

CIS367 - Computer Graphics Introduction

Erik Fredericks - frederer@gvsu.edu
W2022

WELCOME TO

H E L L

CAYMAN ISLANDS BWI



Textbook

No formal textbook for this course, but I've used this one in the past:

Interactive Computer Graphics **7th edition**
Angel and Shreiner

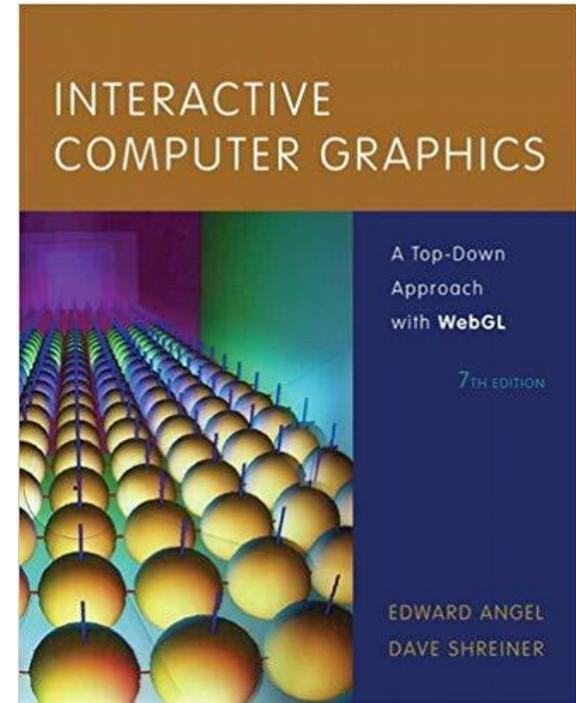
Can I get a previous version?
if you want ... this one uses WebGL though...

Why?

All prior eds use OpenGL, not WebGL

Book website:

<https://www.cs.unm.edu/~angel/BOOK/INTERACTIVE COMPUTER GRAPHICS/SEVENTH EDITION/>



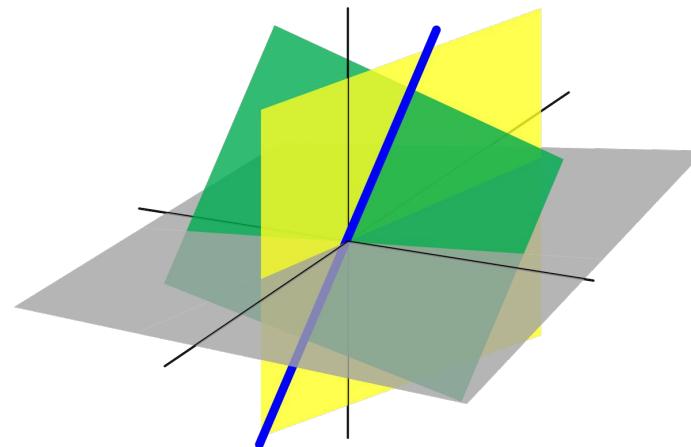
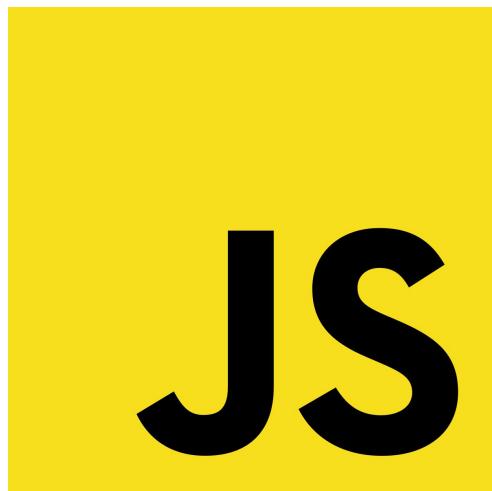
Syllabus

S y l l a b u s

A photograph showing a stack of four books. The top book is white with blue stripes on its cover. Resting on top of the stack is a row of ten light-colored wooden blocks, each featuring a black capital letter. The letters spell out the word "Syllabus".

Expectations

Ability to learn new things on the fly



No official prerequisites, but...

You should be comfortable with **coding**

- Picking up a new language (JS) is actually not too bad

And you should be reasonable with **math**

- We will cover the necessary concepts, but you may be asked to learn things outside of class

You should also be able to navigate a **Linux terminal**

What we will cover

Many things related to graphics, graphics programming, etc.
Shaders, pipelines, renders, texturing, all the buzzwords

Everything graphics-related under the umbrella of WebGL



What we will **not** cover

Raw OpenGL in your favorite language of choice

In-depth views of various tools

Blender, Unity, UnrealEngine, Godot, etc.

(the tool shouldn't matter, the concepts are important)

(we will go over Blender and Unity towards the end though)

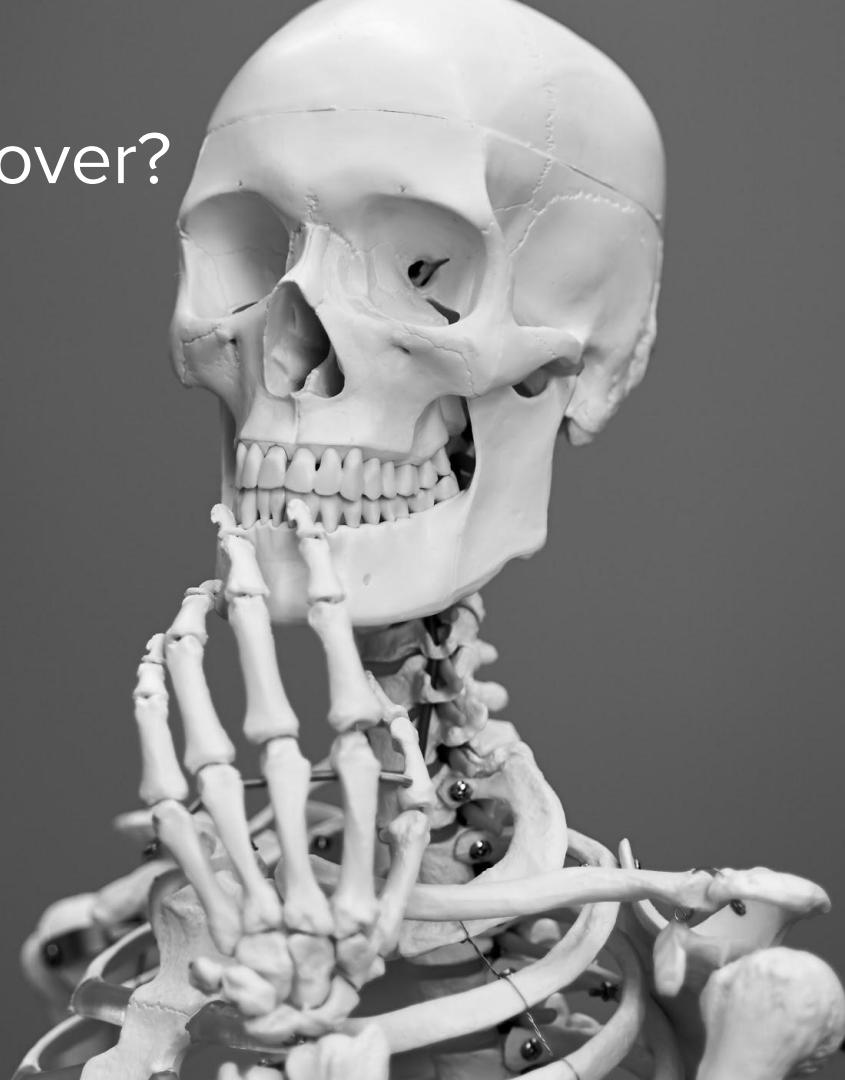
In-depth game design

HOWEVER

You can **definitely** develop a game in this class for your project!

Keep in mind it should be web-based though...

What do you want to cover?



WHOOO. ARE. YOUEEE?

Let's introduce ourselves

- Name
- Rank
- Serial number

Or how about, name, programming background, and what you want to get out of this class?

Course outline

Many in-class assignments

~**6** programming projects

2 exams

Term project

You may work in teams of 1-3, where anything > **3** must be approved **and** worthwhile

Today

Your environment

JavaScript examples

Intro to course concepts

First assignment

Your environment

All *official* homework submissions and project **must** live on EOS or GitHub Pages

- You just need to send me the link as part of your HW submissions

VPN: <https://www.cis.gvsu.edu/ciscomputinglabs/remote-connections/>

Web site: <https://www.cis.gvsu.edu/ciscomputinglabs-student-web-space/>

Your environment

Feel free to develop locally (all you need is a current browser and a text editor)

BUT

I can't grade/troubleshoot things on your computer
(and the school servers were being a pain when I was setting up this class)

Lecture examples

The **official** website for the course content is on GitHub Pages:

<https://efredericks.github.io/gvsu-cis367/>

All **homework submissions** must go through Blackboard though!

(All deadlines will be posted here as well)

<https://lms.gvsu.edu/>

Using JavaScript

Feel free to use whichever IDE/environment you prefer

Visual Studio / VSCode (download user installer)

Netbeans

Sublime

I will use...



with either



or



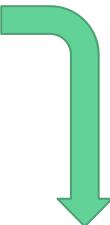
...there is a good chance I use VSCode as well though ._.

A WebGL program

JavaScript code + Browser =
WebGL program!

For this example,
HTML is green
JS is blue

This example is truncated,
full source below



```
<html>
<body>
  <canvas id="c"></canvas>
</body>
<script>
  main();

  function main() {
    const canvas = document.querySelector("#c");
    const gl = canvas.getContext("webgl");

    gl.clearColor(0.0, 0.0, 0.0, 1.0);
    gl.clear(gl.COLOR_BUFFER_BIT);
  }
</script>
</html>
```

A more fun WebGL program

<https://vchirkov.github.io/webgl-stuff/>

https://www.cs.unm.edu/~tcorriveau/WebGL%20Final%20Project/TCorriveau_FinalProject.html

Why is WebGL cool?

Works on any OS that supports a modern browser

Easy interaction provided by HTML5
Including audio/video



First, a note on 'good practices'

- Use appropriate spacing to separate out your code and make it readable
 - If I can't read your code, I will be less inclined to help you
- Ensure your code is well-commented
 - It will help both you and future-you
 - And me
- Put your header block, including your **name** in a comment in **every** file you write
- Separate out your HTML and JavaScript code

hello-world.html

```
<!--  
hello-world.html  
Erik Fredericks, W2021  
This file shows a black box.  
-->  
<html>  
<body>  
  <canvas id="c"></canvas>  
</body>  
<script src="blackbox.js"></script>  
</html>
```

blackbox.js

```
/*  
blackbox.js  
Erik Fredericks, W2021  
Basic WebGL clearing example.  
*/  
main();  
  
function main() {  
  const canvas = document.querySelector("#c");  
  const gl = canvas.getContext("webgl");  
  
  gl.clearColor(0.0, 0.0, 0.0, 1.0);  
  gl.clear(gl.COLOR_BUFFER_BIT);  
}
```

Introduction to Computer Graphics

So what are graphics **exactly**

- Creating images with a computer using
 - Hardware
 - Software
 - Applications

Graphics

How could this guitar be made?

Application:

Artist render of guitar

Software:

Maya (built on OpenGL)

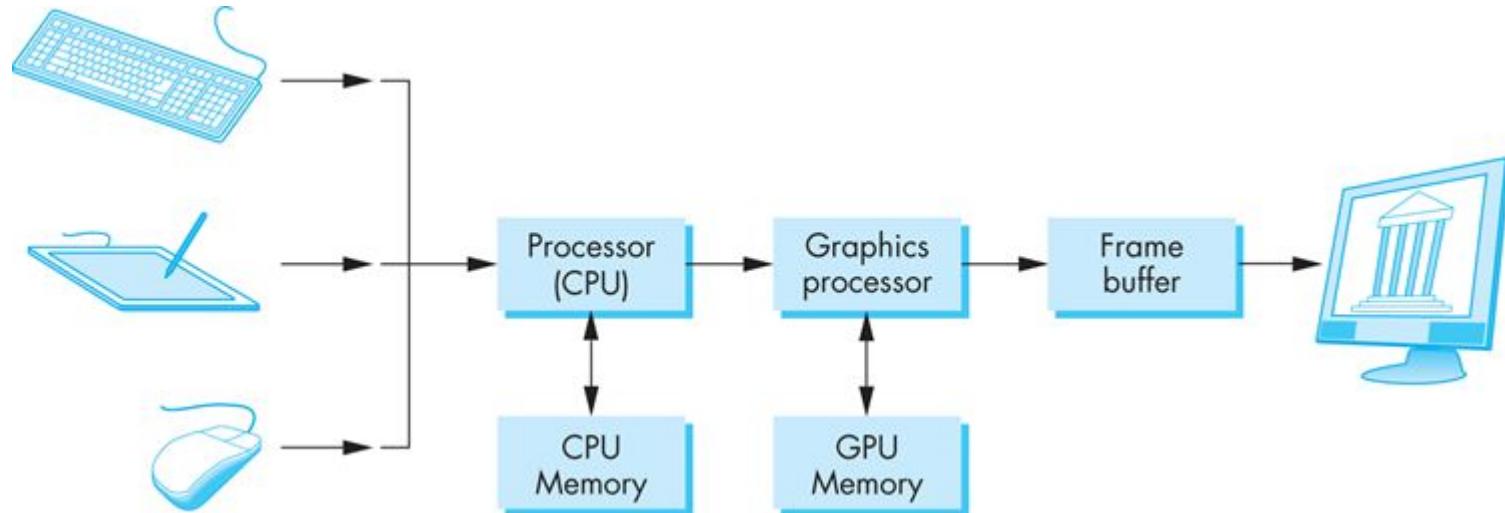
Hardware:

Graphics card on PC



https://cdn.tutsplus.com/cg/uploads/legacy/349_Maya_Guitar_Rendering/Preview.jpg

Basic Graphics System



WebGL Example

<https://shellshock.io/>

History



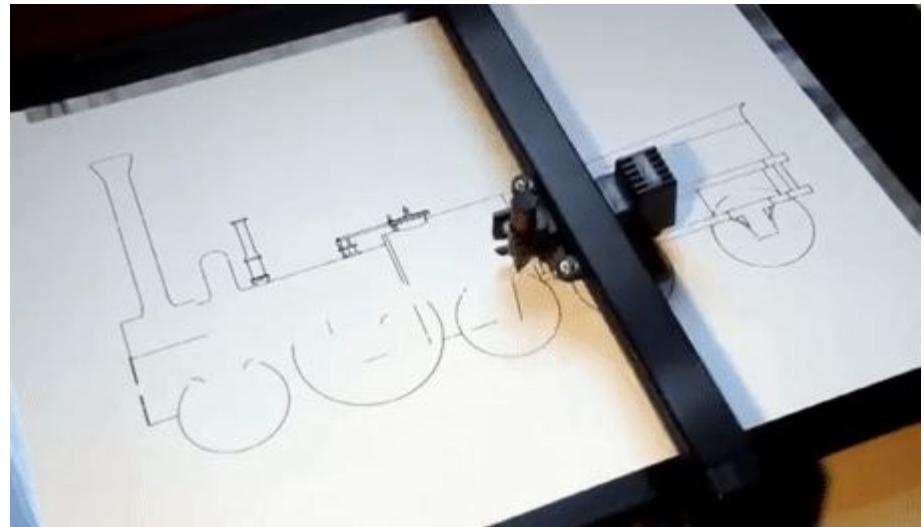
<https://www.youtube.com/watch?v=aMli33ornEU>

1950-1960

Strip charts
Pen plotters
Simple CRT

Basically:

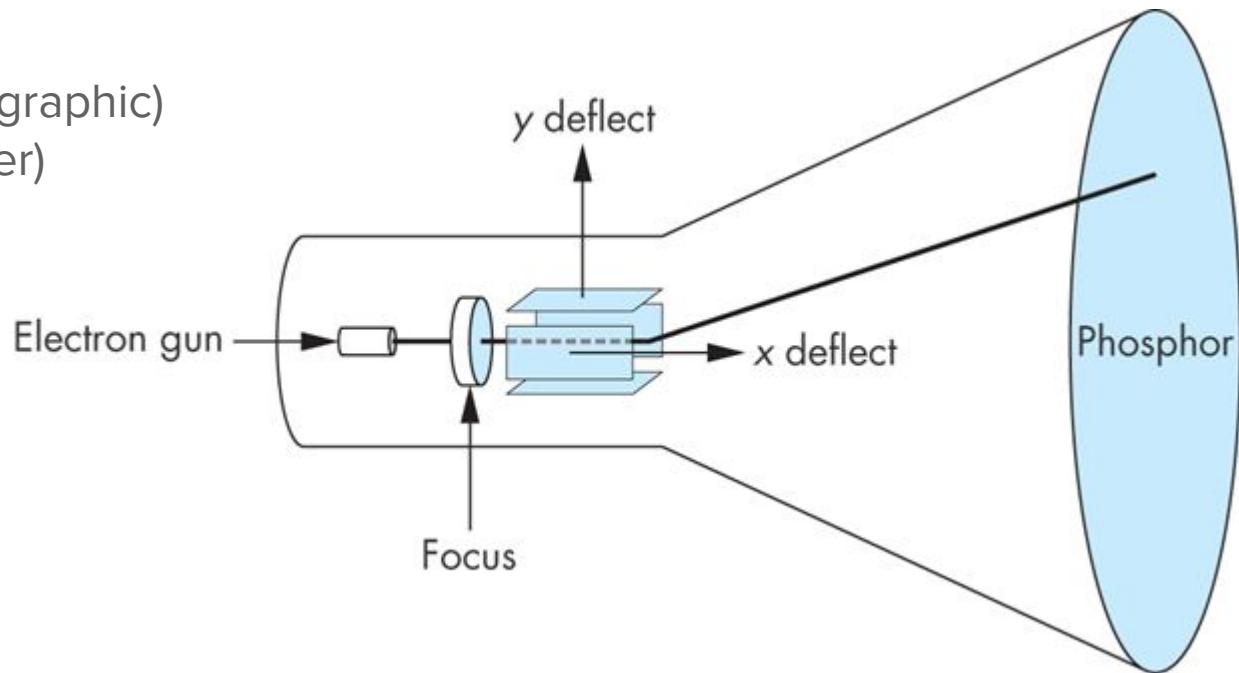
- Simplistic graphics
- Expensive computing equipment



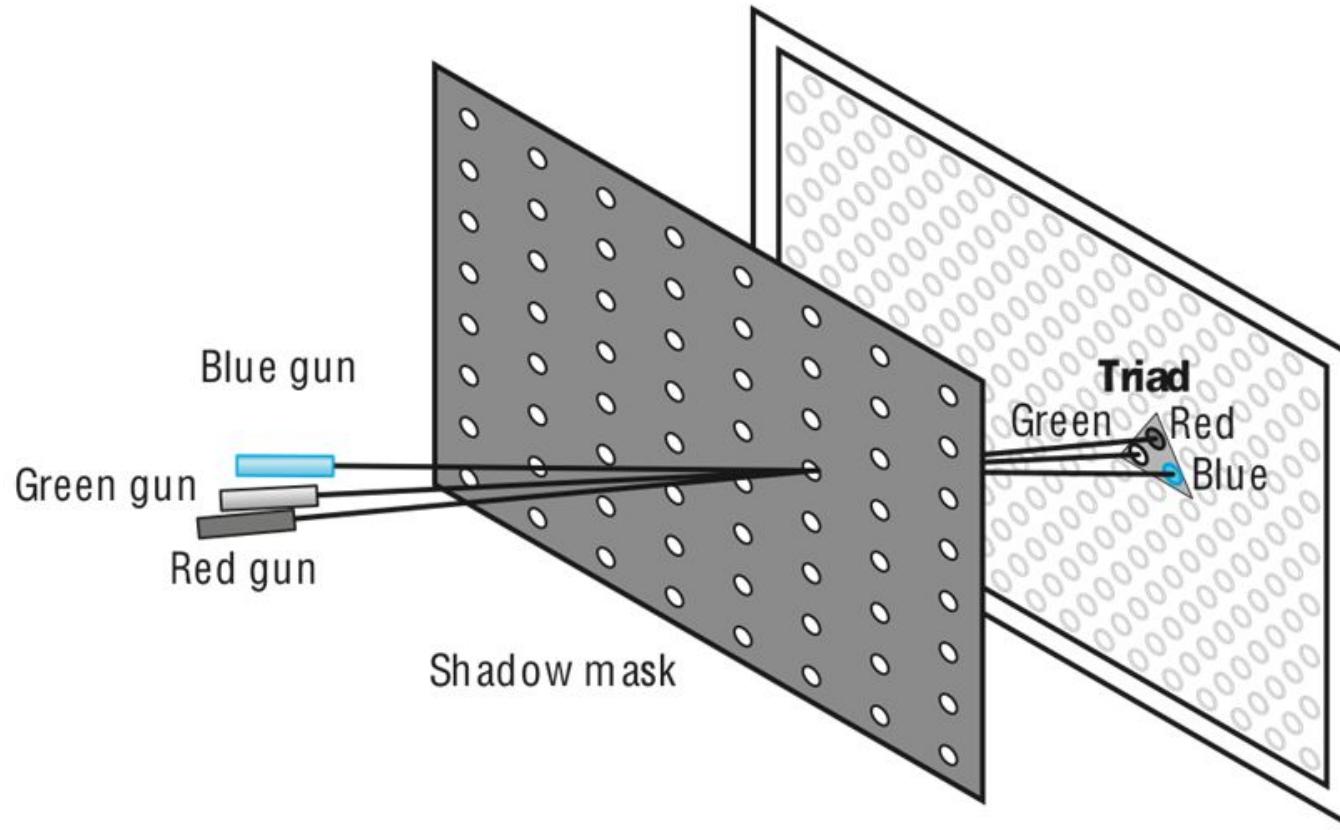
CRT

Modes:

- Line drawing (calligraphic)
- Raster (frame buffer)



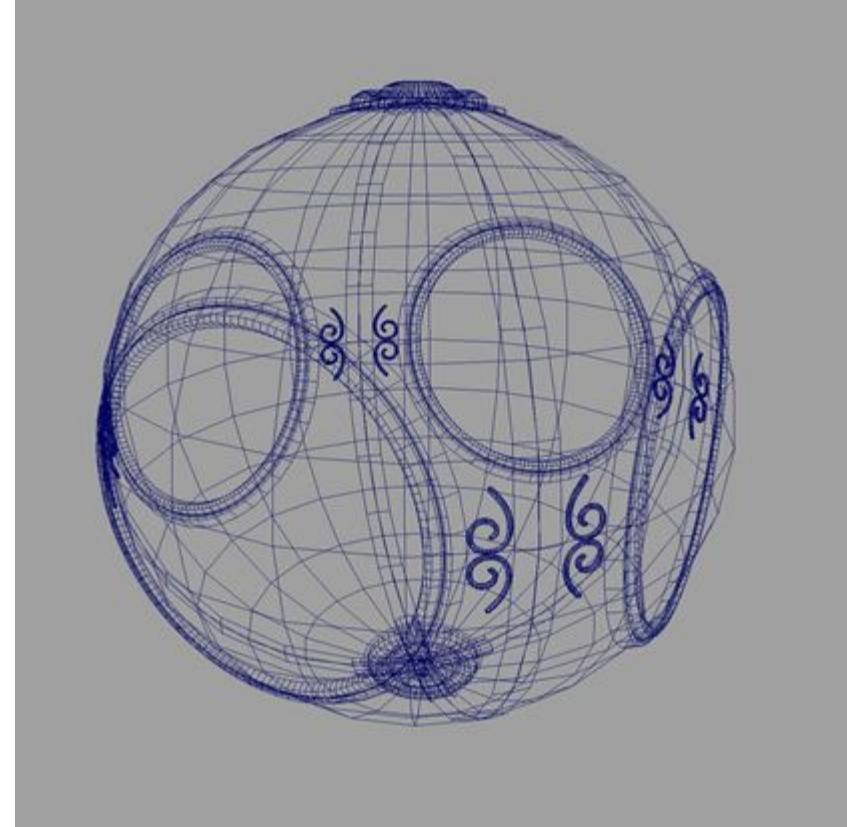
Shadow mask CRT



1960-1970

Wireframe graphics

- Based on display processors and storage tubes



1970-1980

Raster graphics come to the forefront

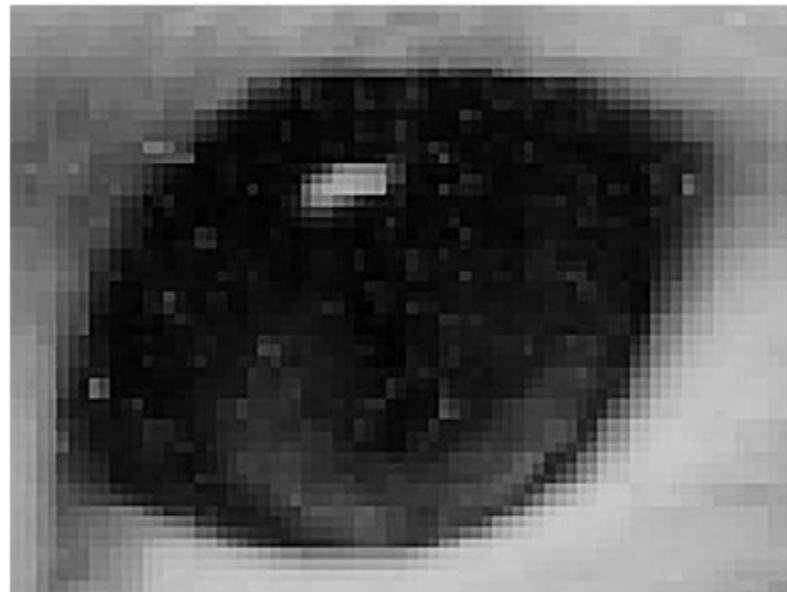
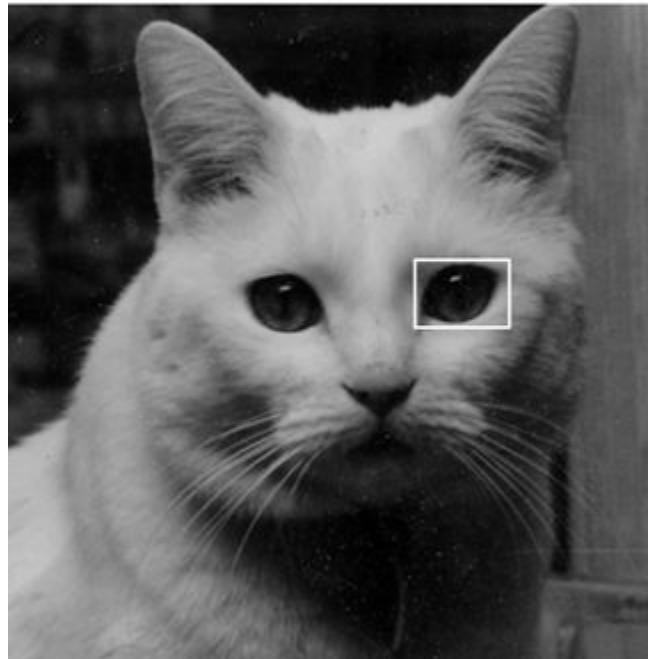
- Standards begin developing
- Workstations/PCs become common



Atari Battlezone (1980)

Raster graphics?

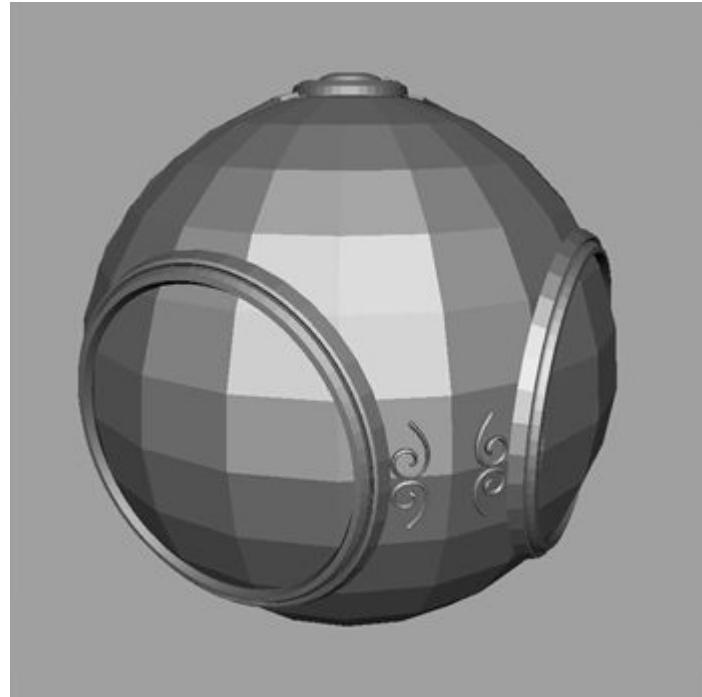
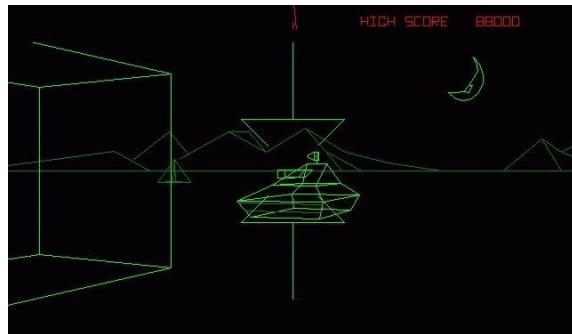
Image is an array (raster) of pixels in frame buffer



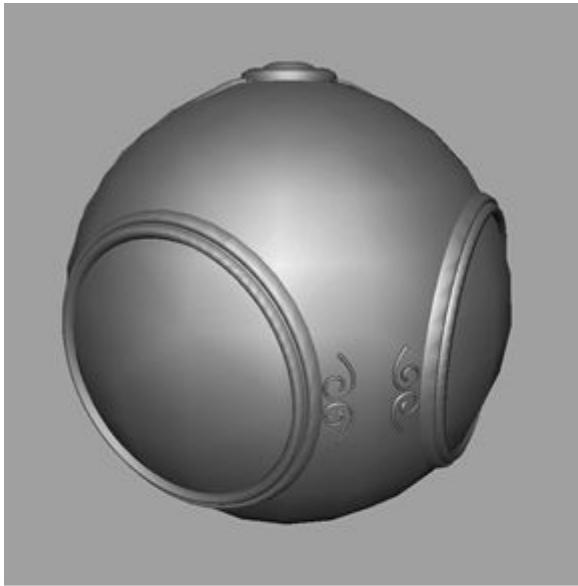
Raster graphics

Takes us from wireframe to filled polygons

- Images are no longer vectors, but pixel arrays displayed in frame buffers



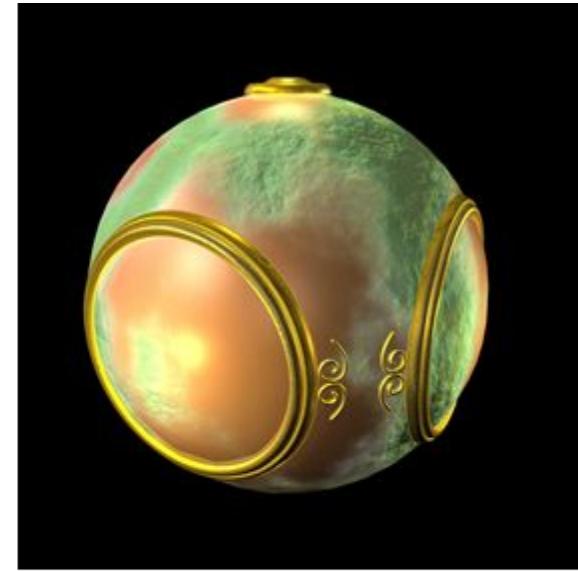
1980-1990



smooth shading



environment
mapping



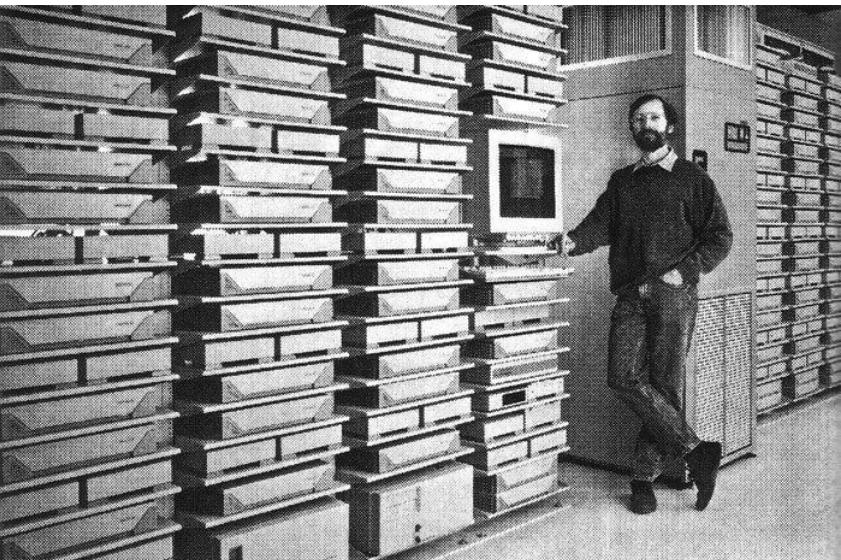
bump mapping

1990-2000

OpenGL

- Texture mapping
- Blending
- etc.

Pixar render farm (1995) →



2000-2010

Photorealism and graphics cards

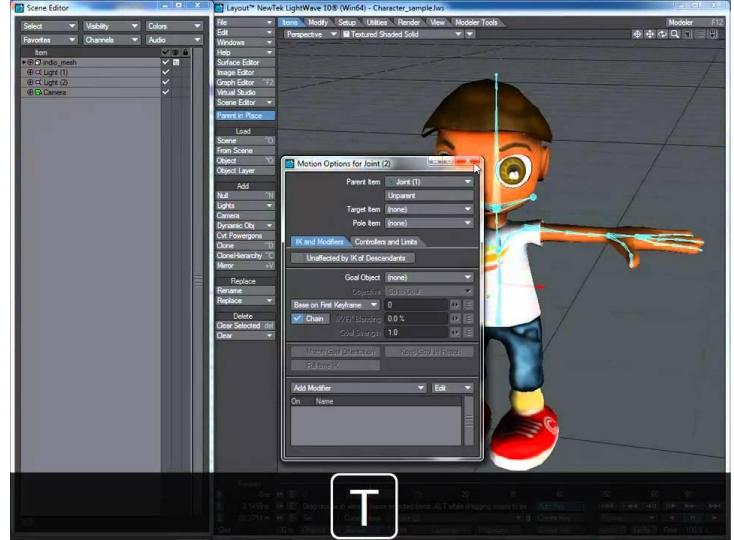
- Mainly Nvidia and ATI

Gaming pushing market even more

Maya / Lightwave common software packages used in movies

Graphical pipelines begin maturing

Newer displays (think flat panels and all variants)



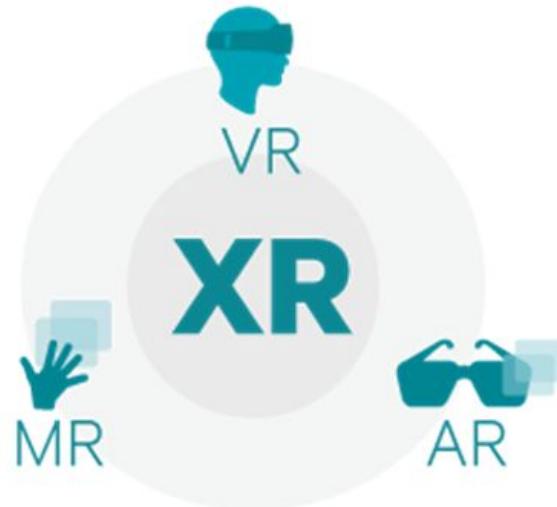
Lightwave character model

2011-?

GPUs dominating market, enabling near real-time rendering

Graphics can be handled by embedded (mobile, Raspberry Pi) to ultra high-def

VR / AR / XR



Andy Serkis talking about advances in graphics:
<https://youtu.be/1qMmPSSN2-Y?t=1049>

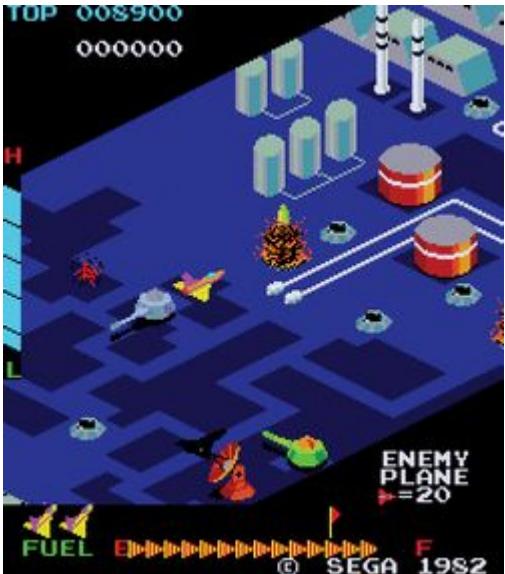
Images

Different viewpoints

2D

3D

2.5D?



Isometric (2D)



Top-down (2D)

FLOOR	SCORE	LIVES	HEALTH	AMMO	
1	0	3		100%	4



Image basics

We need an:

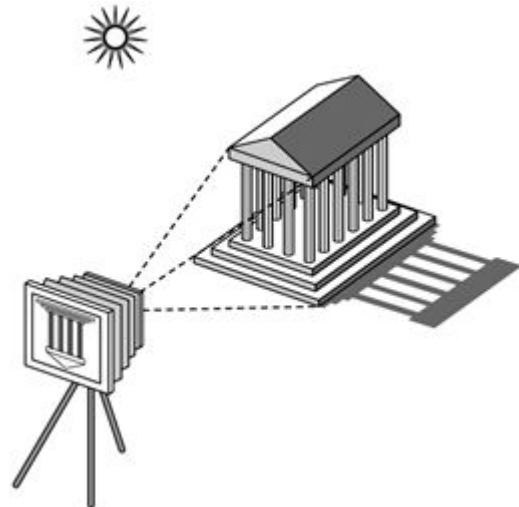
Object

- Some geometric object defined in a space
 - Defined by **vertices**
- Dimensions of your choosing

Viewer

- Eyeball
- Camera
- etc.

Light source

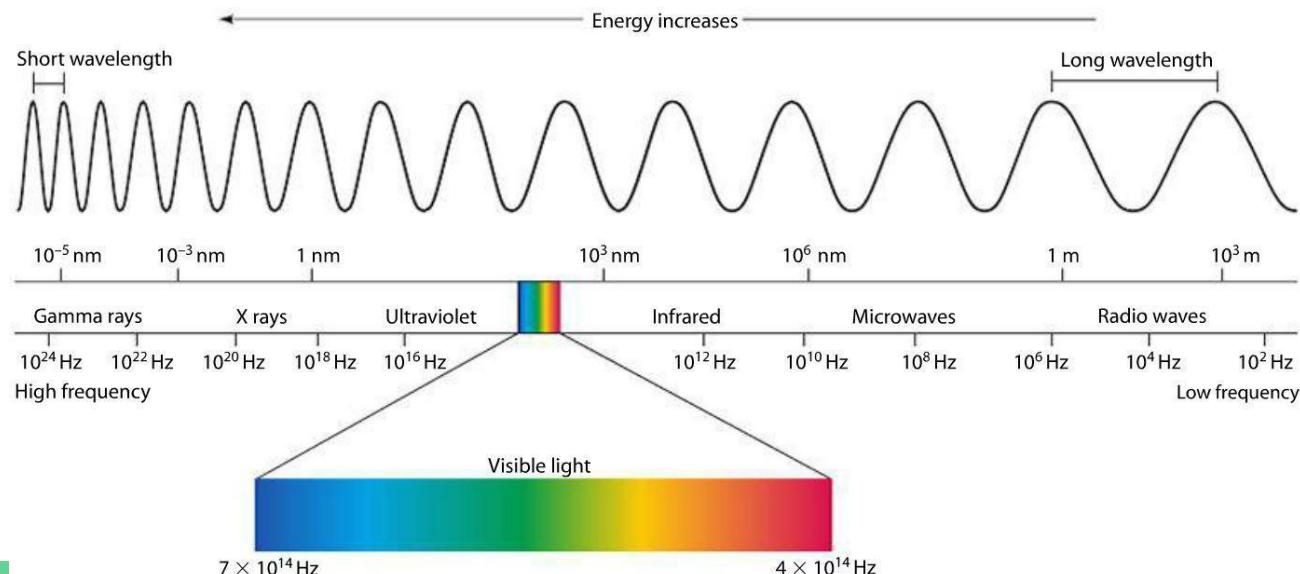


Light

Part of electromagnetic spectrum

- Wavelengths \sim 350-750nm

Longer are reddish, shorter are bluish



Imaging models

Multiple methods for creating images

- Including lighting
- Ray tracing (light rays)
- Radiosity

Ray tracing

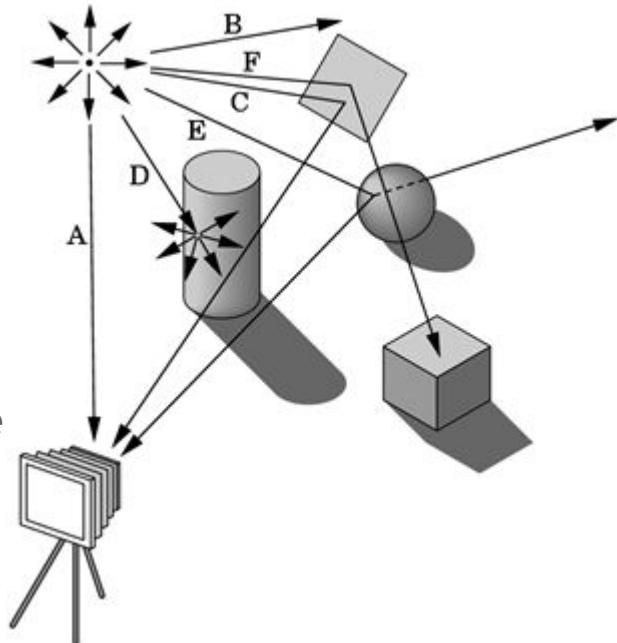
Follow rays of light from a source to destination

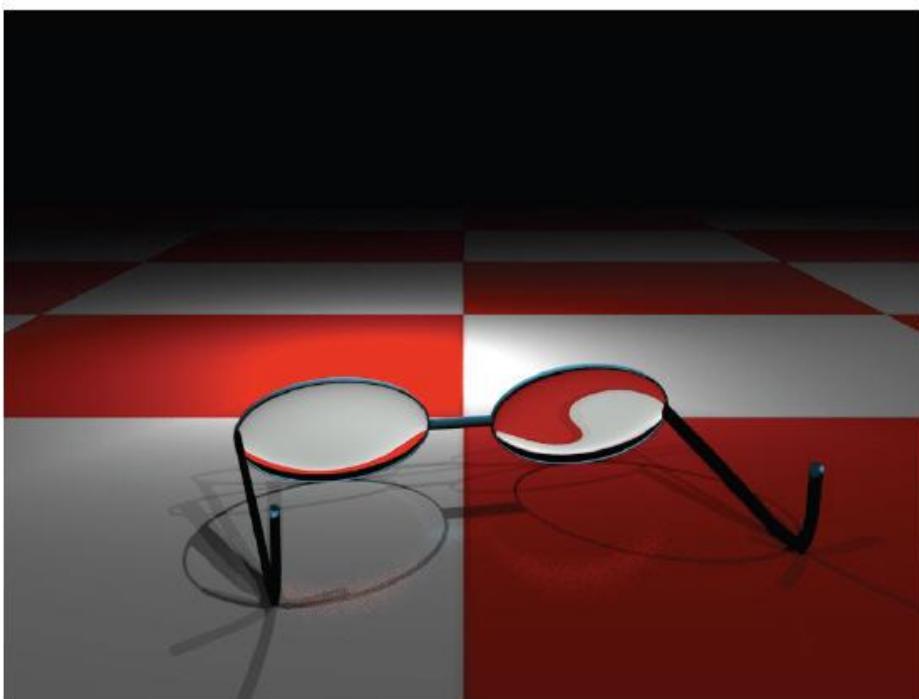
- Figuring out where rays enter camera
- Each ray may interact with scene differently

Some may hit objects

Some may go off into the ether and have **0** effect on the scene

What about mirrors?





Ray Tracing



Ray Tracing w/ Photon Map

Catherine Bendebury and Jonathan Michaels
CS 184 Spring 2005

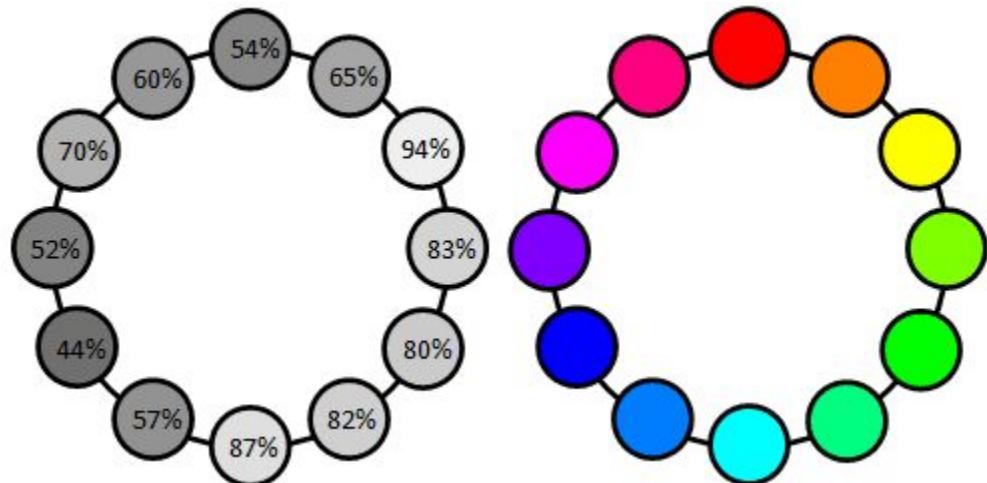
Colors

Luminance

- Different shades of gray
- Monochromatic (range from white to black)

Color

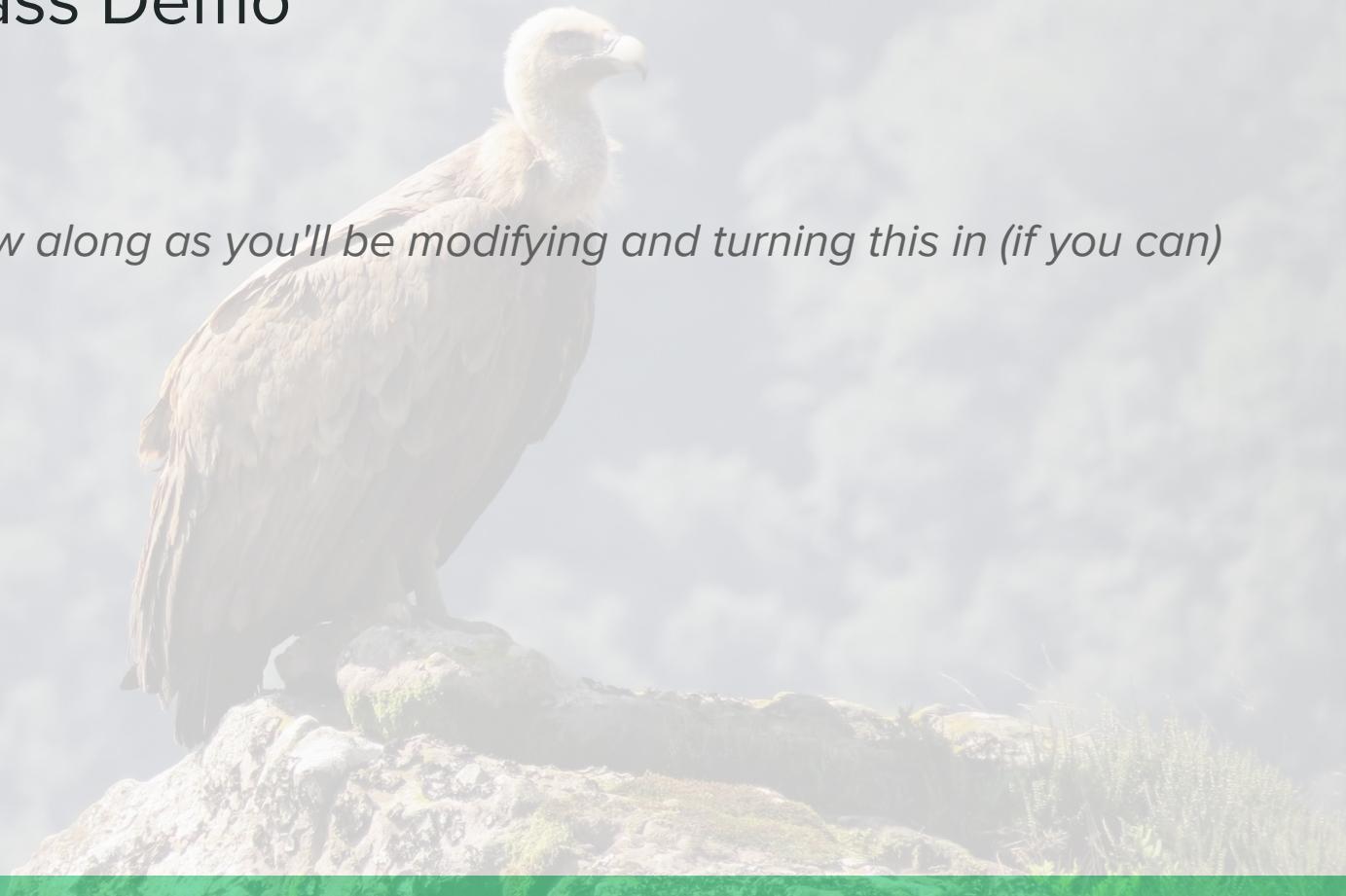
- Hue, saturation, lightness
- Red, green, blue
- etc.



Triangle In-Class Demo

In-class demo!

Hint: you should follow along as you'll be modifying and turning this in (if you can)



Let's draw a triangle (or, the 'other' hello world')

What do we need to get WebGL up and running?

Let's draw a triangle (or, the 'other' hello world)

What do we need to get WebGL up and running?

- **Browser**
- **JavaScript**
- **HTML**

(And to make our lives easier, or more complicated, depending on your perspective)

- **Three.js** – <https://threejs.org/>

Barebones triangle.html

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
</head>
<body>
</body>
</html>
```

Where we'll end up:

<https://efredericks.github.io/gvsu-cis367/demos/hello-world-triangle.html>

1 - three.js

Download three.js and extract. All we need for now is three.min.js!

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
</head>
<body>
  <script src="three.min.js"></script>
  <script>
    // our code will go here (for now)
  </script>
</body>
</html>
```

2 - the scene

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

// camera!
const camera = new THREE.PerspectiveCamera(
    45, window.innerWidth / window.innerHeight, 1, 100);
camera.position.set(0, 0, 10);
camera.lookAt(scene.position);
scene.add(camera);

// renderer!
const renderer = new THREE.WebGLRenderer({antialias: true});
renderer.setClearColor(0x000000, 1);
renderer.setSize(window.innerWidth, window.innerHeight);

// add to dom!
document.body.appendChild(renderer.domElement);
renderer.render(scene, camera);
```

3 - the triangle (under scene.add(...))

```
// setup our triangle (via vertices and faces)
let triangleGeom = new THREE.BufferGeometry();
let points = [];

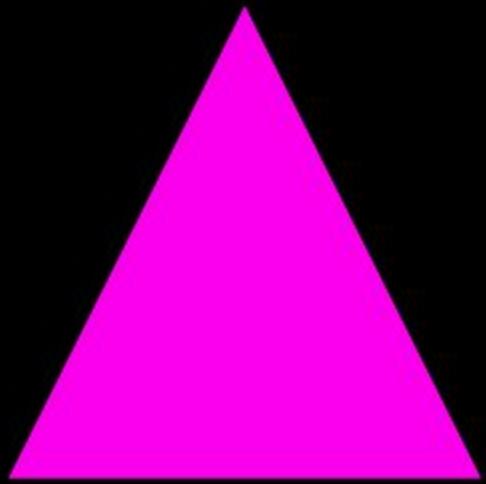
points.push(new THREE.Vector3(0.0, 1.0, 0.0));
points.push(new THREE.Vector3(-1.0, -1.0, 0.0));
points.push(new THREE.Vector3(1.0, -1.0, 0.0));

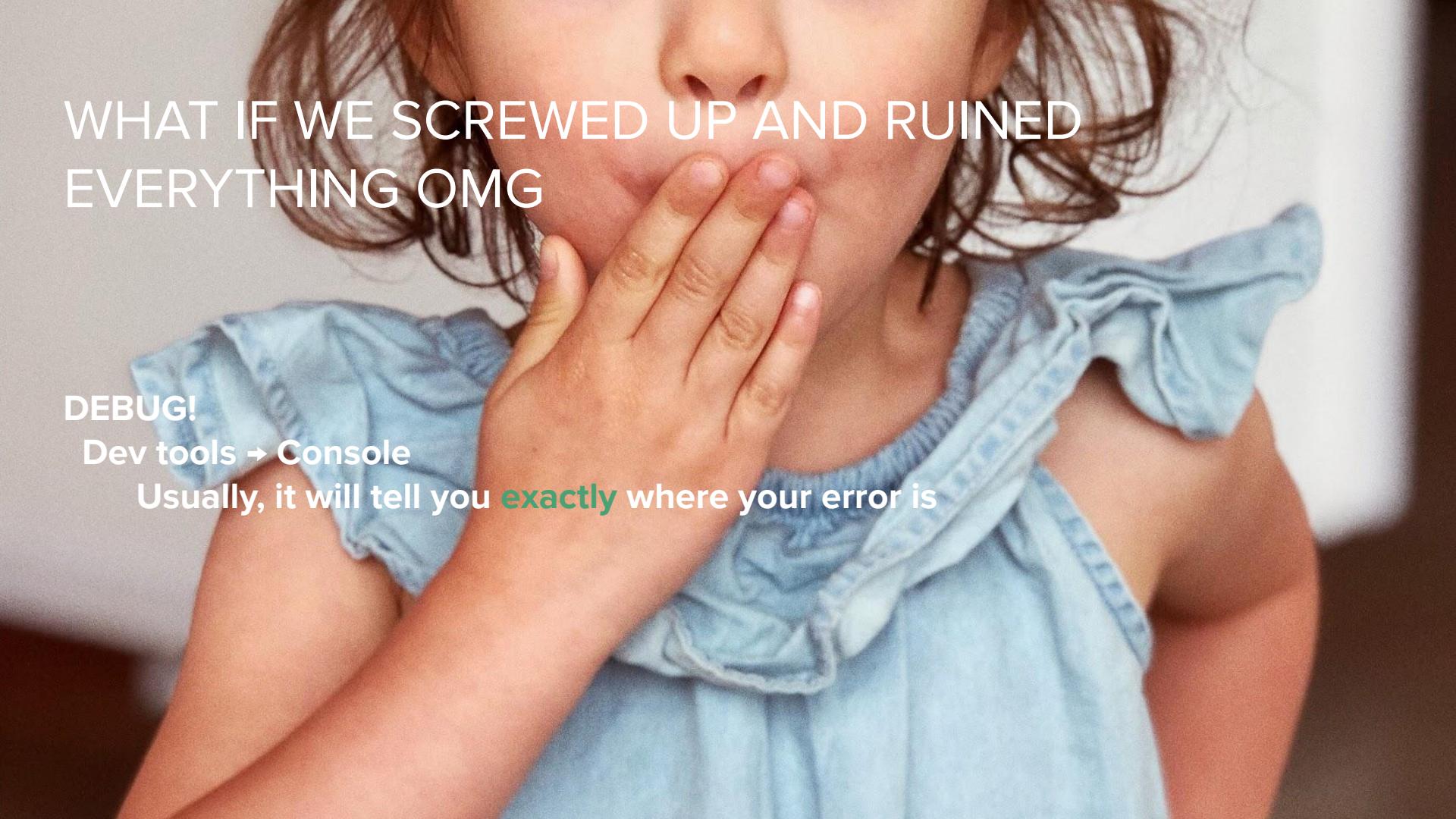
triangleGeom.setFromPoints(points);
triangleGeom.computeVertexNormals();
```

4 - the material (under triangleGeom.computeVertexNormals())

```
// give it a material
let triangleMat = new THREE.MeshBasicMaterial({
  color: 0xFF00FF,
  side: THREE.DoubleSide
});

// and now create the mesh (geom+mat)
let triangleMesh = new THREE.Mesh(triangleGeom, triangleMat);
triangleMesh.position.set(0,0,0); // -1.5, 0.0, 4.0;
scene.add(triangleMesh);
```





WHAT IF WE SCREWED UP AND RUINED
EVERYTHING OMG

DEBUG!

Dev tools → Console

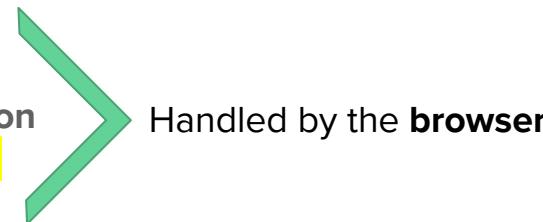
Usually, it will tell you **exactly where your error is**

Let's talk about JavaScript a bit

Web parlance:

- Front end:

- HTML → website **structure**
- CSS → website **prettification**
- JavaScript → website **interaction**



- Backend

- Apache / IIS → serves website **content**
- Scripting languages
 - ASP
 - PHP
 - Python
 - Perl
 - .NET
 - ...



JavaScript

Can be a bit finicky

- Use the browser's developer tools to debug (console)

Is it ... slow?

- For our purposes, the GPU will be handling all the heavy lifting

JavaScript

Few to no native **types**

- Numbers
- Strings
- Booleans

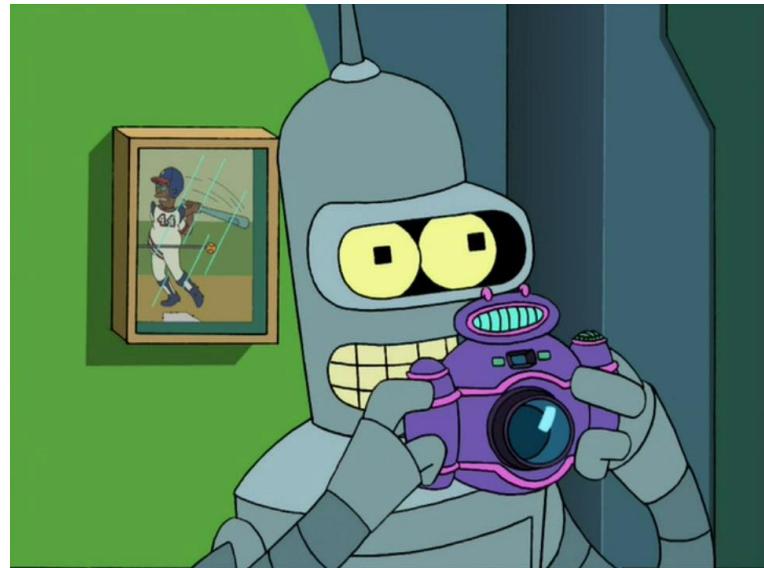
Number → 32 bit float

```
var x = 1
```

```
var y = 1.0
```

And, type can change as you want!
(dynamic typing)

We'll use WebGL's types



Notes

We will just use normal HTML and JS (no TypeScript)

Our graphics may not necessarily be strikingly beautiful
I'm not an artist, sorry

You are not limited though, as long as it is runnable

"In-Class" Assignment #1

- Visit the class GitHub page
 - <https://efredericks.github.io/gvsu-cis367/>
 - Check out some of the examples
- Get the triangle program running on your computer. Change its color. Take a screenshot and submit to Blackboard.

Resources

<https://efredericks.github.io/qvsu-cis367/#resources>

- <https://webglsamples.org/>
- www.opengl.org
- <http://get.webgl.org>
- www.chromeexperiments.com/webgl
- <https://www.shadertoy.com/>