ThreeJS and WebGL

GPUs or bust

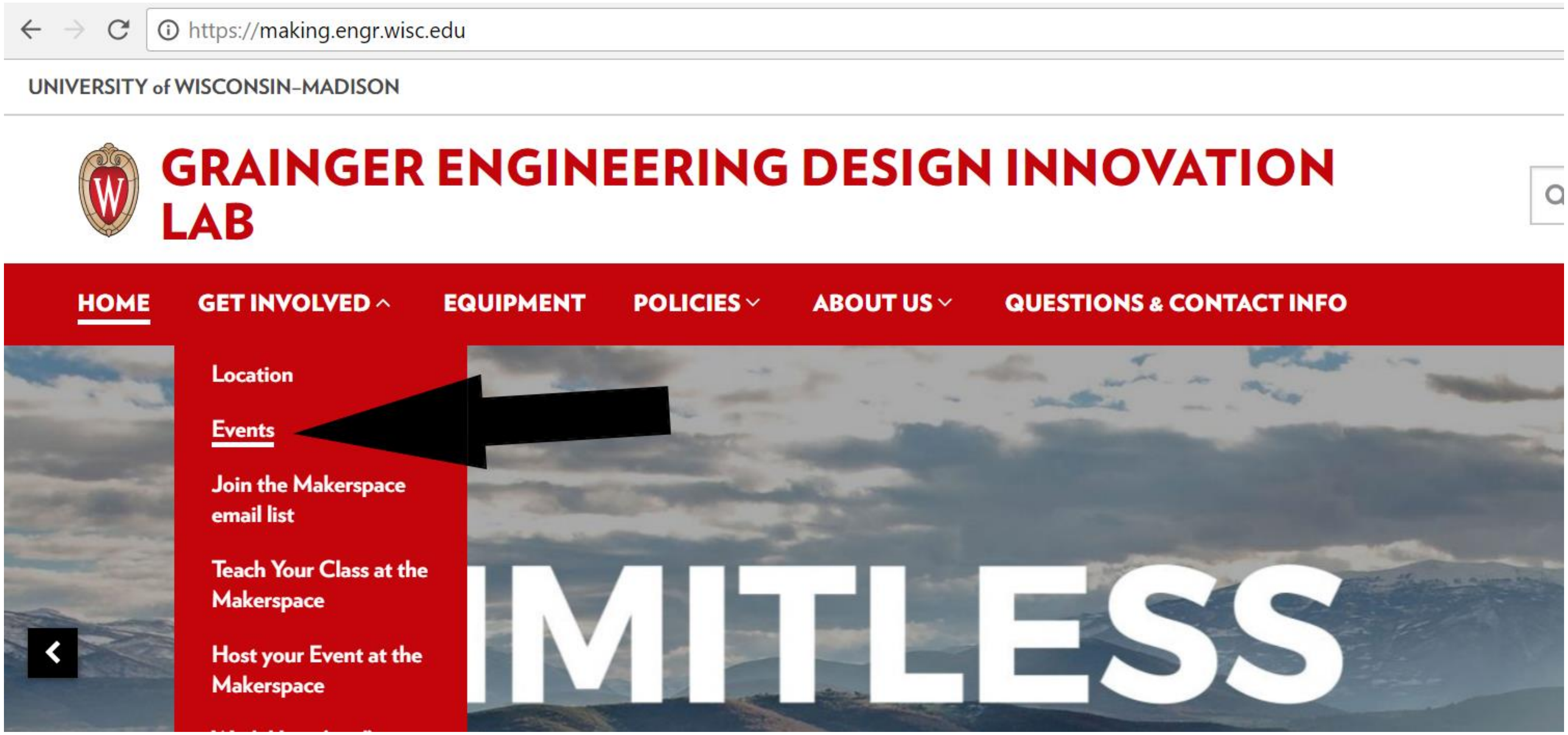
Hosted by:

IEEE

UW Madison

More Workshops This Year!

https://making.engr.wisc.edu



https://making.engr.wisc.edu

UNIVERSITY of WISCONSIN–MADISON

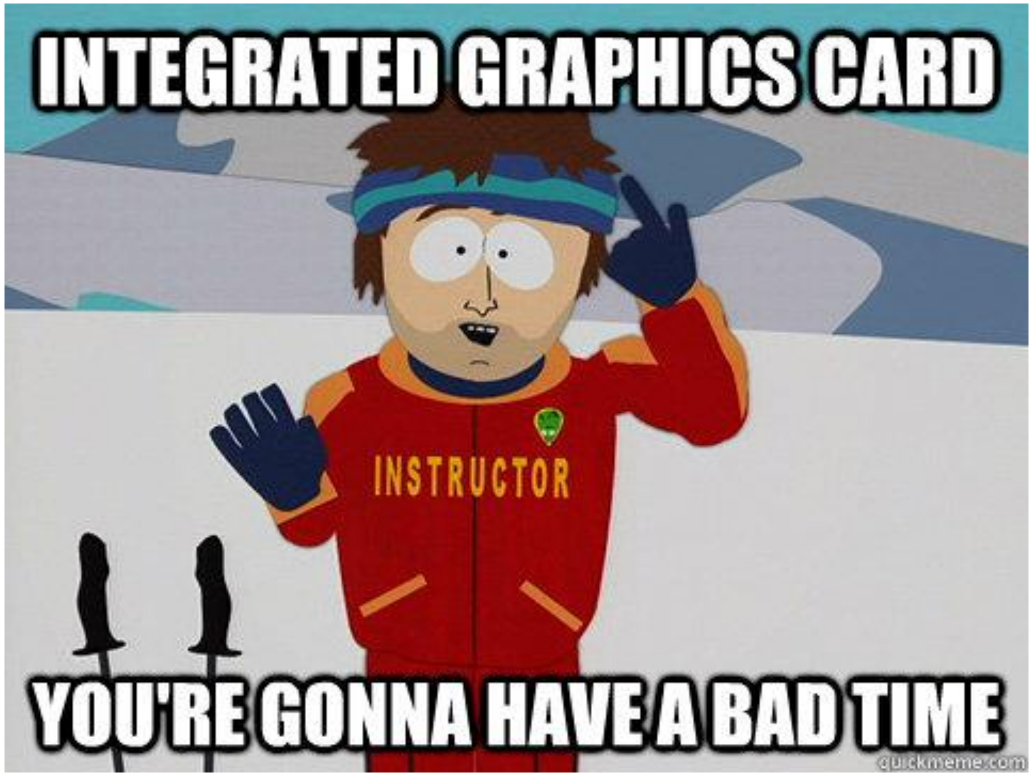
GRAINGER ENGINEERING DESIGN INNOVATION LAB

HOME | GET INVOLVED | EQUIPMENT | POLICIES | ABOUT US | QUESTIONS & CONTACT INFO

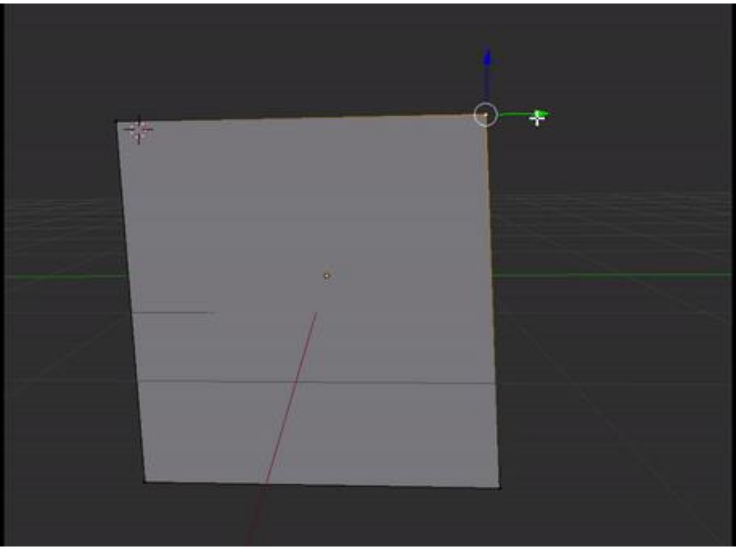
- Location
- Events
- Join the Makerspace email list
- Teach Your Class at the Makerspace
- Host your Event at the Makerspace

What is a GPU

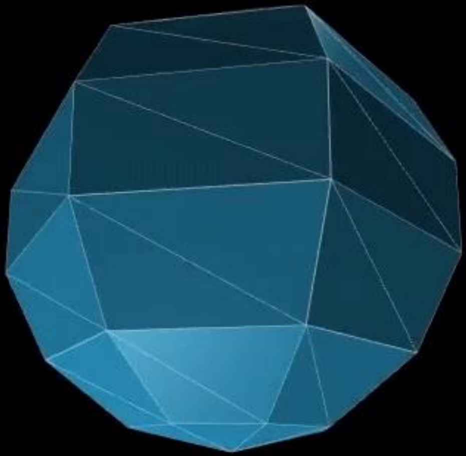
- Graphics Processing Unit
- Separate hardware designed to process graphics



Why all this trouble for GPU



I want to move a vertex



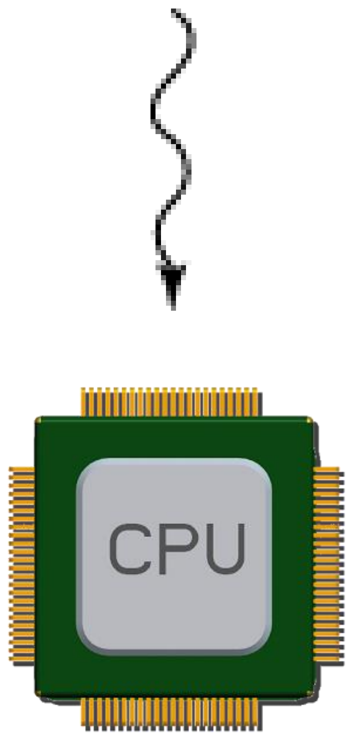
...LOTS of vertices

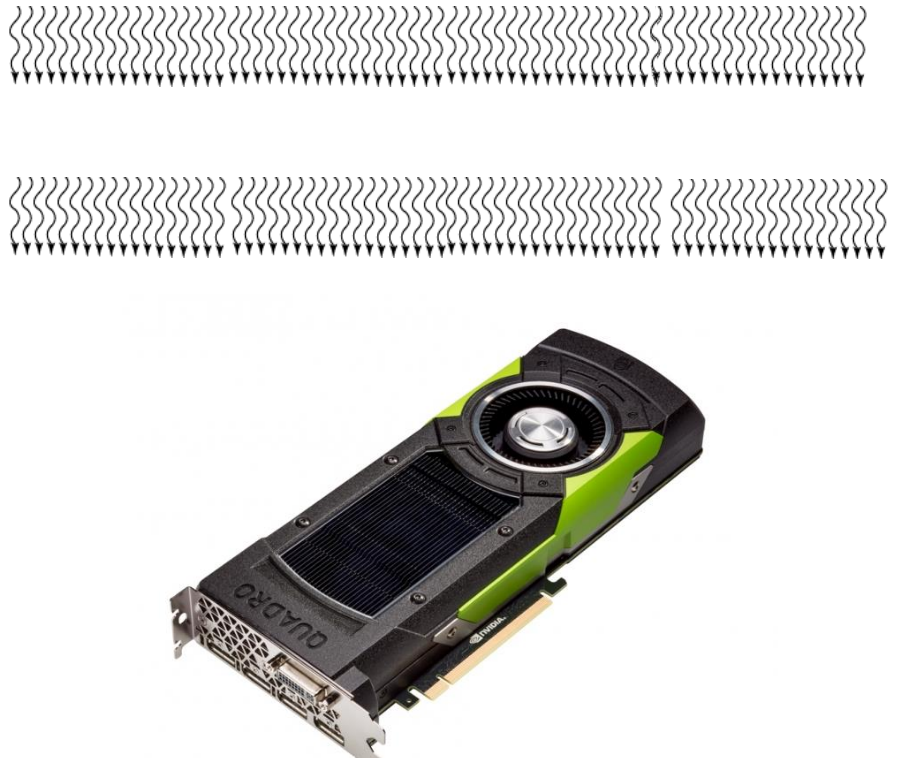
$$\frac{1 \text{ second}}{60 \text{ frames}} = 16.66 \text{ ms per frame}$$

Translation matrix

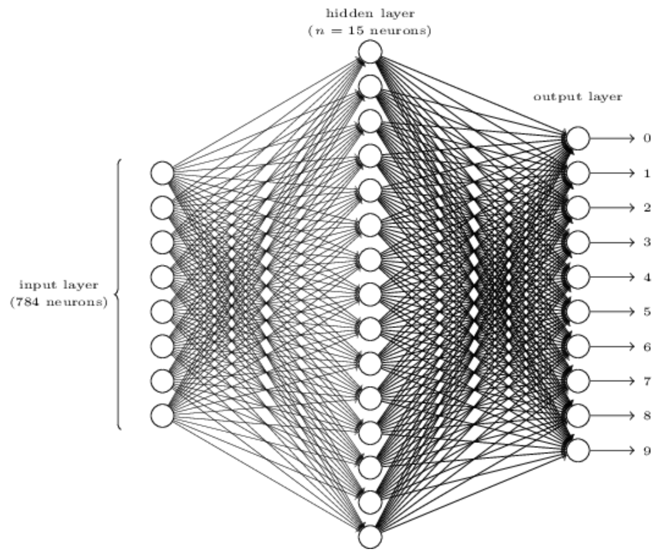
$$\begin{bmatrix} 1 & 0 & 0 & dx \\ 0 & 1 & 0 & dy \\ 0 & 0 & 1 & dz \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x + dx \\ y + dy \\ z + dz \\ 1 \end{bmatrix}$$

GPU designed for parallel





GPU outside graphics

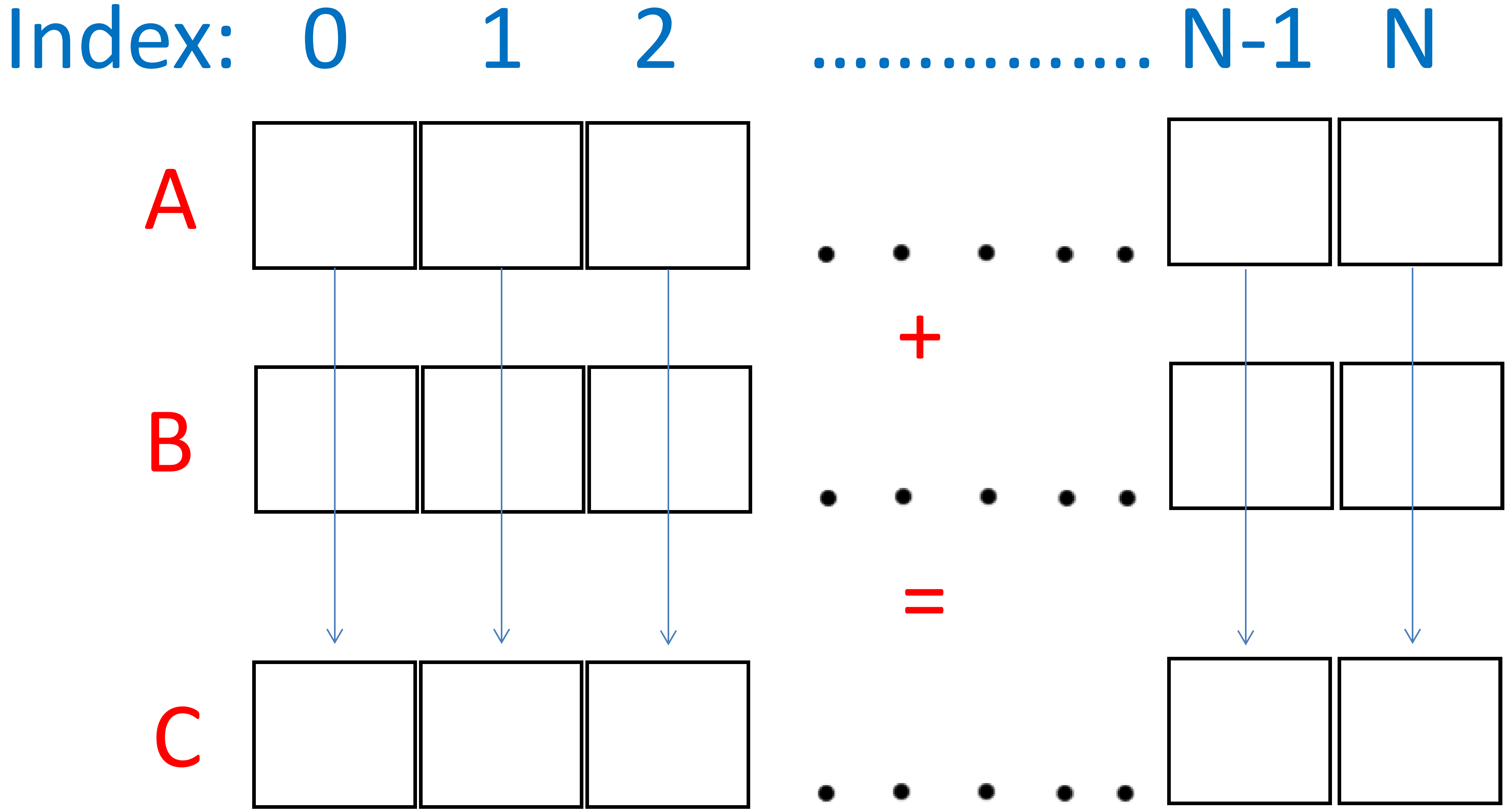


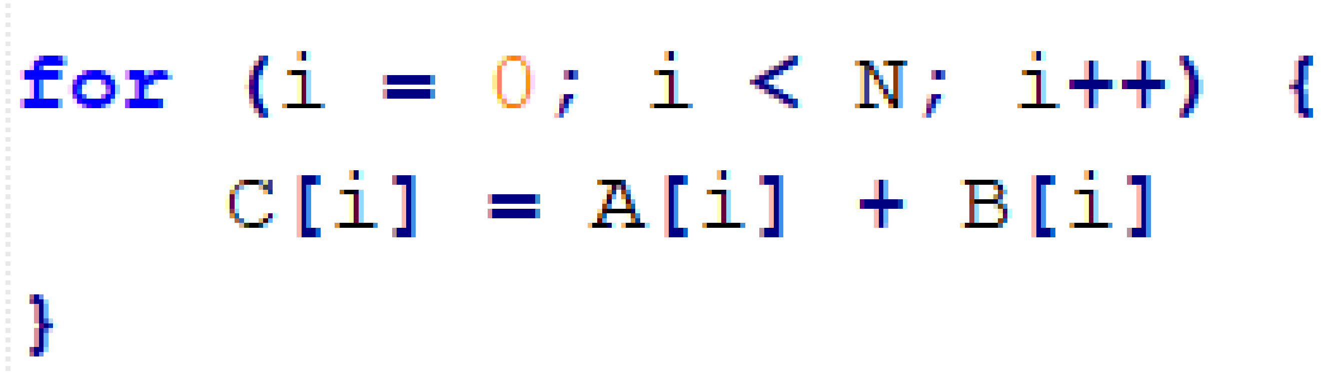
ethereum

Pr

MAX

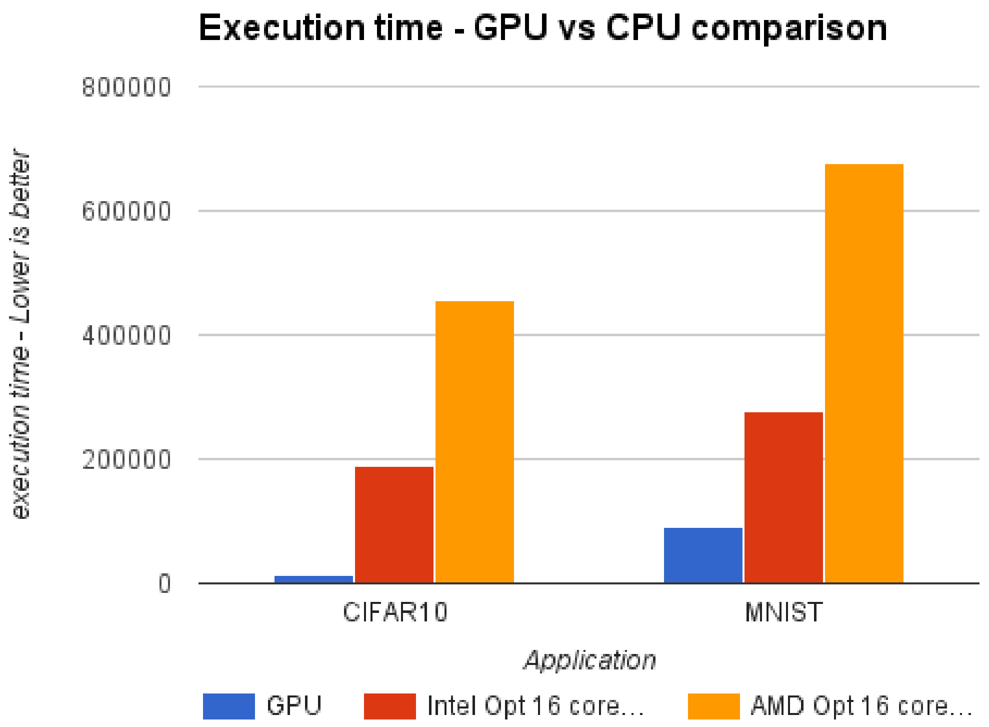
Case example

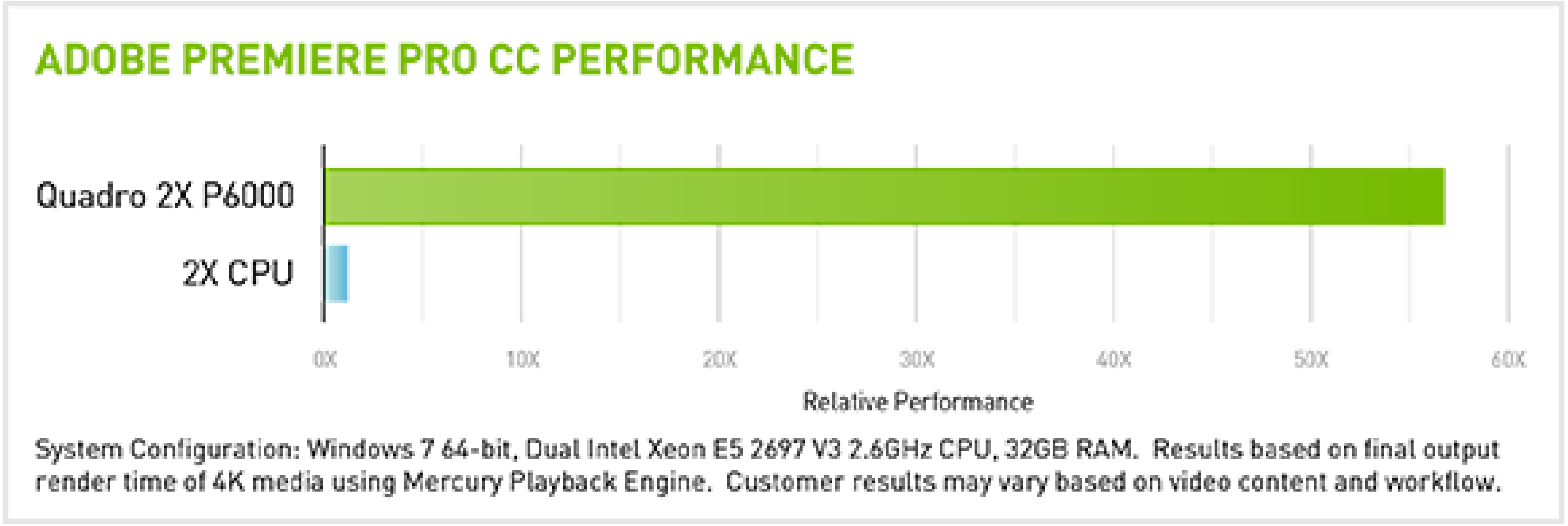




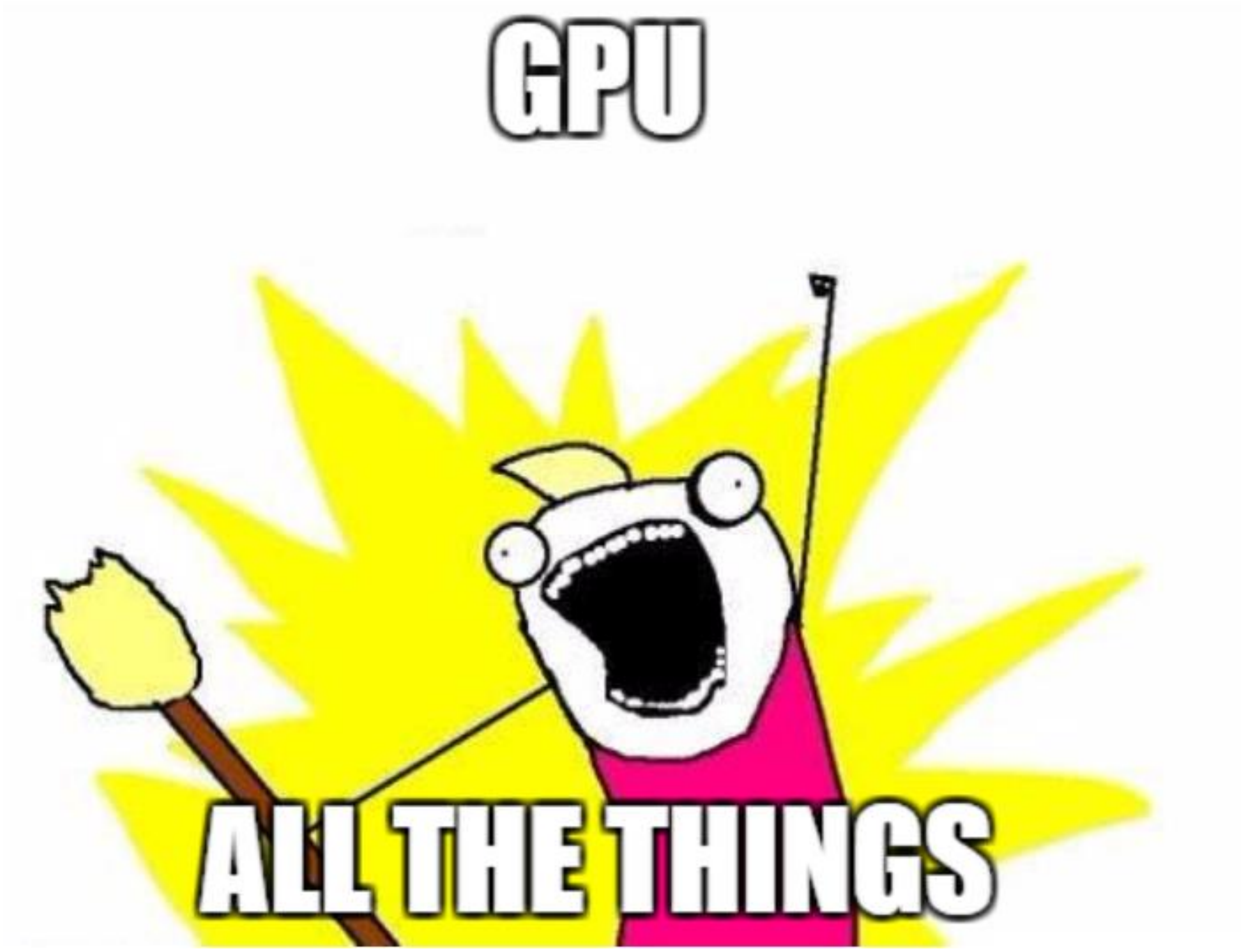
```
for (i = 0; i < N; i++) {
    C[i] = A[i] + B[i]
}
```

- N = 10 billion
- CPU @ 3Ghz – 1 thread
  - 3.3 seconds
    - Not considering the rest of the computer can’t use the CPU for this time
- GPU @ 1Ghz – 4096 threads
  - 2.45 milliseconds

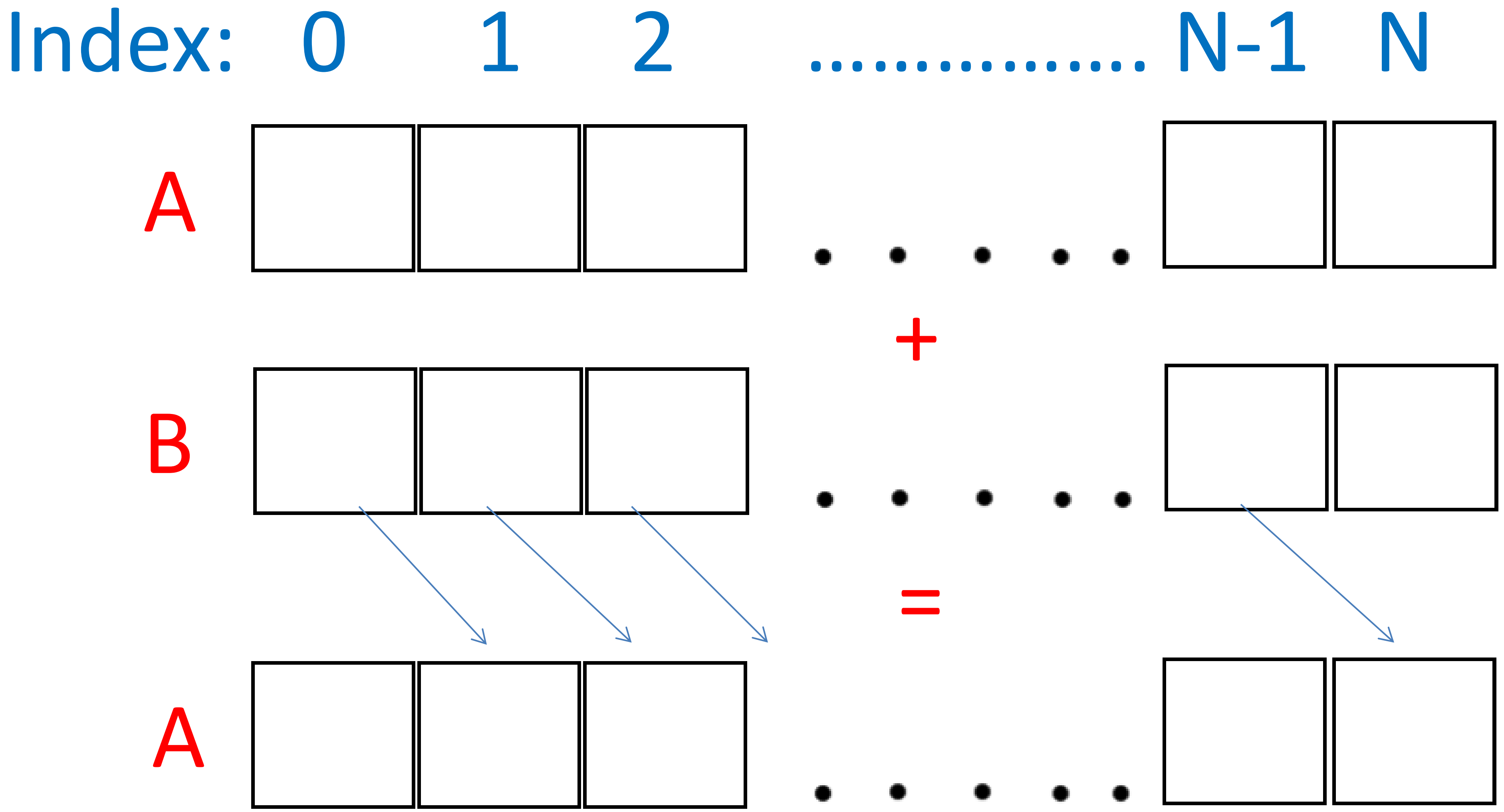




System Configuration: Windows 7 64-bit, Dual Intel Xeon E5 2697 V3 2.6GHz CPU, 32GB RAM. Results based on final output render time of 4K media using Mercury Playback Engine. Customer results may vary based on video content and workflow.


GPU
ALL THE THINGS

Case example



```
for (i = 0; i < N-1; i++) {
    A[i + 1] = A[i] + B[i]
}
```

- A[4] = A[3] + B[3] and A[3] = A[2] + B[2]
- Run in parallel
  – no way to know which happens first

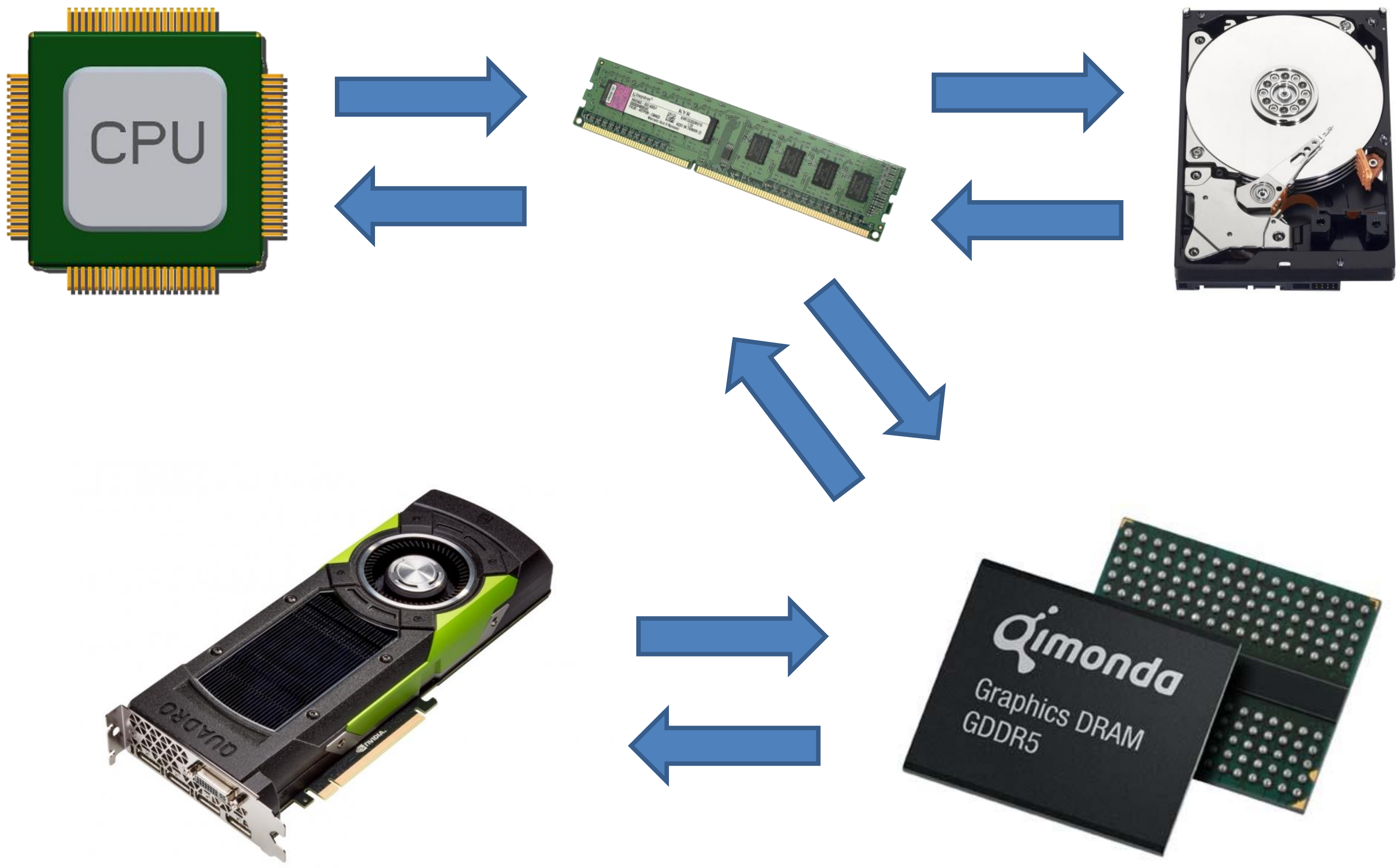
Data transfer is bottleneck

- CPU takes 1 second to compute
- GPU takes 0.00001 seconds to compute
- Takes 1 second to transfer from CPU to GPU

Dual GPU

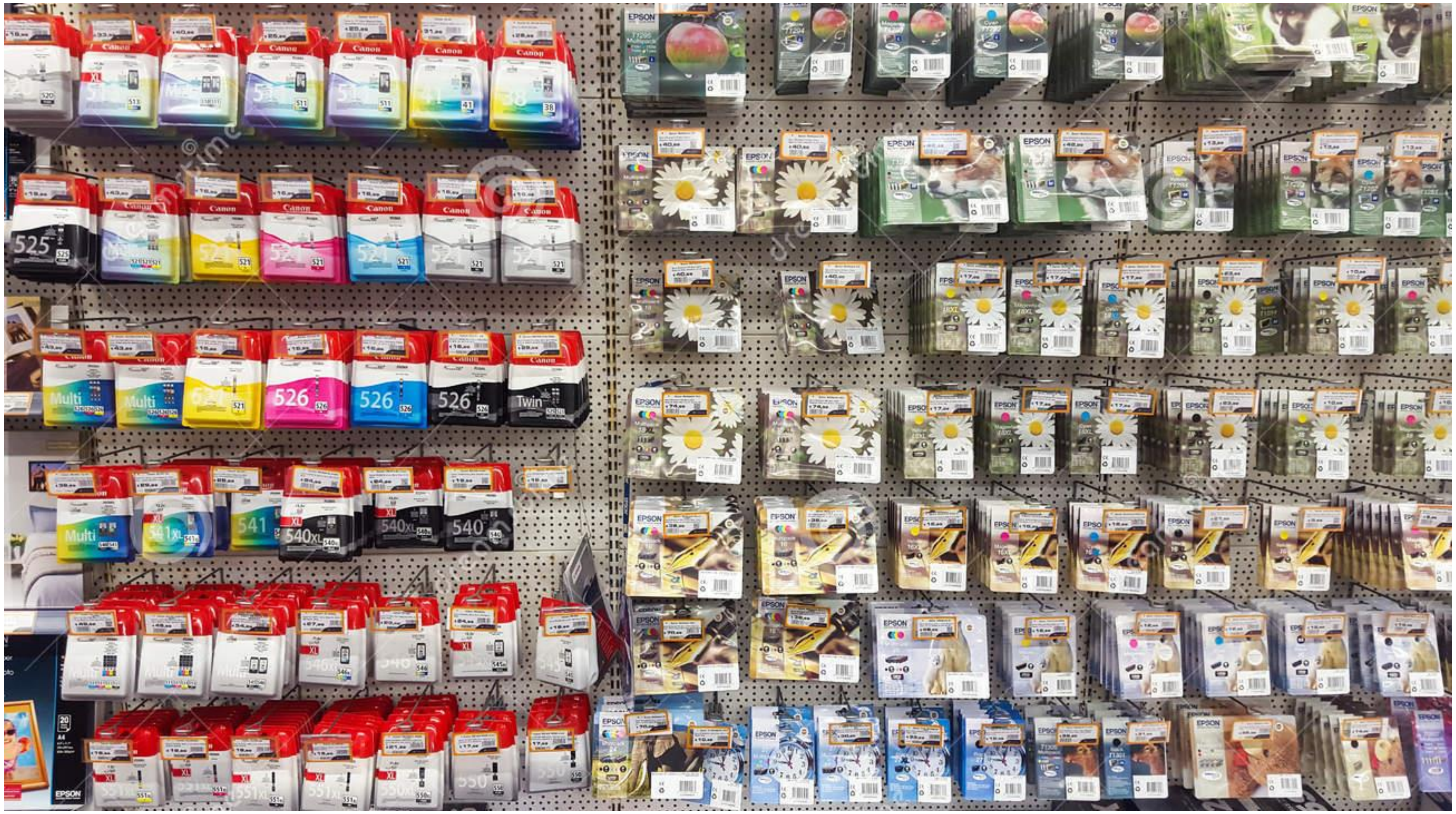
- Can have some tricky bugs of own
- Increase GPGPU computation
- Increase total graphical computation
- Not useful for live graphics, such as games
  - Need to share same memory
  - Only can process it at same time

Current Computer Architecture



What if printers all had different ink cartridges

Oh wait…



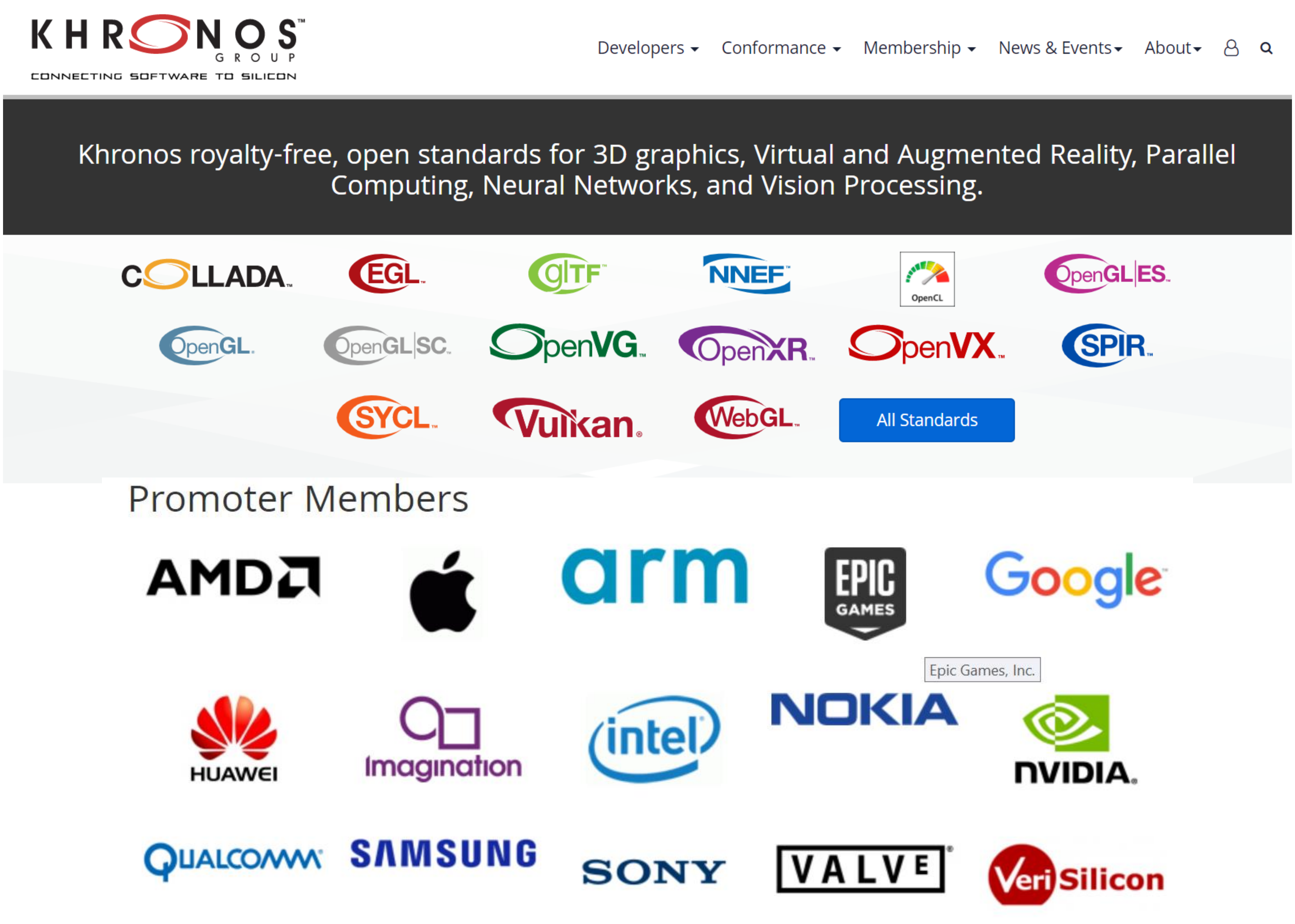
Imagine if GPU were the same

NVIDIA®

AMD

intel®

QUALCOMM®



Khronos royalty-free, open standards for 3D graphics, Virtual and Augmented Reality, Parallel Computing, Neural Networks, and Vision Processing.

Promoter Members

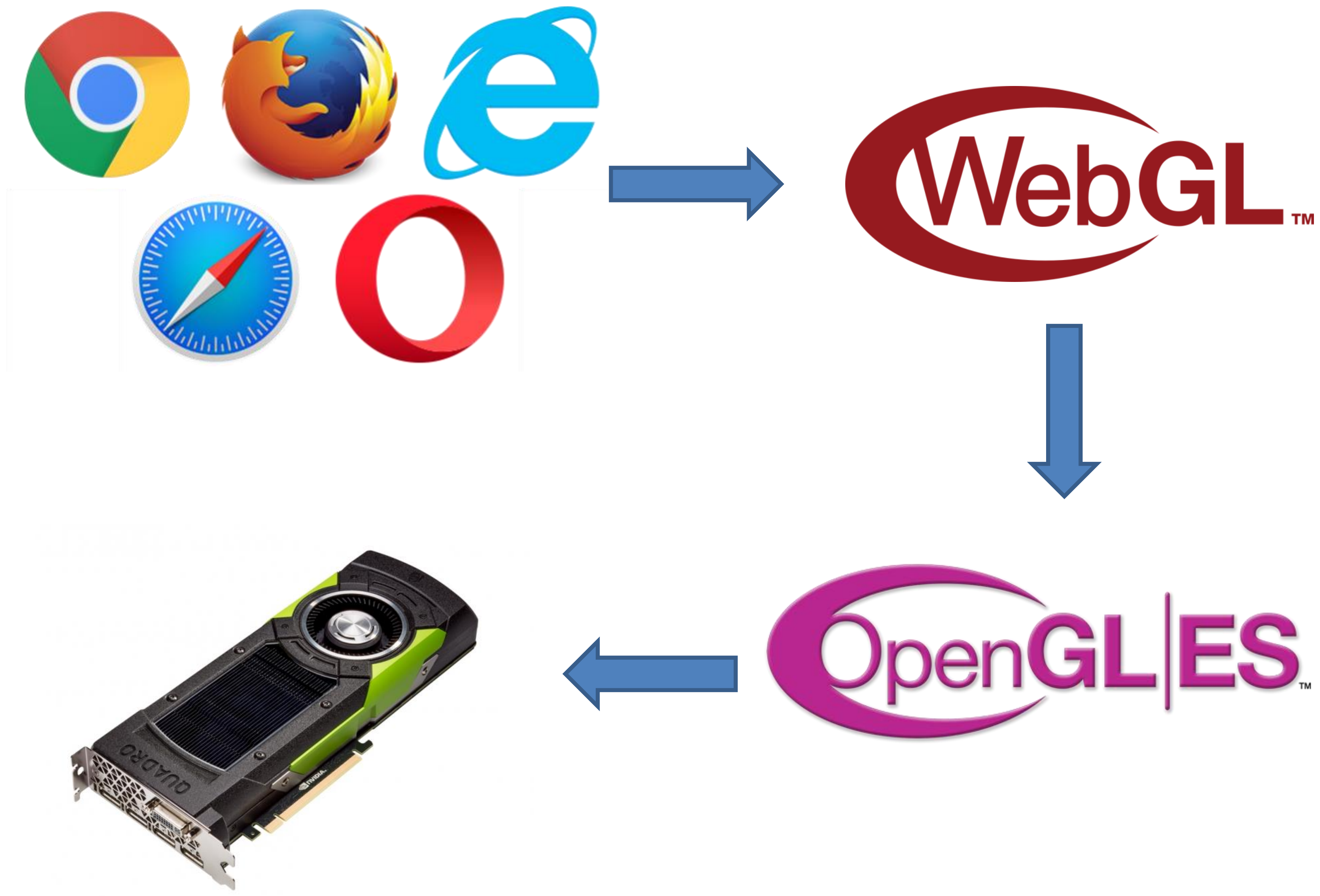
- Set of functions to talk to GPU
- Almost everything supports now
- All C/C++ code

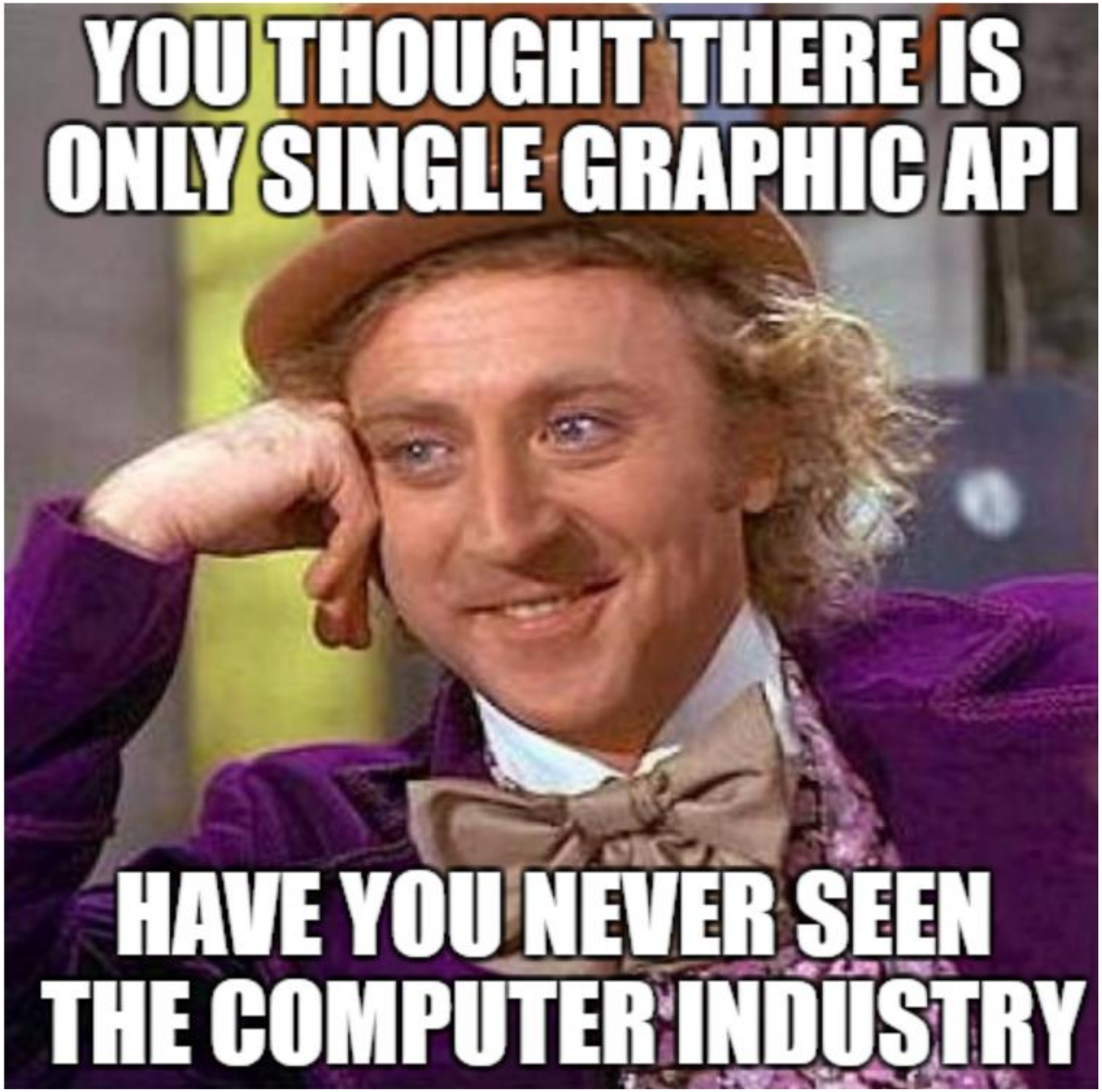
- ES stands for “Embedded Systems”
- Lighter version, runs on Phones

WebGL™

- All browsers agree on a spec
- Talks to OpenGL under the hood
- Lets browsers use GPU for 3D graphics
- Supports around 98% of devices in world

How it all works




YOU THOUGHT THERE IS ONLY SINGLE GRAPHIC API
HAVE YOU NEVER SEEN THE COMPUTER INDUSTRY

Vulkan™

- “New” OpenGL
- Aim to be truly cross-platform

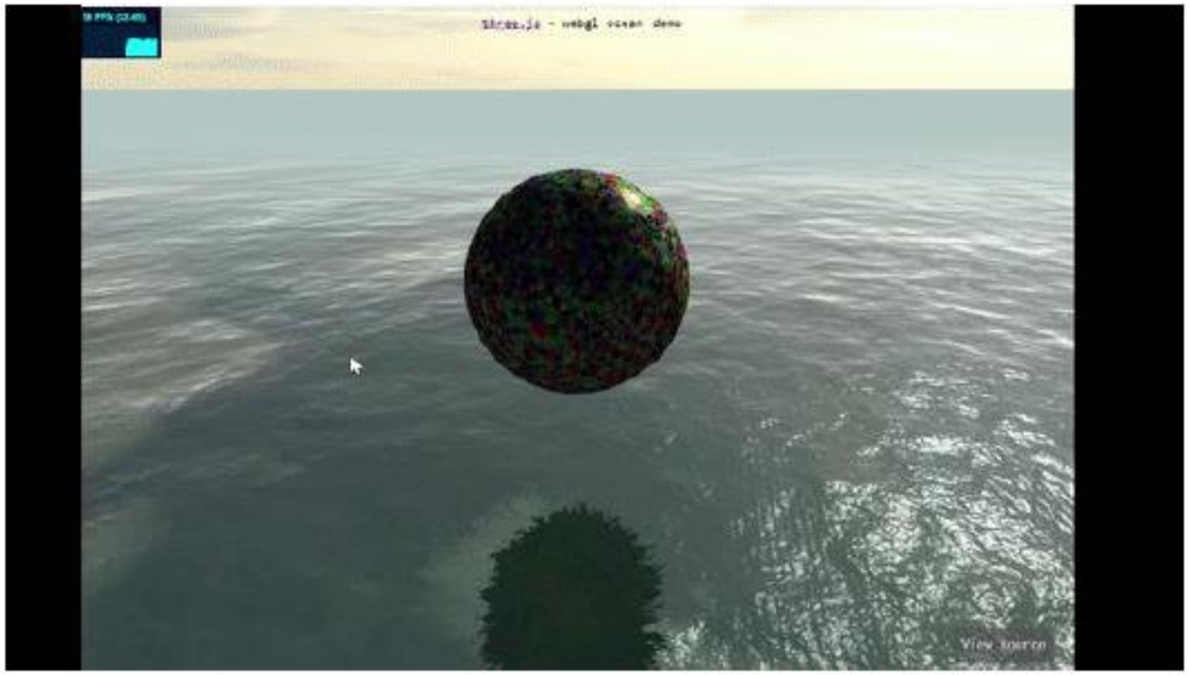
Metal

- Apple own GPU API
- Very closed community
- Designed by Apple for Apple

DirectX 12

- Runs on Windows .NET Runtime
- Most games currently use
- Uses left-hand rule cause Windows

Time for some ThreeJS now!



Level of Abstraction

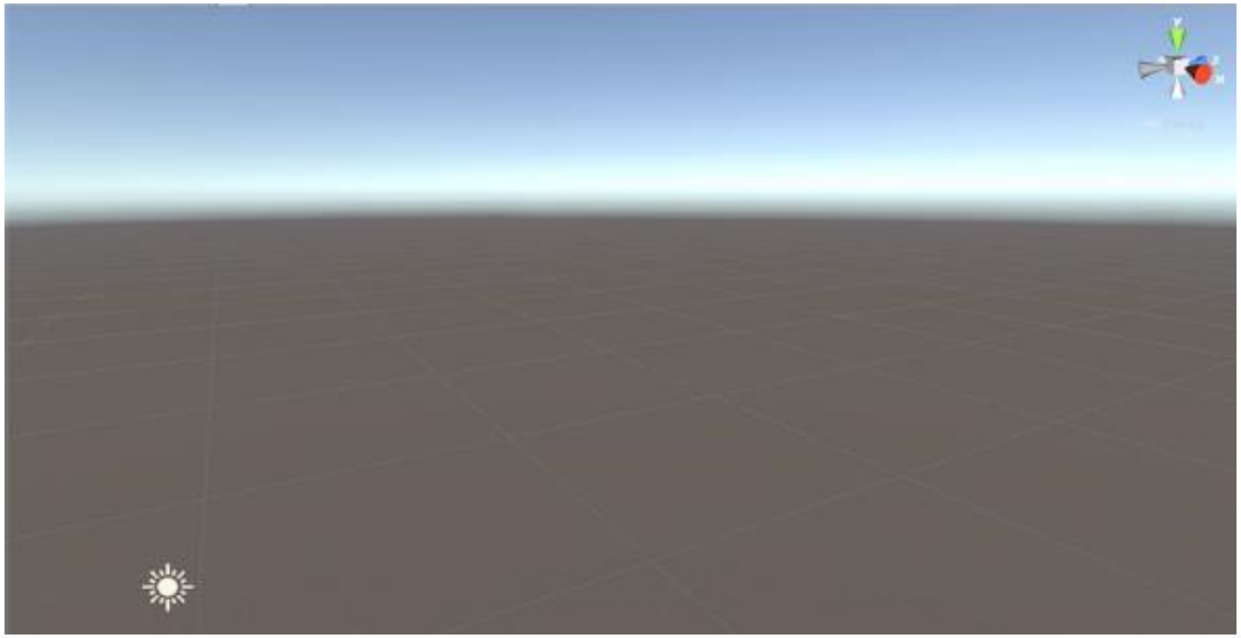
- High Level
  - Unity or Unreal Engine
  - **Pros:** Can make cool things fast
  - **Cons:** Performance cost and programmable limits
- Lower Level
  - Raw WebGL/OpenGL
  - **Pros:** Limit is your imagination
  - **Cons:** Limit is your time and patience
    - Taught in CS 559 – Will teach you a LOT about graphics

The middle ground

- ThreeJS
  - Framework that wraps WebGL
  - High level to make developing fast and easy
  - Low level to still let you do anything
  - Works basically on every device

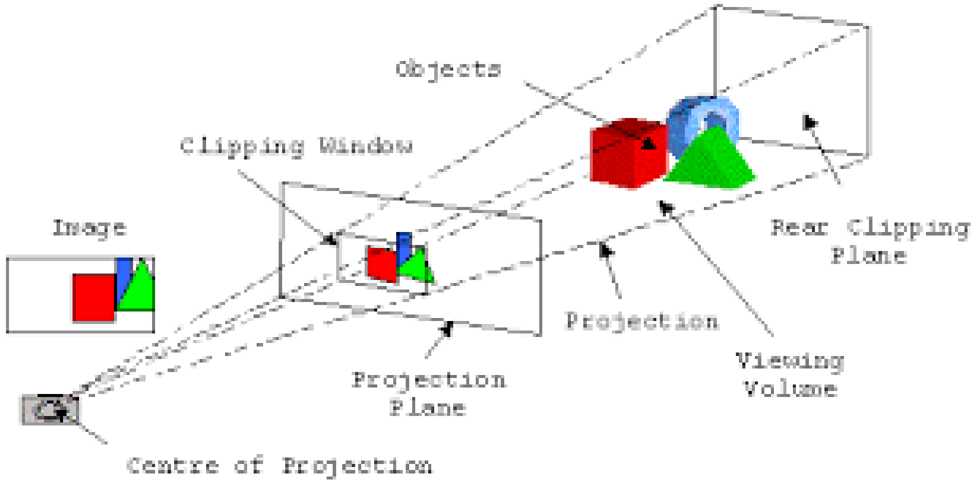
The Scene

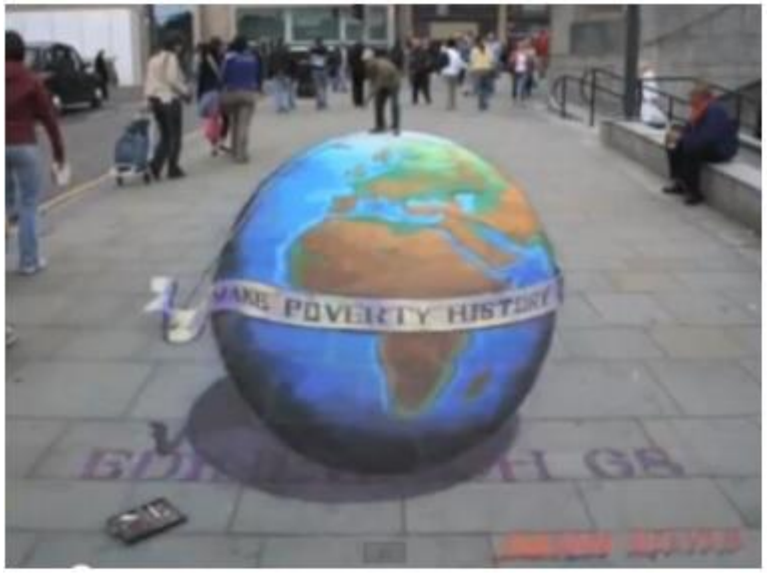
- The entire 3D realm your graphics live
- Everything has a (x,y,z) coordinate

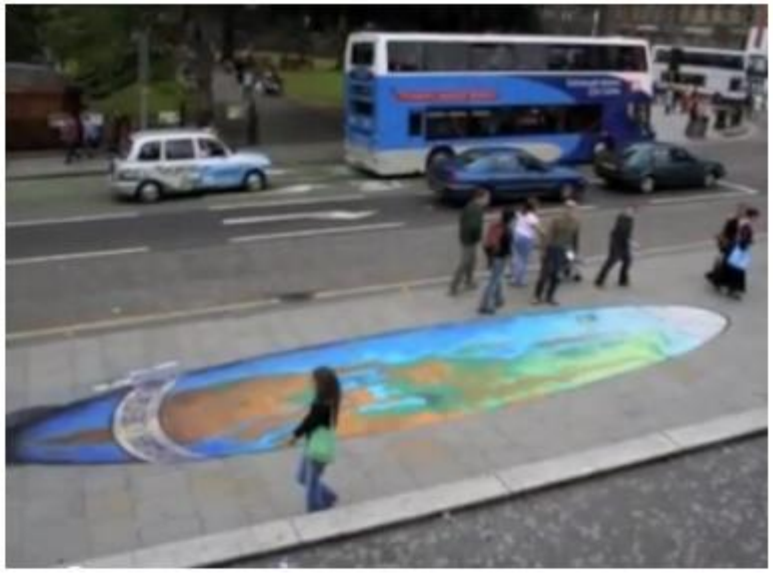


```
var scene = new THREE.Scene();
```

Camera – The View

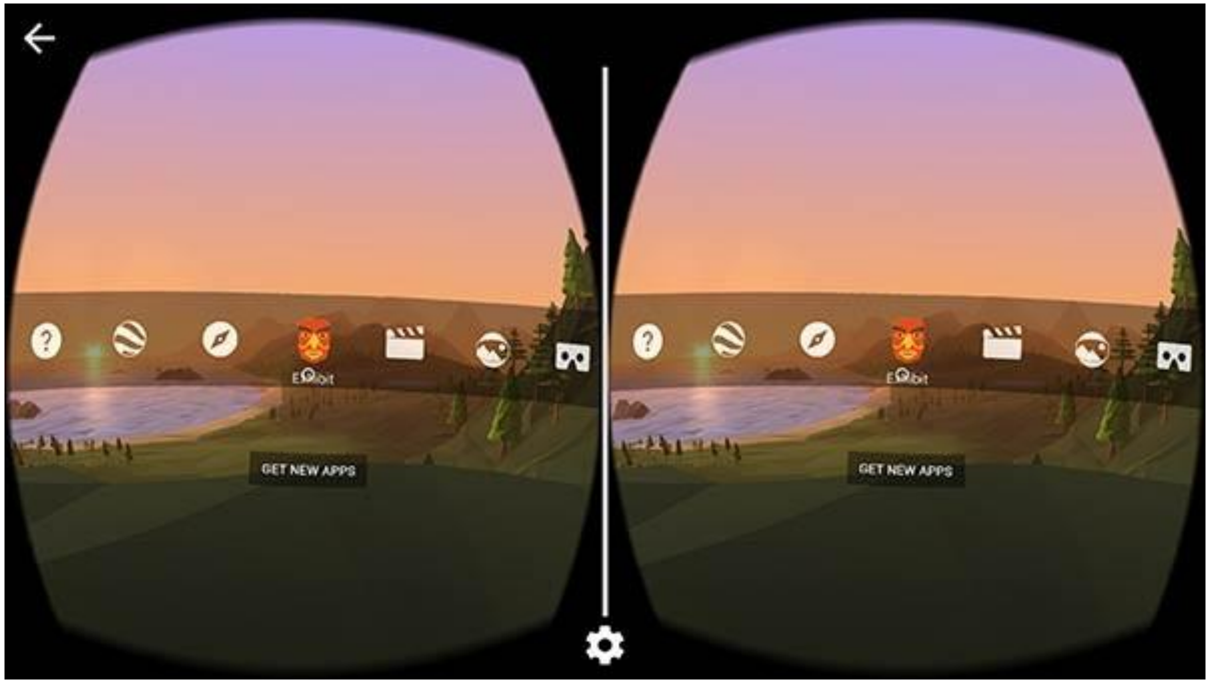






Camera – The View

- VR isn't too much different
- Same camera, split into two views for headset

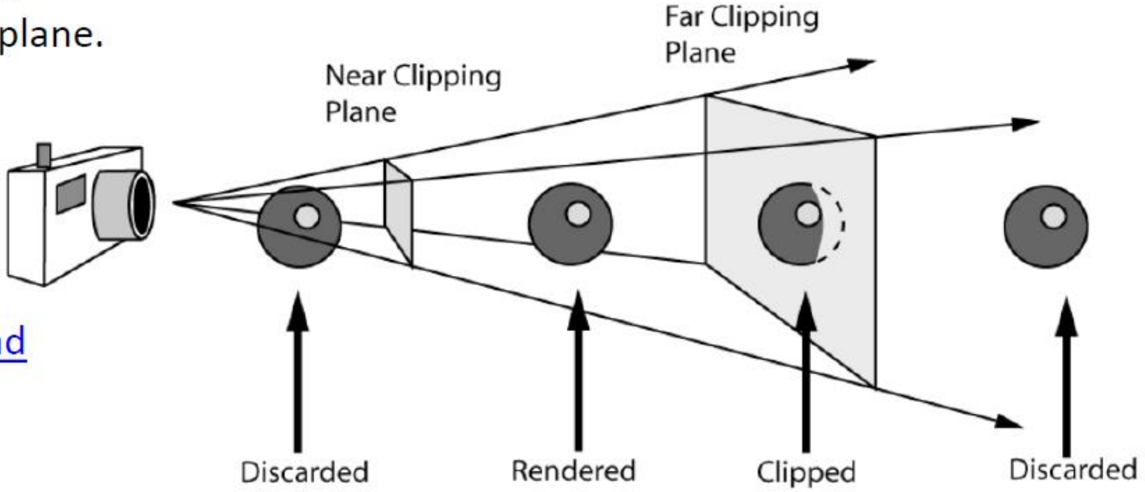


Camera in ThreeJS

```
var camera = new THREE.PerspectiveCamera( 45, width / height, 1, 1000 );
scene.add( camera );
```

PerspectiveCamera(fov, aspect, near, far)
PerspectiveCamera(45, width/height , 1, 1000)

fov — Camera vertical field of view angle
aspect — Camera frustum aspect ratio (16:9, 4:3, etc)
near — Camera near plane.
far — Camera frustum far plane.



http://the3dwebcoder.typepad.com/blog/2015/04/webgl-101-getting-started.html

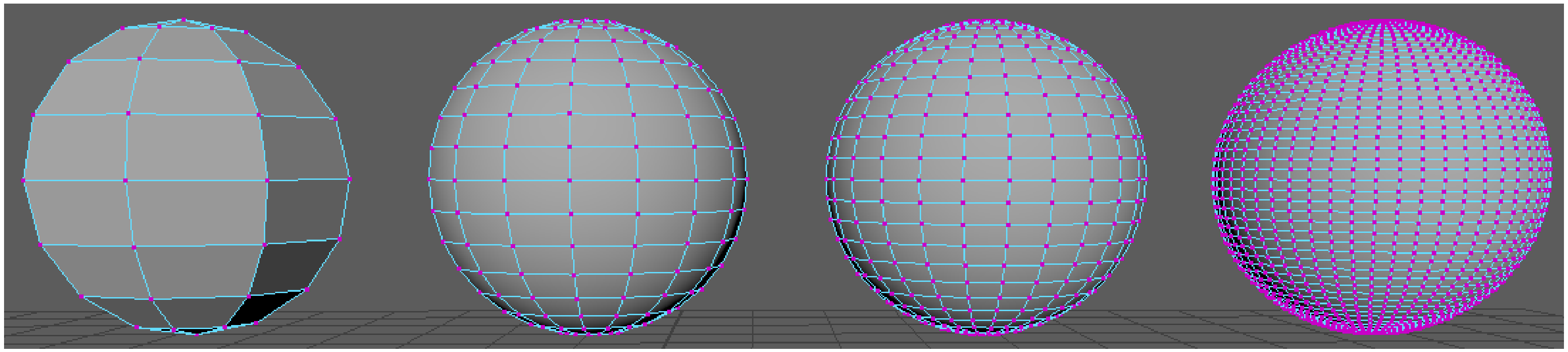
3D Models

- Nurbs
  - Mathematically based
  - SolidWorks, AutoCAD
  - Used for realistic modeling
  - Hard to model
- Pologonal
  - Shape made up of individual Vertices
  - Maya, Blender, 3ds Max
  - Can look like anything you want
  - Used for Movies CGI, Games, VR, etc

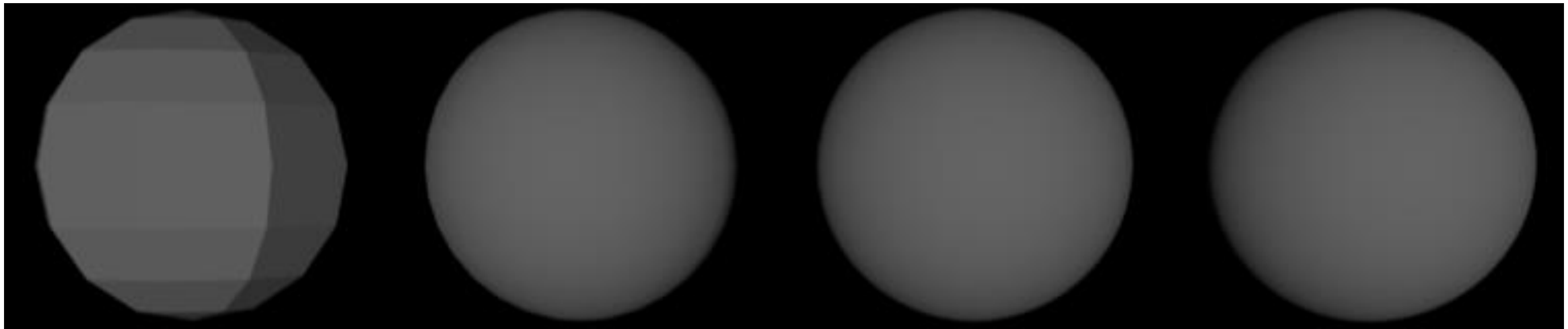
3D models

- Collections of **Vertices** that each have (x,y,z)
- 3 or more **Vertices** make a **Face**
- Ex. Cube
  - Has 8 Vertices, 12 Edges, 6 Faces

- Poly Count
  - The more vertices, the more realistic/detailed
    - Also more computer has to computer.

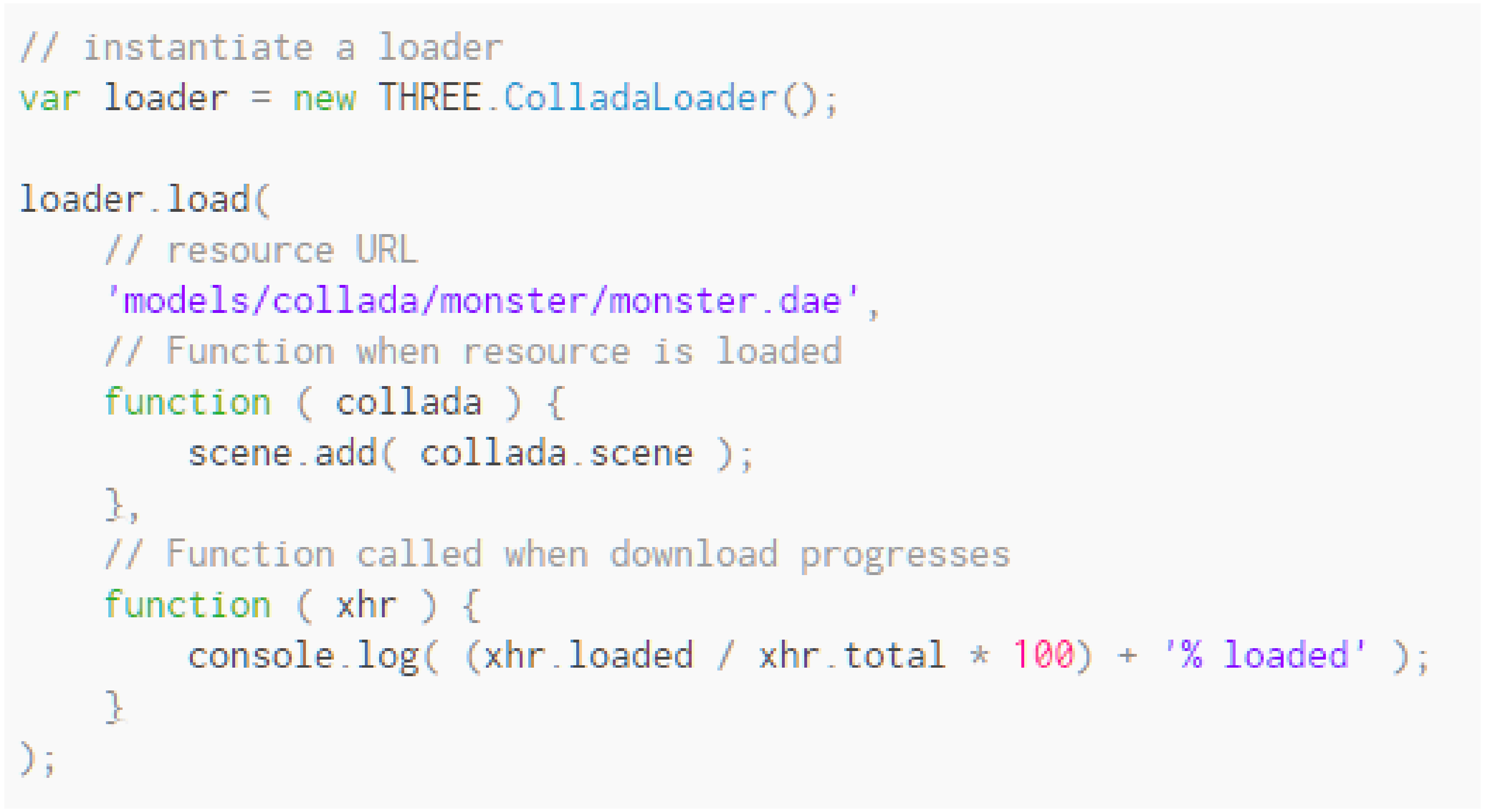


58 Verties	242 Verties	554 Verties	2452 Verties
64 Faces	256 Faces	576 Faces	2500 Faces



Importing Models – THREE.js

- THREE.js has “loaders” for various types

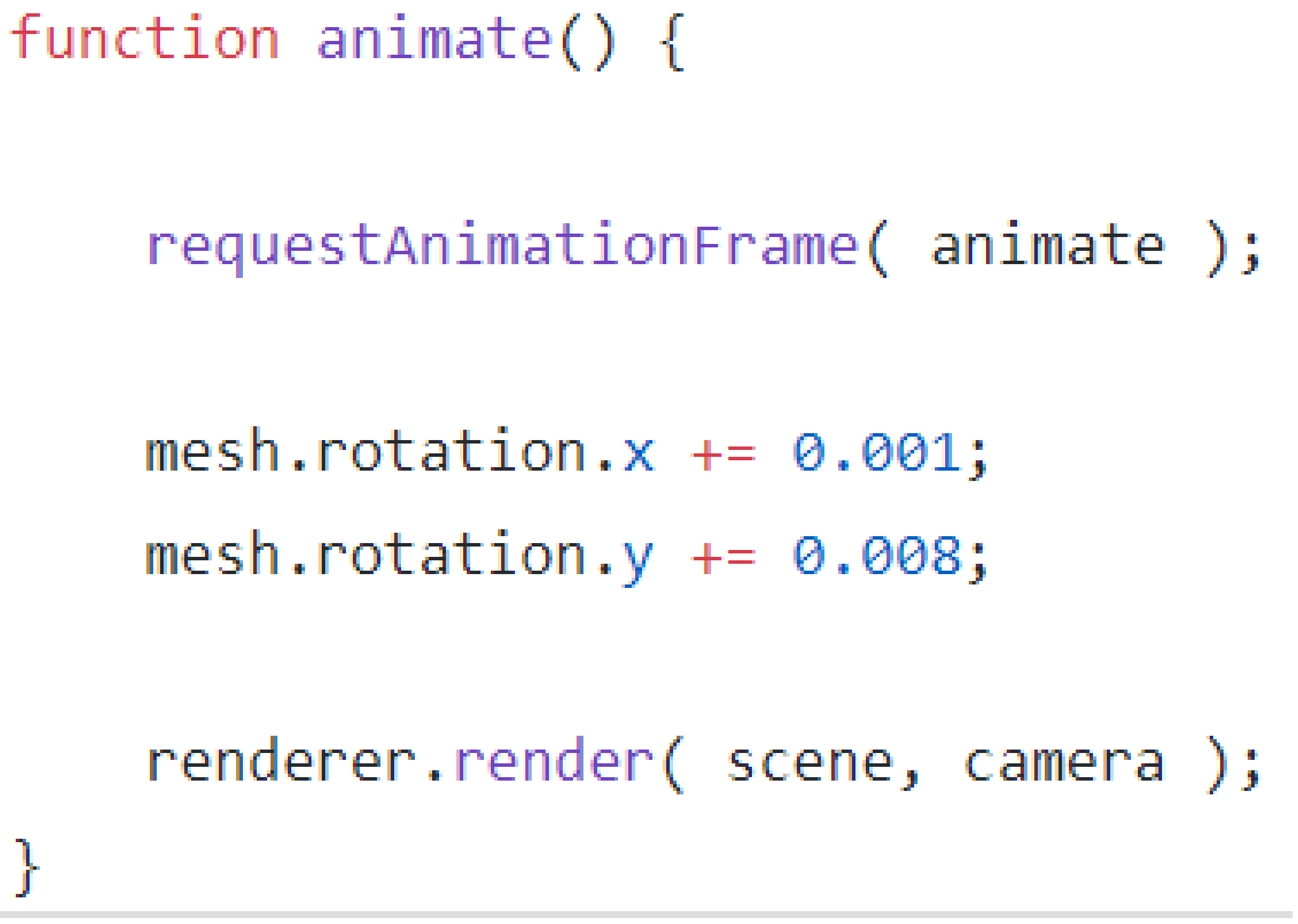


```
// instantiate a loader
var loader = new THREE.ColladaLoader();

loader.load(
    // resource URL
    'models/collada/monster/monster.dae',
    // Function when resource is loaded
    function ( collada ) {
        scene.add( collada.scene );
    },
    // Function called when download progresses
    function ( xhr ) {
        console.log( (xhr.loaded / xhr.total * 100) + '% loaded' );
    }
);
```

Animation Cycle

- Everytime the program computes and displays the image results
- Called “Frames”
- Note: 1sec / 60 = 16.6 ms
  - That’s not much time for a computer to compute everything it needs

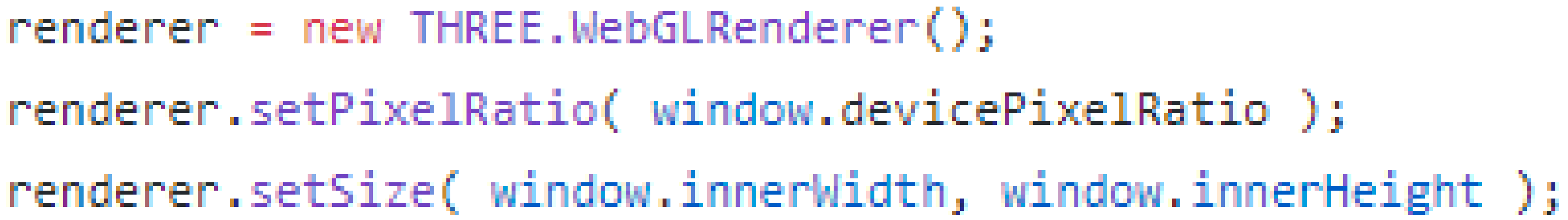
Animation Cycle – THREE.js



```
function animate() {

    requestAnimationFrame( animate );

    mesh.rotation.x += 0.001;
    mesh.rotation.y += 0.008;

    renderer.render( scene, camera );
}
```

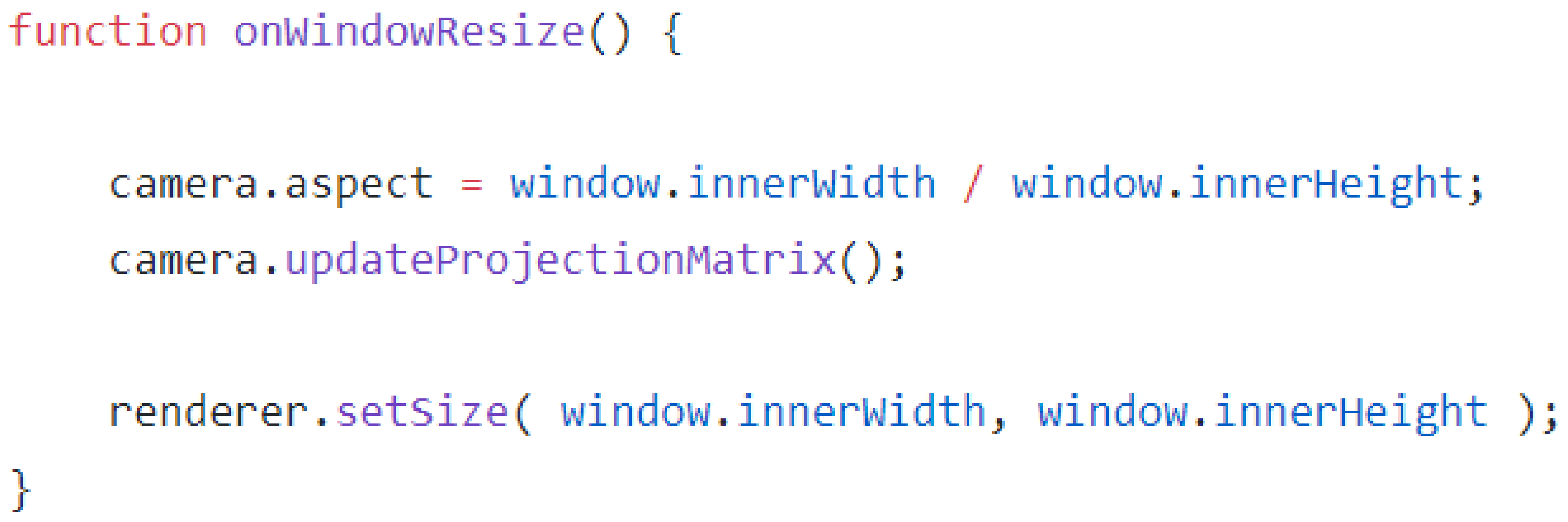
Renderer

- The part that makes the WebGL calls
- LOTS of “magic” under the hood
- Accept it works to start off



```
renderer = new THREE.WebGLRenderer();
renderer.setPixelRatio( window.devicePixelRatio );
renderer.setSize( window.innerWidth, window.innerHeight );
```

Animation Window Resize

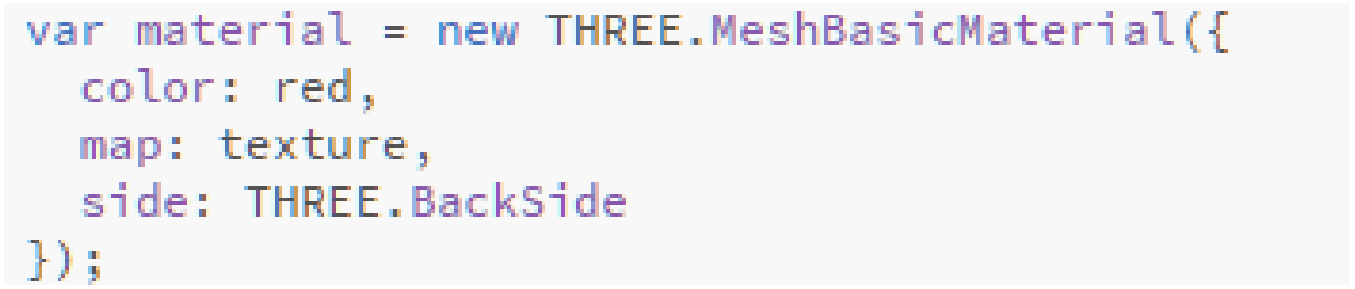


```
function onWindowResize() {

    camera.aspect = window.innerWidth / window.innerHeight;
    camera.updateProjectionMatrix();

    renderer.setSize( window.innerWidth, window.innerHeight );
}
```

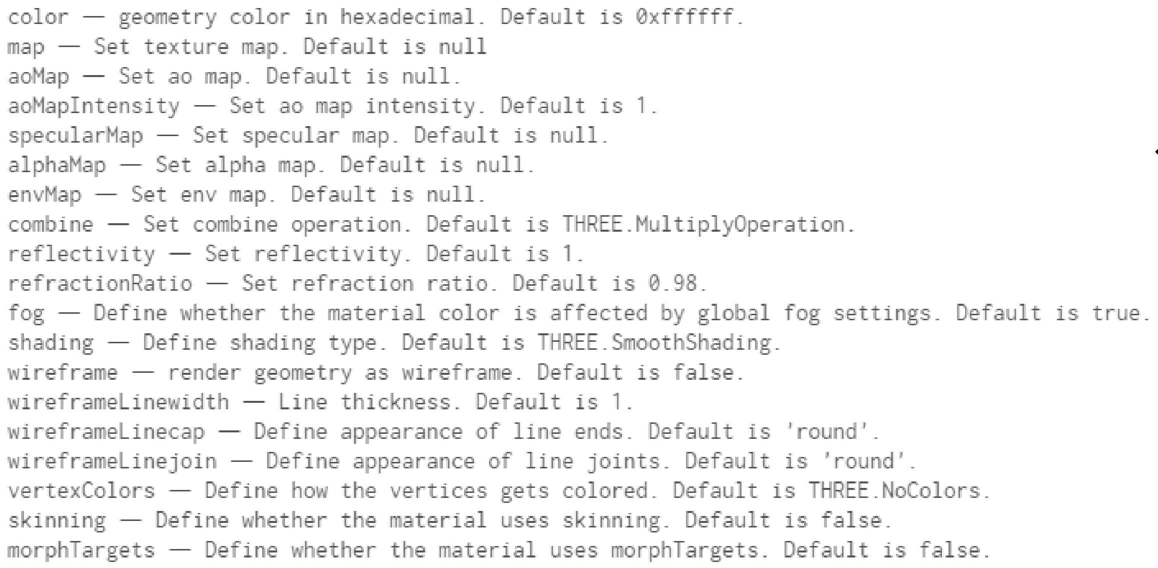
Material and Textures

- Material != Textures
- Material can have multiple textures on them
- Each object has only 1 material on it.
  - A more complex object can be made of many smaller objects (ex. Car has different materials for tires then doors)
- Materials hold info like Color, Transparency, Reflection, etc.
- Textures are generally pictures (.jpeg, .png, etc.)

Material – THREE.js



```
var material = new THREE.MeshBasicMaterial({
  color: red,
  map: texture,
  side: THREE.BackSide
});
```



```
color — geometry color in hexadecimal. Default is 0xffffff.
map — Set texture map. Default is null
aoMap — Set ao map. Default is null.
aoMapIntensity — Set ao map intensity. Default is 1.
specularMap — Set specular map. Default is null.
alphaMap — Set alpha map. Default is null.
envMap — Set env map. Default is null.
combine — Set combine operation. Default is THREE.MultiplyOperation.
reflectivity — Set reflectivity. Default is 1.
refractionRatio — Set refraction ratio. Default is 0.98.
fog — Define whether the material color is affected by global fog settings. Default is true.
shading — Define shading type. Default is THREE.SmoothShading.
wireframe — render geometry as wireframe. Default is false.
wireframeLinewidth — Line thickness. Default is 1.
wireframeLinecap — Define appearance of line ends. Default is 'round'.
wireframeLinejoin — Define appearance of line joints. Default is 'round'.
vertexColors — Define how the vertices gets colored. Default is THREE.NoColors.
skinning — Define whether the material uses skinning. Default is false.
morphTargets — Define whether the material uses morphTargets. Default is false.
```

← Many parameters possible to set

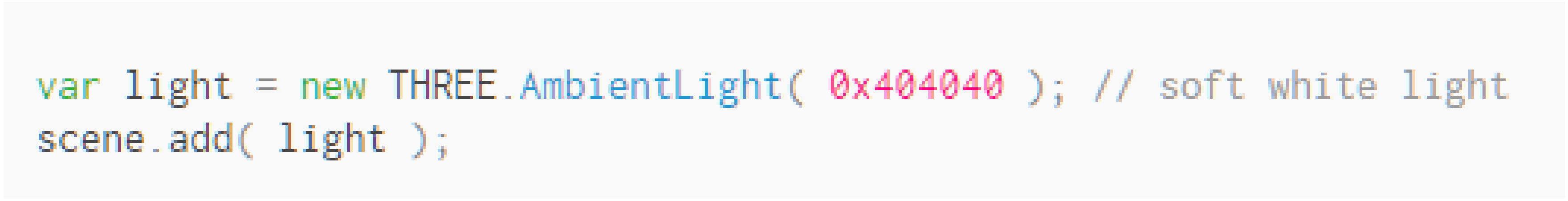
Lights

- #1 answer to “why is my scene not loading?”
- Can have multiple light sources
- Different types
- Can set light color, intensity, direction, etc
- Light controls how materials look like
  - Ex. Red in bright light looks different then dim light
- Could take a whole semester course in lighting both theoretical or practical

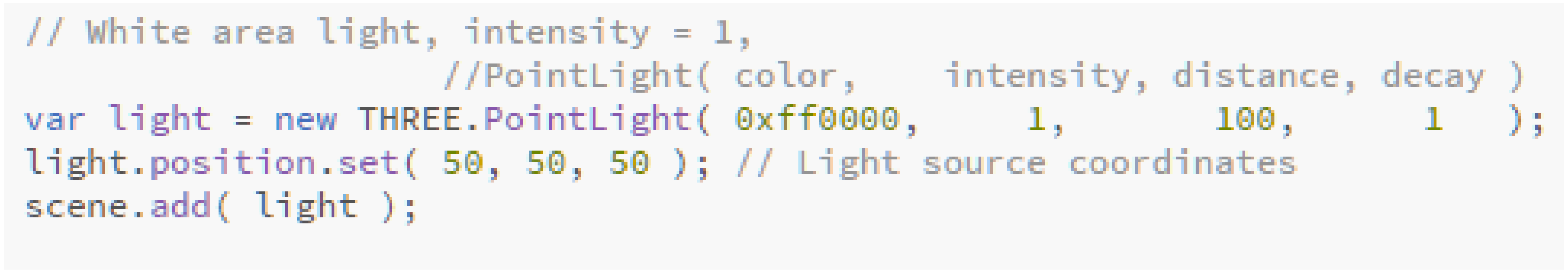
Light Types

- Ambient
  - Objects have basic light to them, no direction therefore no shadows
  - Default in ThreeJS
- Direction
  - Simulates a even light source all aimed in same direction
  - Most common type for a sun
- Point / Spot
  - Light has a source coordniate and is pointed in a direction
  - Equivlent of turning a flash light on in a dark room
- Area
  - Light has a source coordniate
  - Emits lights in all directions around it

Lights – THREE.js



```
var light = new THREE.AmbientLight( 0x404040 ); // soft white light
scene.add( light );
```



```
// White area light, intensity = 1,
                    //PointLight( color,    intensity, distance, decay )
var light = new THREE.PointLight( 0xff0000,     1,       100,      1   );
light.position.set( 50, 50, 50 ); // Light source coordinates
scene.add( light );
```

Time to get some practice!

https://github.com/uwmadisonieee/Tutorials