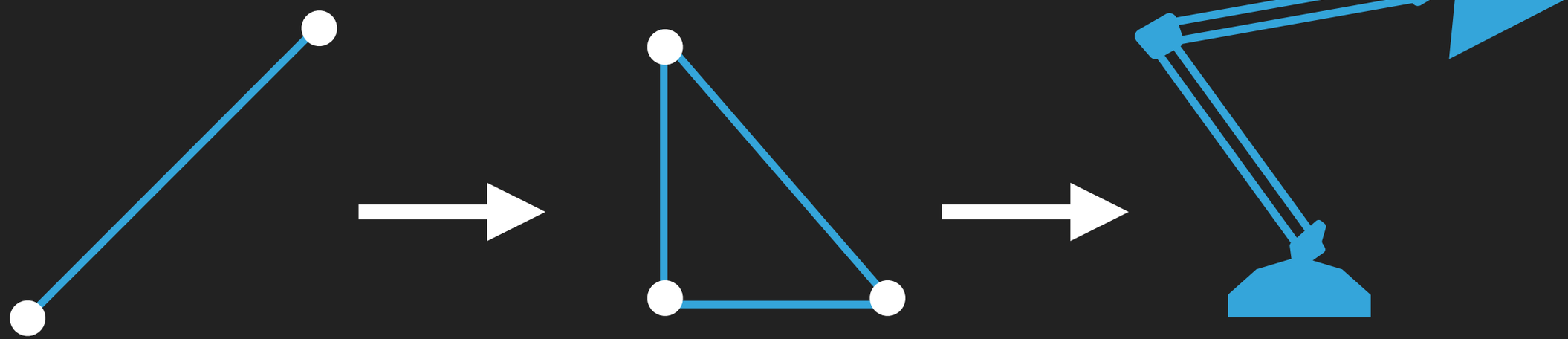
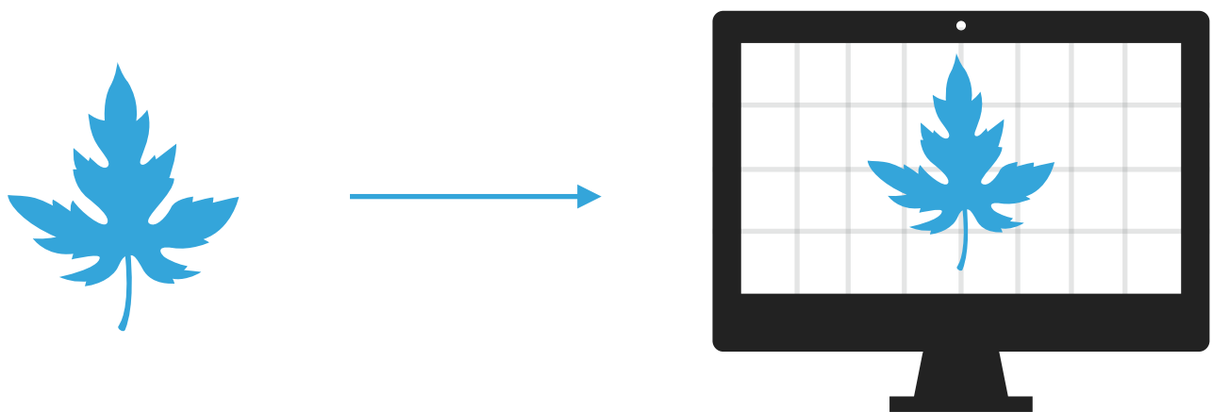


INTRODUCTION TO COMPUTER GRAPHICS AND WEBGL

AUDIO VISUAL PROGRAMMING

GRAPHICS DEFINITION

A graphic is an image or visual representation of an object



Computer graphics are “simply” images displayed on a computer screen.

SOME EXAMPLES

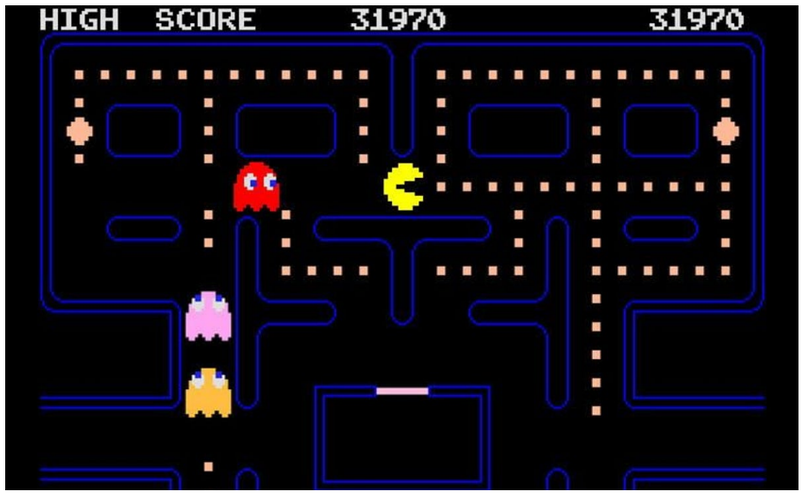
Photograph

Logos

3D models

GRAPHICS DIMENSIONS

2D Graphics



3D Graphics

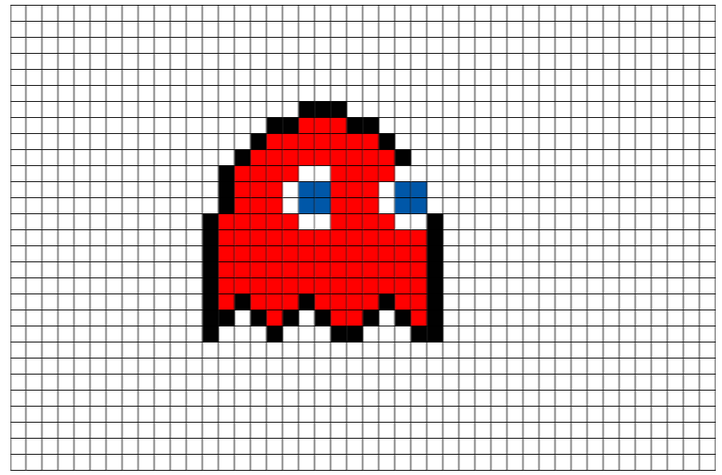
Wait, isn't the computer screen just a flat 2D surface?

How does it reproduce 3D visual representations?

2D GRAPHICS

Raster Graphics (JPEG,PNG, PSD)

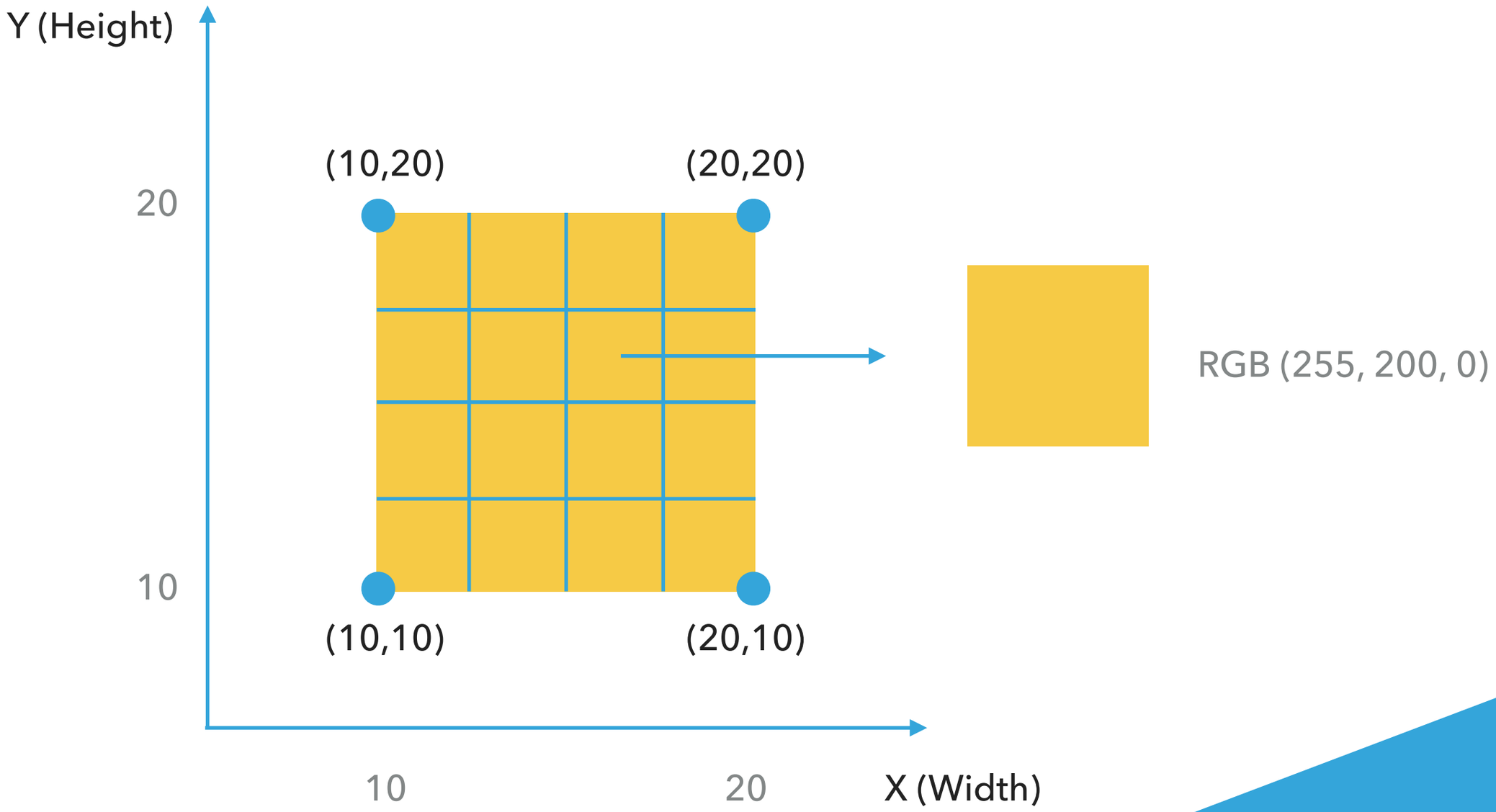
- Most common
- Used for digital photos, Web graphics, and icons
- Composed of a simple grid of pixels



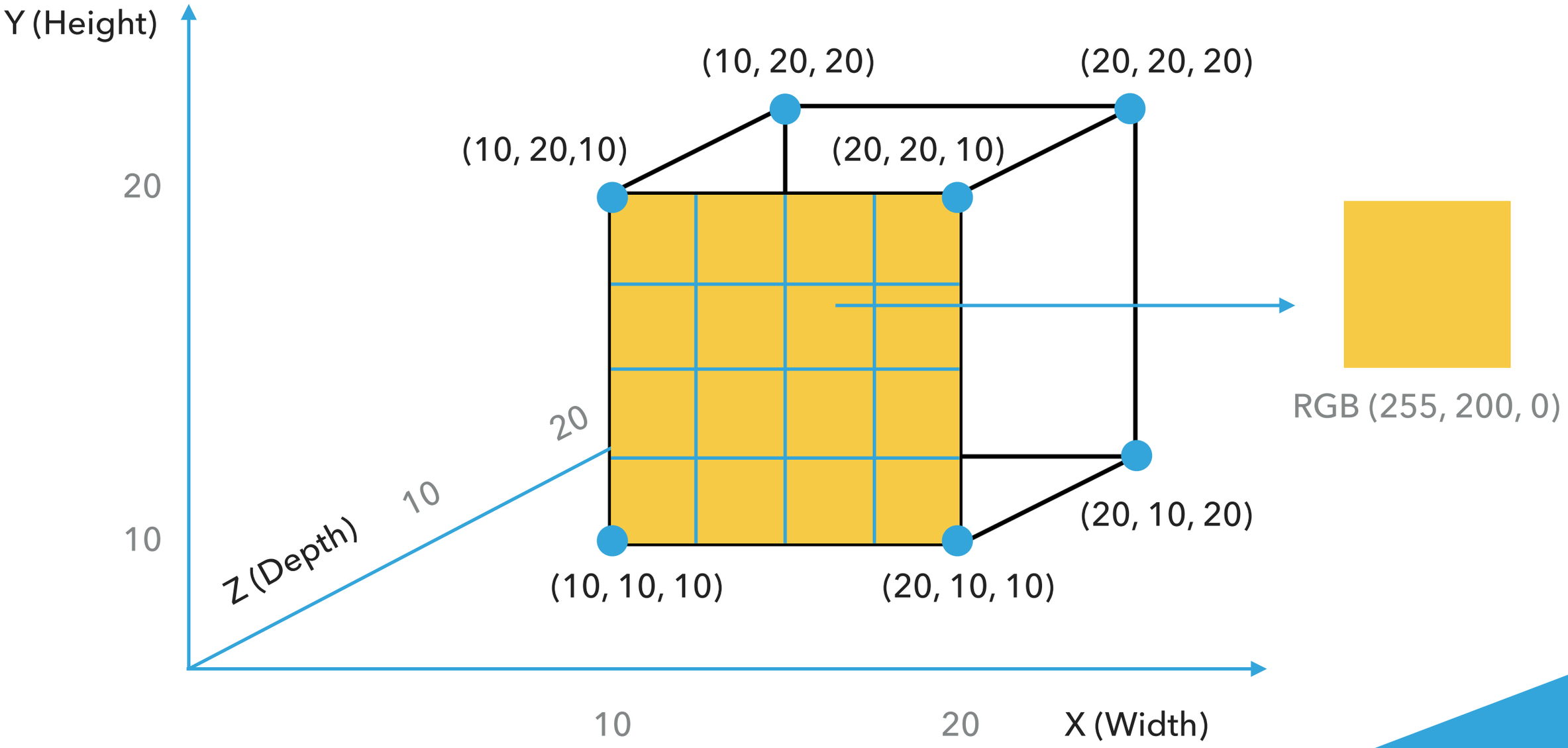
Vector Graphics (SVG, PDF, AI)

- Made of paths and mathematical formulas
- Used for logos, signs, and banners
- Can be scaled without losing quality

2D GRAPHICS AXIS



3D GRAPHICS

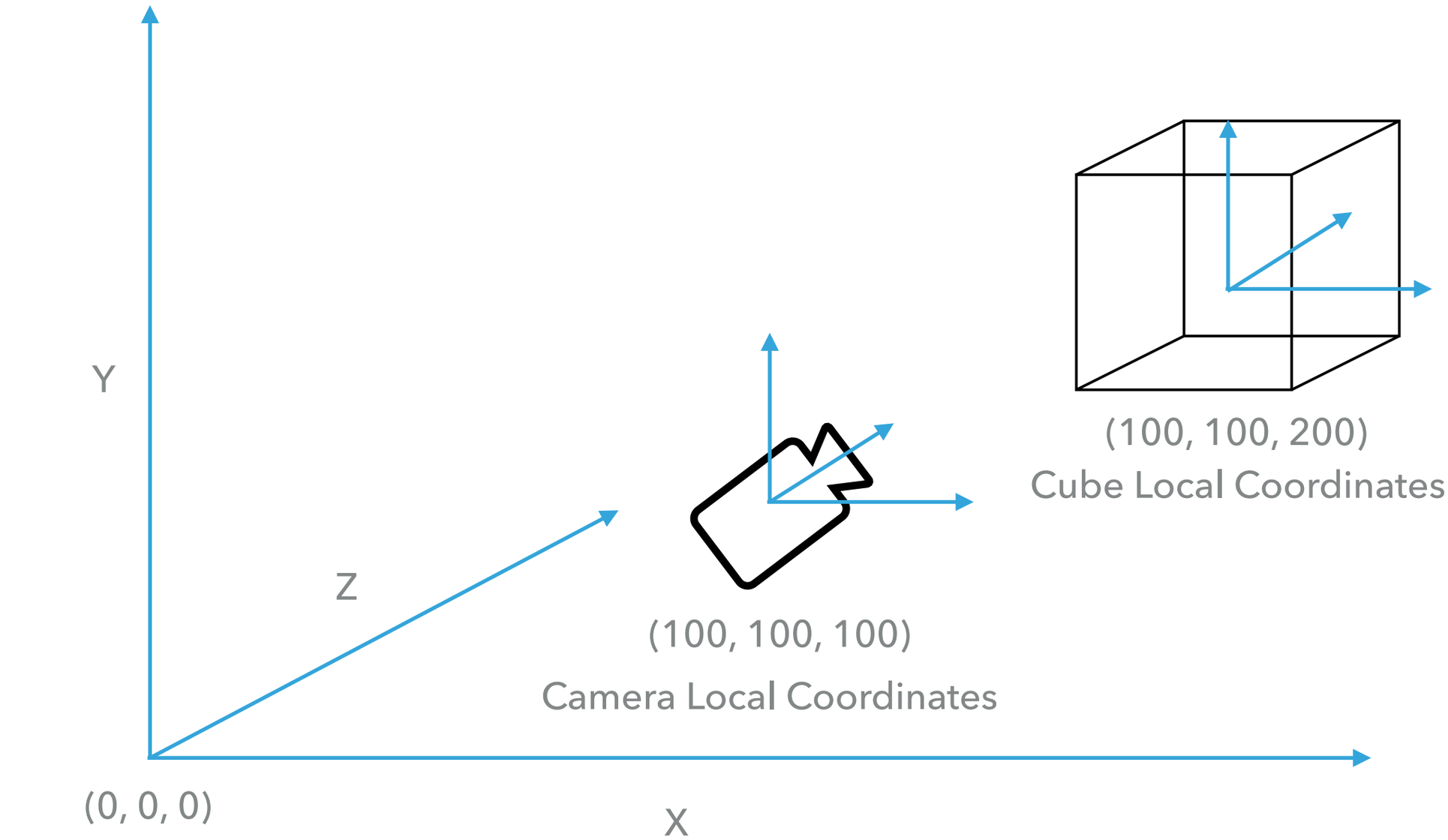


The main challenge of 3D computer graphics is to represent 3D objects into a 2D screen.

3D GRAPHICS TO 2D SCREEN

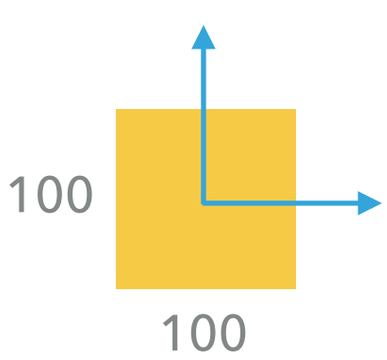
- The easiest way to think about the conversion from 3D world to 2D screen is with a camera.
- World coordinates VS Local/Object coordinates
- Transformations
- Decide what to show and what to hide
- Finally, since the end product will be a 2D representation of the 3D space, we can add some lighting to improve the 3D experience

3D GRAPHICS TO 2D SCREEN

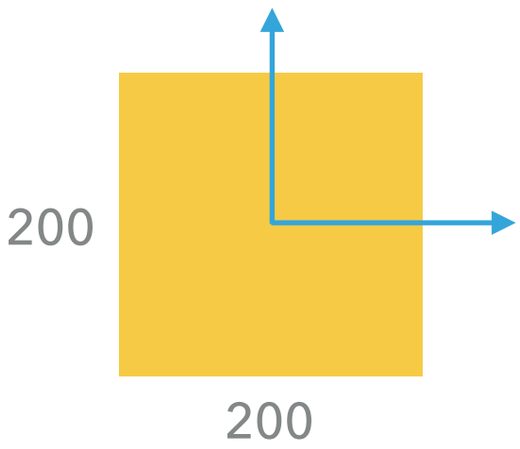


3D GRAPHICS TO 2D SCREEN

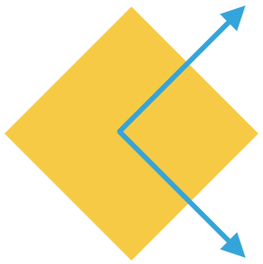
Object base



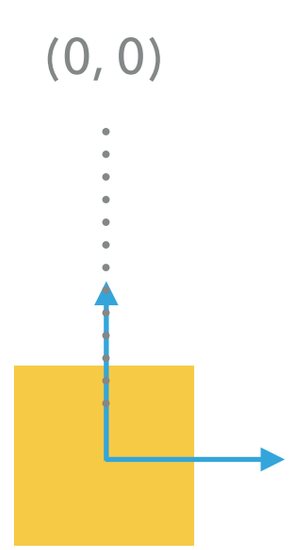
Scale



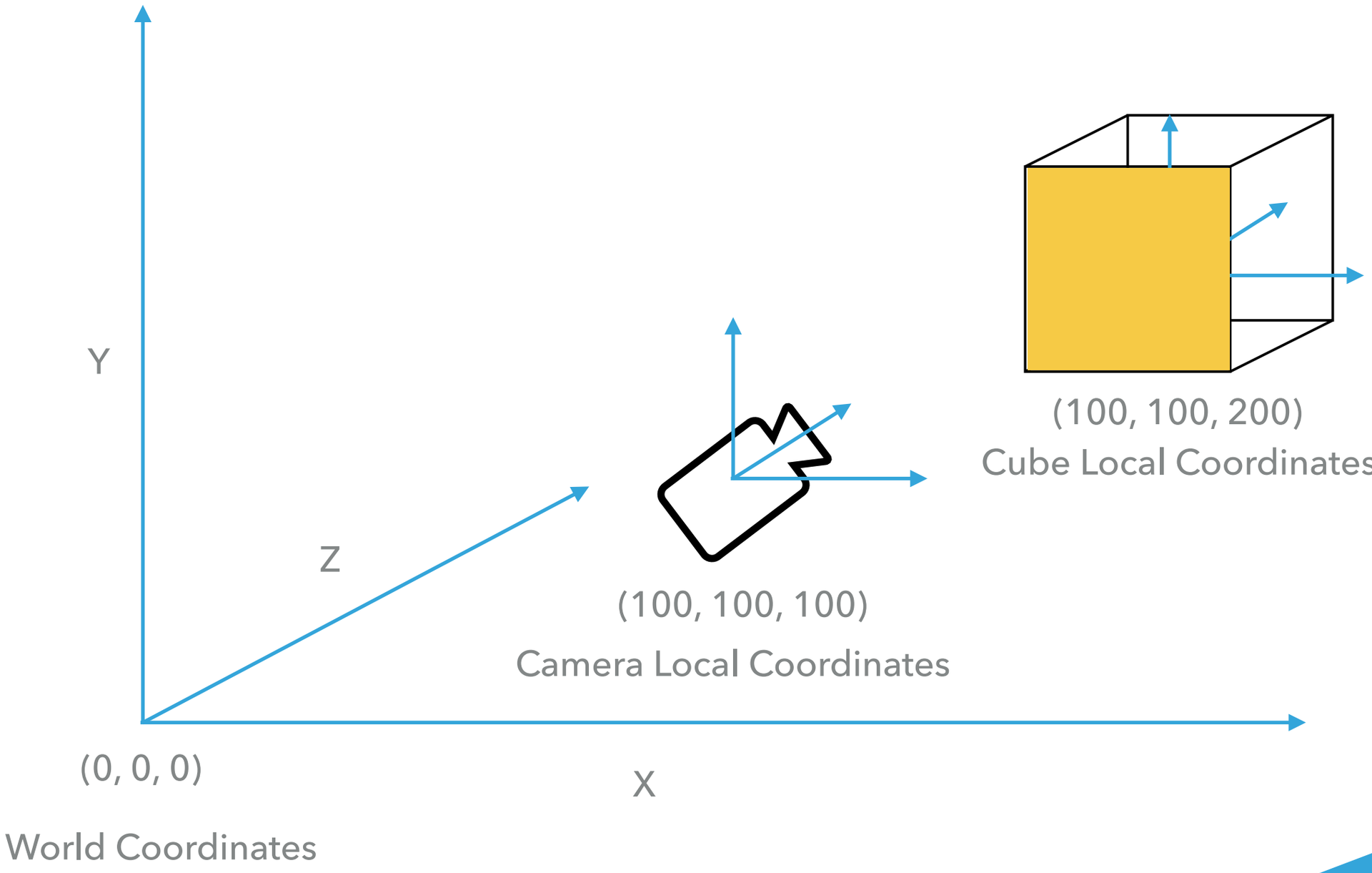
Rotation



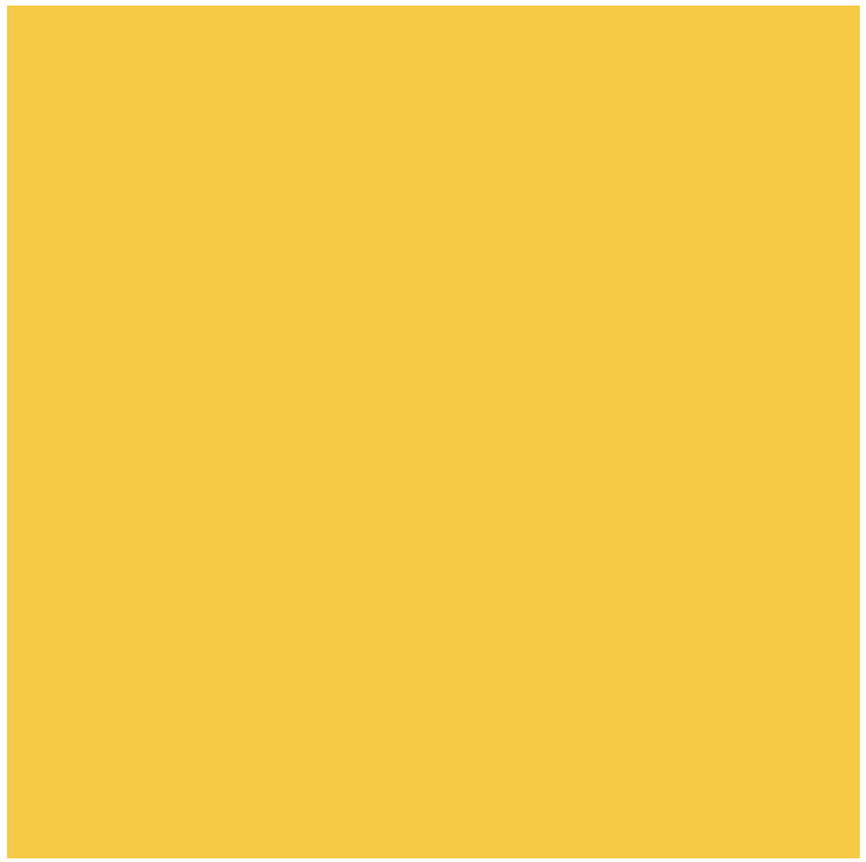
Translate



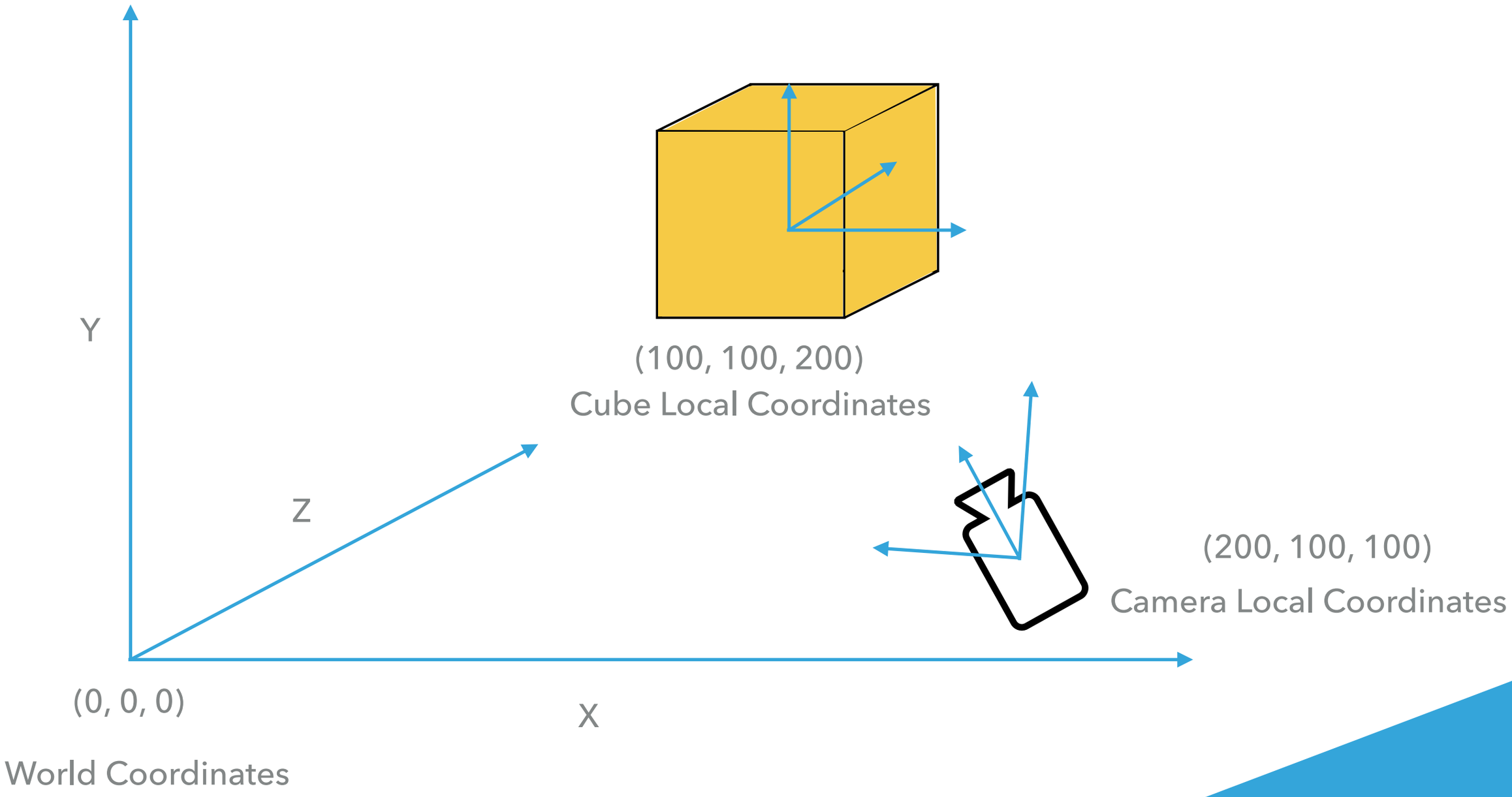
3D GRAPHICS TO 2D SCREEN



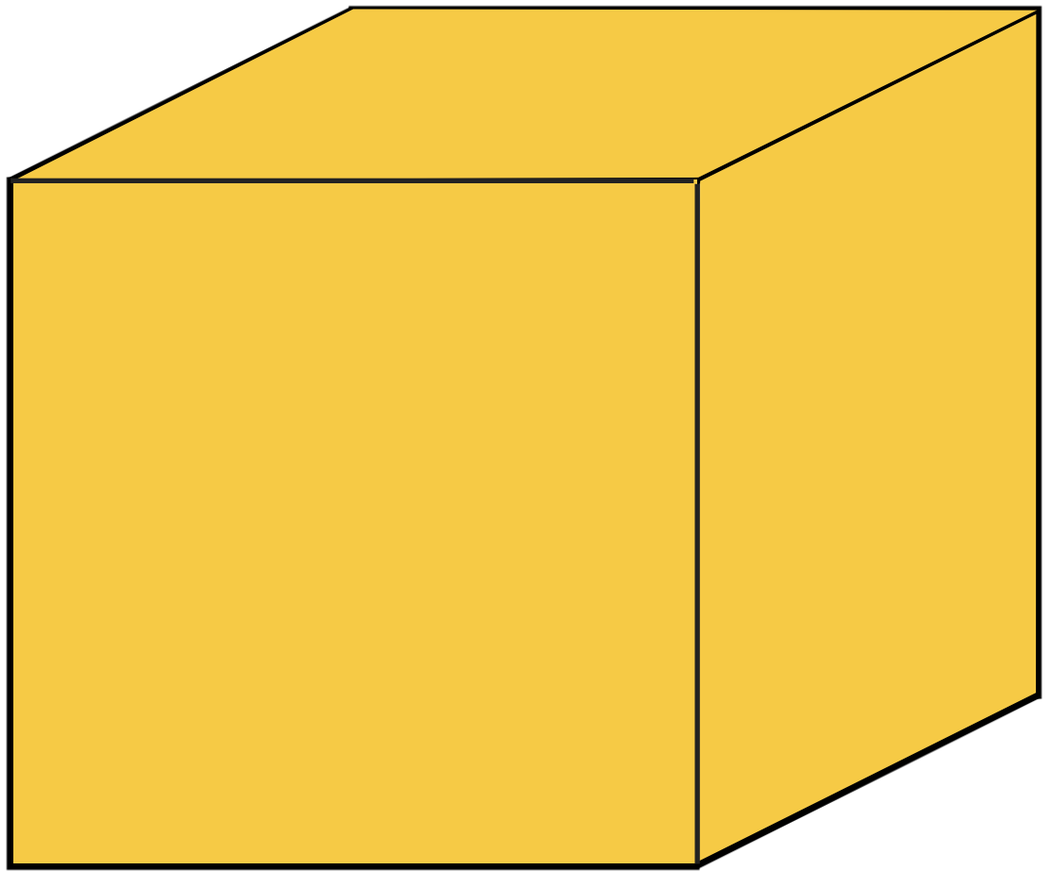
3D GRAPHICS TO 2D SCREEN



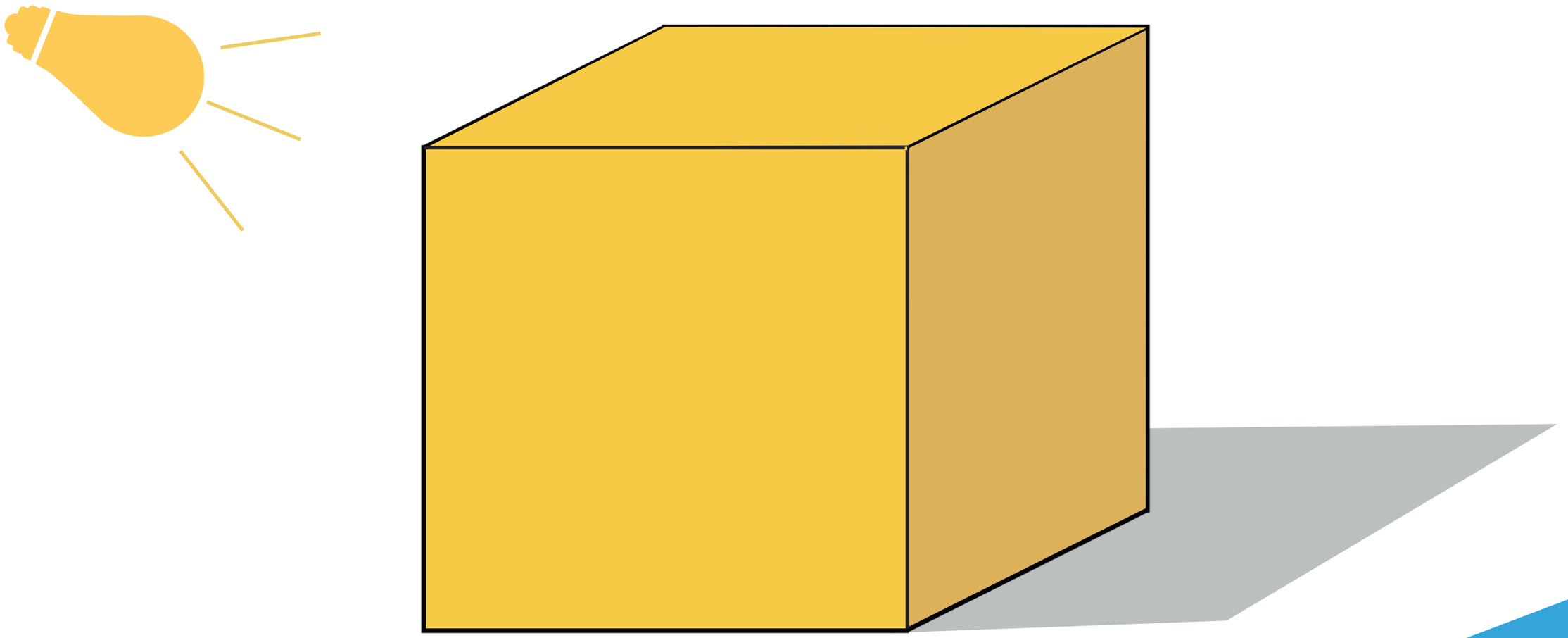
3D GRAPHICS TO 2D SCREEN



3D GRAPHICS TO 2D SCREEN



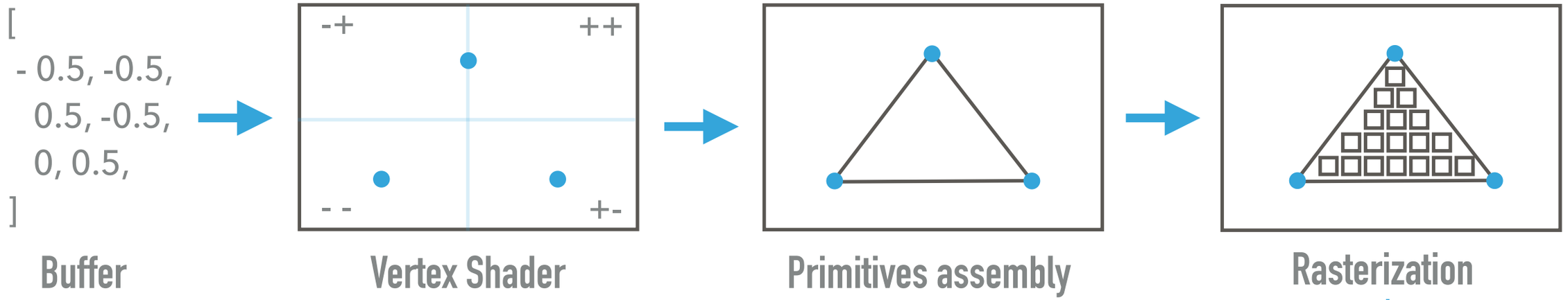
3D GRAPHICS TO 2D SCREEN



WEBGL

- Stands for Web Graphics Library
- JavaScript API that allows compatible web browsers to render 2D and 3D graphics
- Based on OpenGL
- Written in Javascript and executed by the GPU
- Uses the HTML5 Canvas element to render graphics

WEBGL PIPELINE



VERTEX SHADER

```
attribute vec2 position;

void main()
{
  gl_Position = vec4(position, 0.0, 1.0);
}
```

FRAGMENT SHADER

```
precision highp float;
uniform vec4 color;

void main()
{
  gl_FragColor = color;
}
```