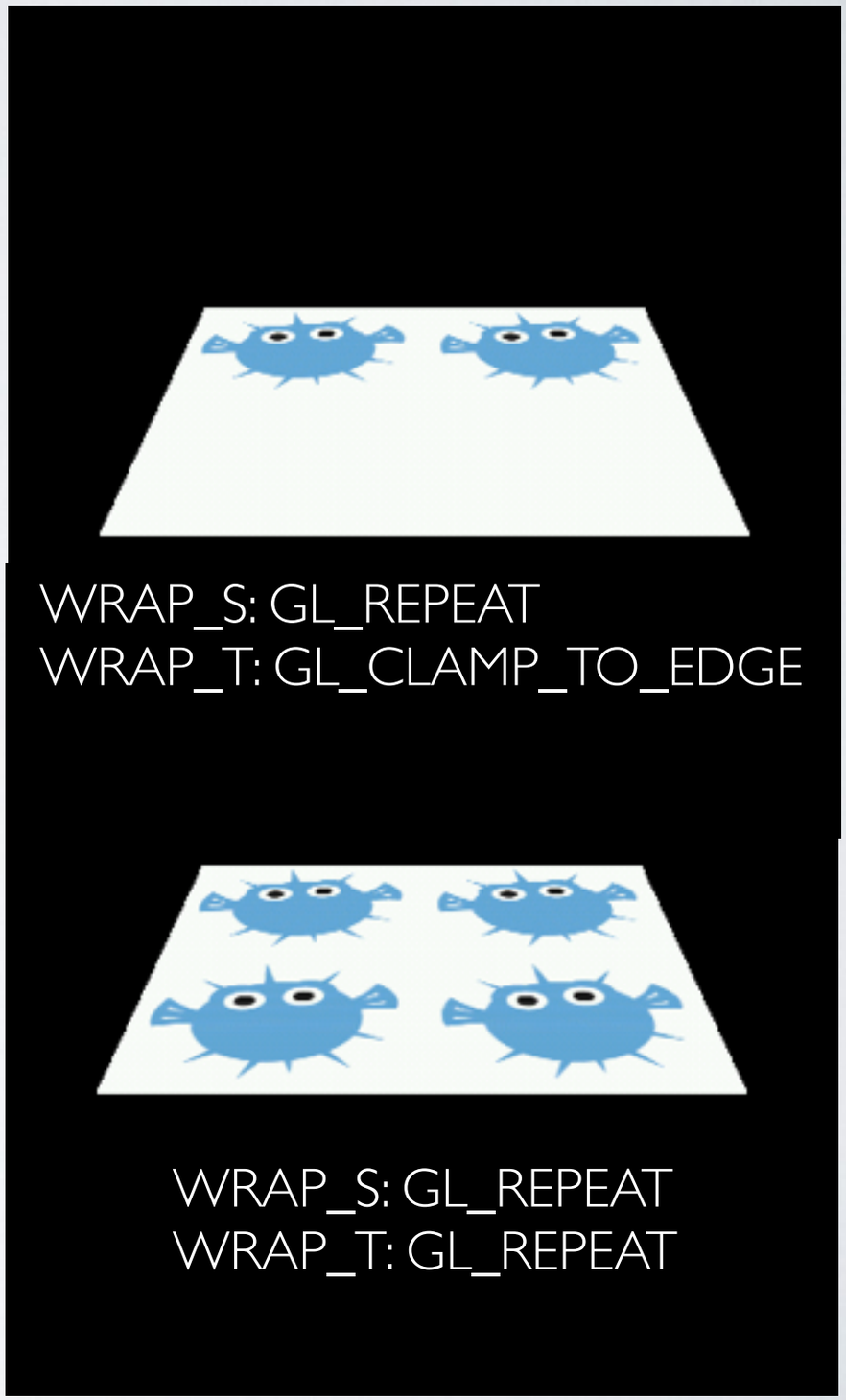
These parameters specify what algorithm OpenGL should use when the texture gets bigger or smaller

TEXTURE PARAMETERS

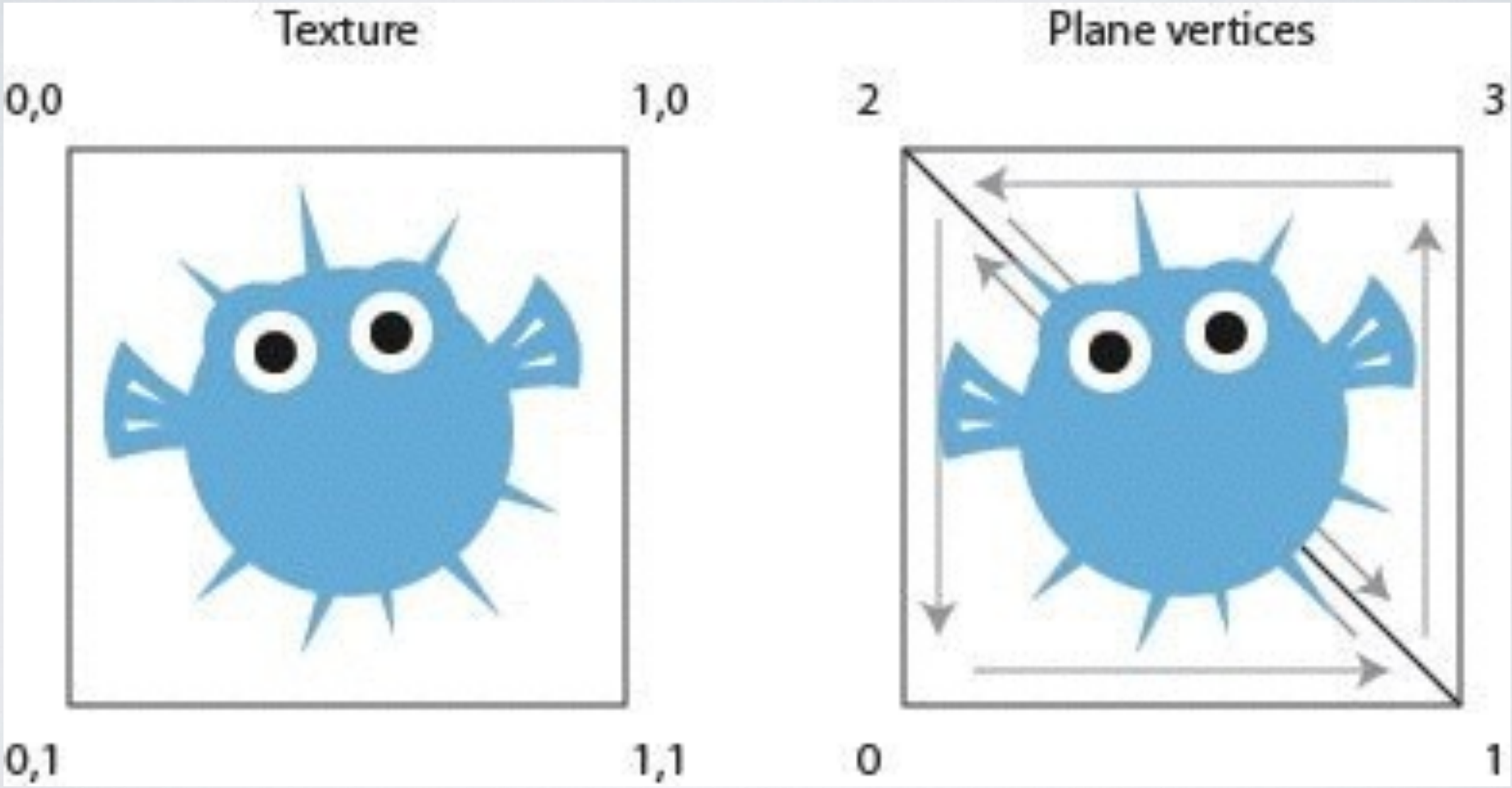
```
gl.glTexParameterf(GL10.GL_TEXTURE_2D,
                   GL10.GL_TEXTURE_WRAP_S,
                   GL10.GL_REPEAT);
```

```
gl.glTexParameterf(GL10.GL_TEXTURE_2D,
                   GL10.GL_TEXTURE_WRAP_T,
                   GL10.GL_CLAMP_TO_EDGE);
```

These parameters specify what OpenGL should do if the user specifies a texture coord outside image original dimensions



TEXTURE PARAMETERS



ANDROID FRAGMENTATION

let the headache begin

ANDROID FRAGMENTATION

The developer of "Angry Birds," a top-selling iPhone game, reported that **bringing the title to Android devices ended up more difficult than anticipated due to fragmentation within the open platform.**

http://appleinsider.com/articles/10/11/19/angry_birds_maker_apologizes_for_android_fragmentation_issues

I HAVE FOUND SO FAR...

- **Galaxy S4:** Only support textures that have sizes with power of 2.

- **Motorola XOOM:** GL_REPEAT only works until some point, then it uses GL_CLAMP_TO_EDGE

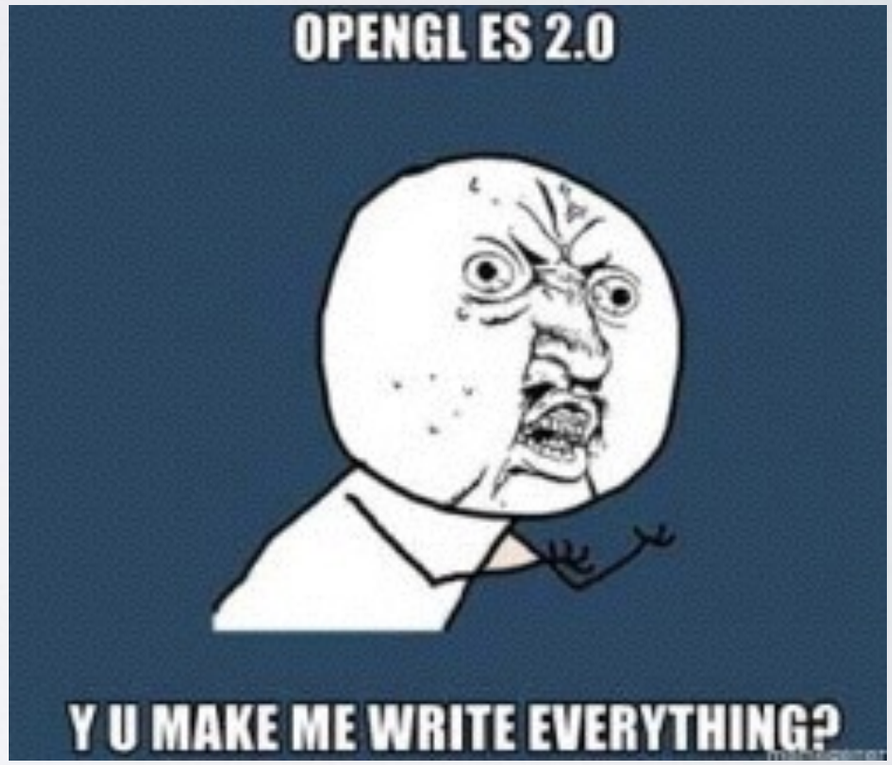
- **Nexus 4:** Had some problems with additive blending (worked different than other phones)

more?

OPENGL ES 2.0

DIFFERENCES WITH VERSION 2.0+

- GLSL (openGL Shading Language)
- Program (Shader+Fragment)
- Programmable Pipeline Concept



GLSL VERTEX CODE

```
uniform mat4 u_MVPMatrix;  // A constant representing the combined
                           // model/view/projection matrix.
attribute vec4 a_Position;     // Per-vertex position information we will pass in.
attribute vec4 a_Color;        // Per-vertex color information we will pass in.

varying vec4 v_Color;  // This will be passed into the fragment shader.


void main()                    // The entry point for our vertex shader.
{
    v_Color = a_Color;  // Pass the color through to the fragment shader.
                        // It will be interpolated across the triangle.
    gl_Position = u_MVPMatrix  // gl_Position is a variable used to store the final position.
          * a_Position;      // Multiply the vertex by the matrix to get the final point in
};                          // normalized screen coordinates.
```

SHADER EFFECTS

Bloom

SHADER EFFECTS

Reflection

Do you?

QUESTIONS

REFERENCES

Great Tutorial for Beginners:

http://www.jayway.com/2009/12/03/opengl-es-tutorial-for-android-part-i/

Android Dev:

http://developer.android.com/

OpenGL ES Documentation:

http://www.khronos.org/opengles/sdk/1.1/docs/man/

Stanford Presentation:

https://www.youtube.com/watch?v=_WcMe4Yj0NM