CIS367 - Computer Graphics Introduction

Erik Fredericks - frederer@gvsu.edu


WELCOME TO
HELL
CAYMAN ISLANDS B.W.I.
CLUB INFERNO
HELL

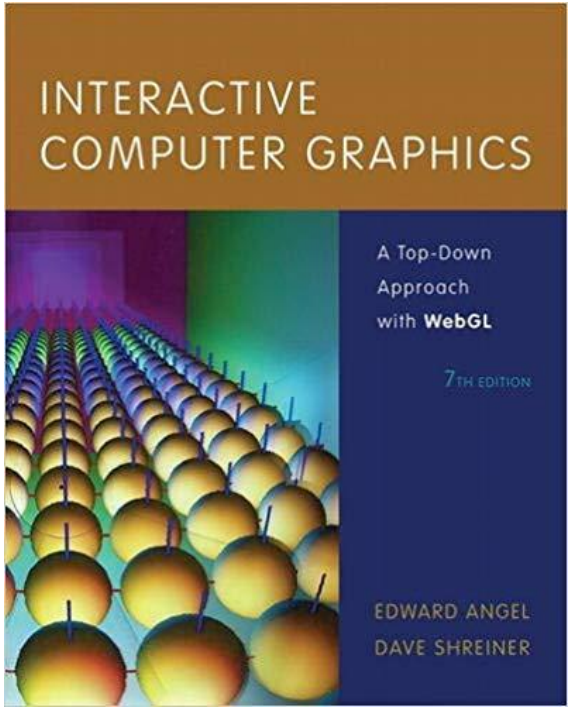
Textbook

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

Is it required?
no

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

Why?
All prior eds use OpenGL, not WebGL

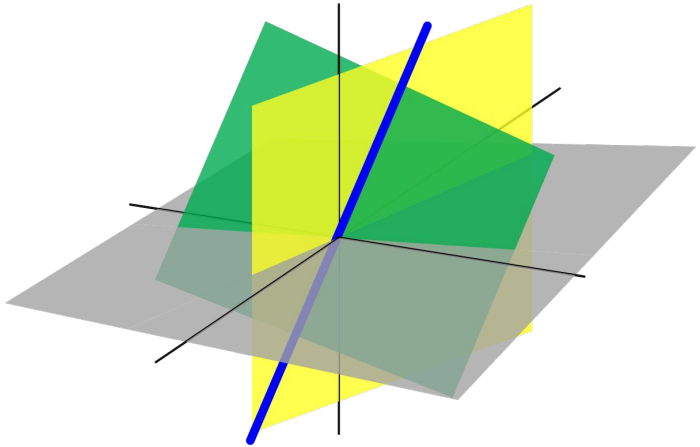


Book website:
https://www.cs.unm.edu/~angel/BOOK/INTERACTIVE_COMPUTER_GRAPHICS/SEVENTH_EDITION/

Syllabus

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 and/or Godot 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?

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* home ions and project **mus**

- You just need t ssions

VPN: https://www.cis.gvsu.ed e-connections/
Web site: https://www.cis nt-web-space/

For now you'll submit your assignments as self-contained ZIP archives to Blackboard

You will also have to make sure they are **publicly-accessible** to me.

This means either:

1) Using the internal GVSU hosting on EOS in /WEB_STUDENT/<your username>
    a) There is a guide in Blackboard for this
2) Using a github.io page
3) Something else you have access to

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 

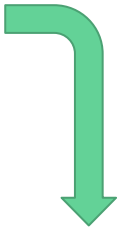
https://webglsamples.org/

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>
```

https://efredericks.github.io/CIS367-ComputerGraphics/demos/hello-world.html

A more fun WebGL program

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

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

https://alexbeutel.com/webgl/voronoi.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?

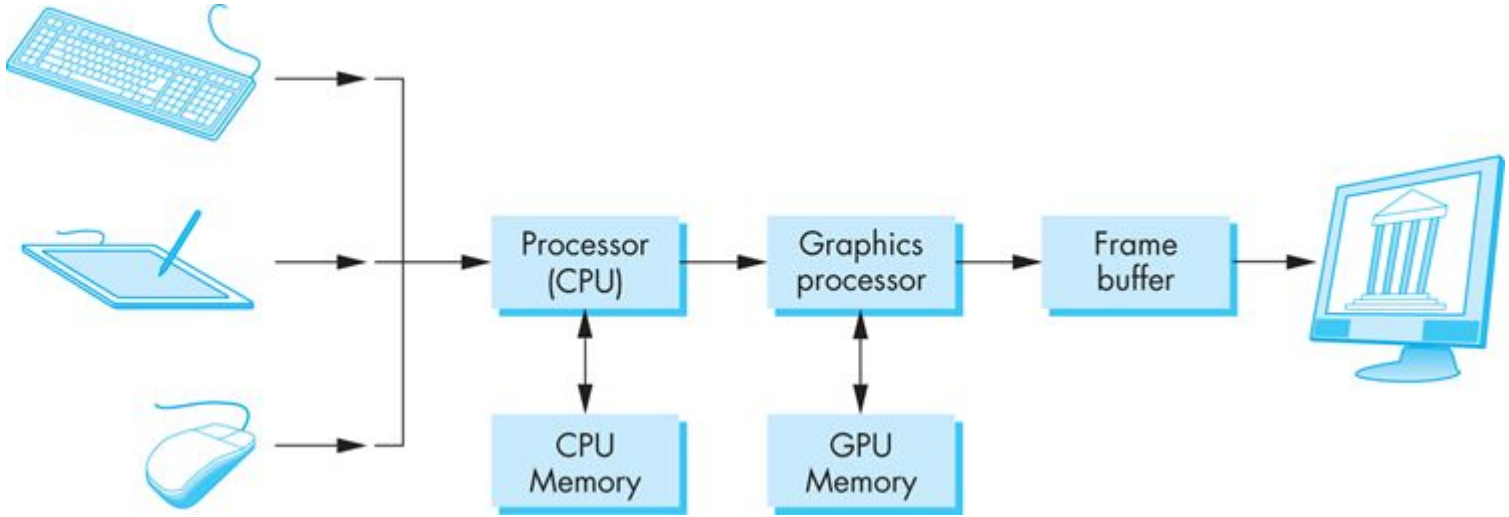
Application:
Artist render of guitar

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

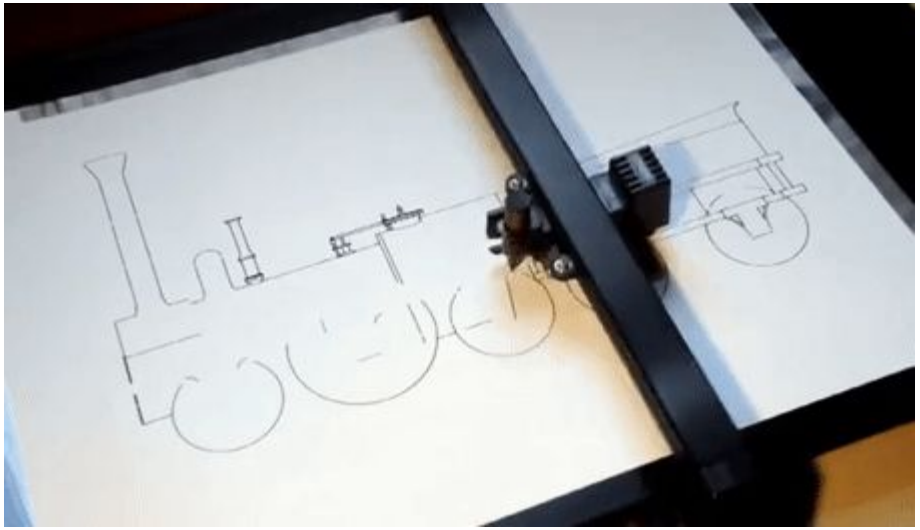
https://www.youtube.com/watch?v=aMli33ornEU

1950-1960

Strip charts
Pen plotters
Simple CRT

Basically:

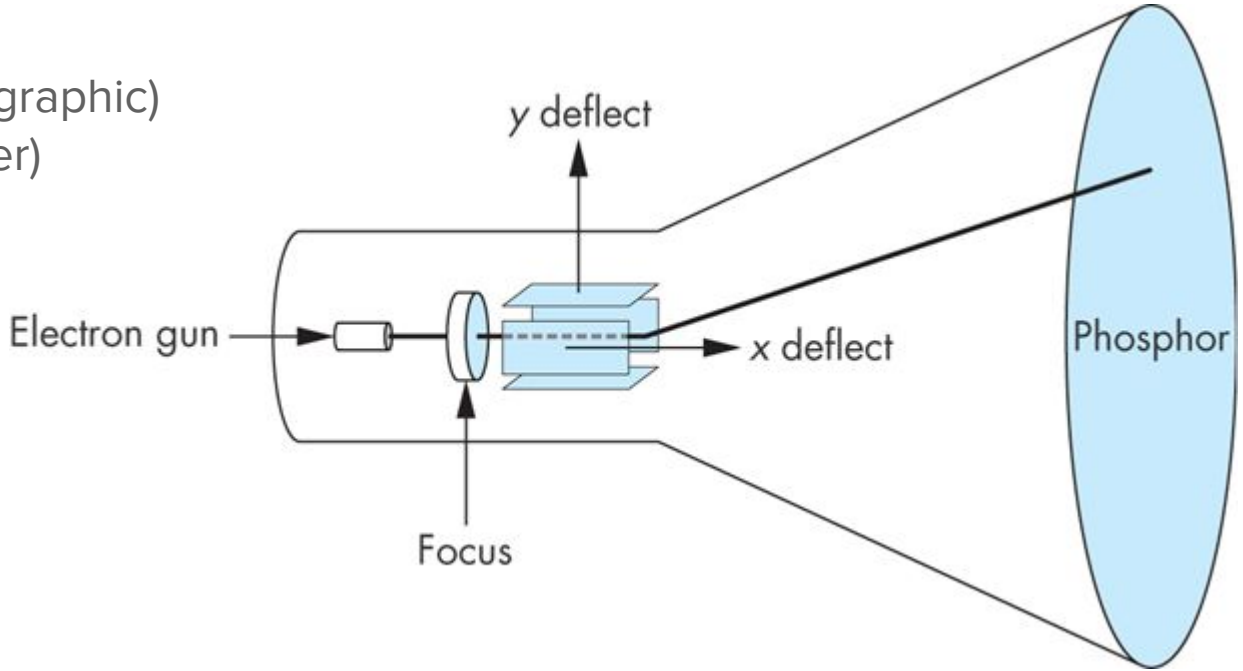
- Simplistic graphics
- Expensive computing equipment



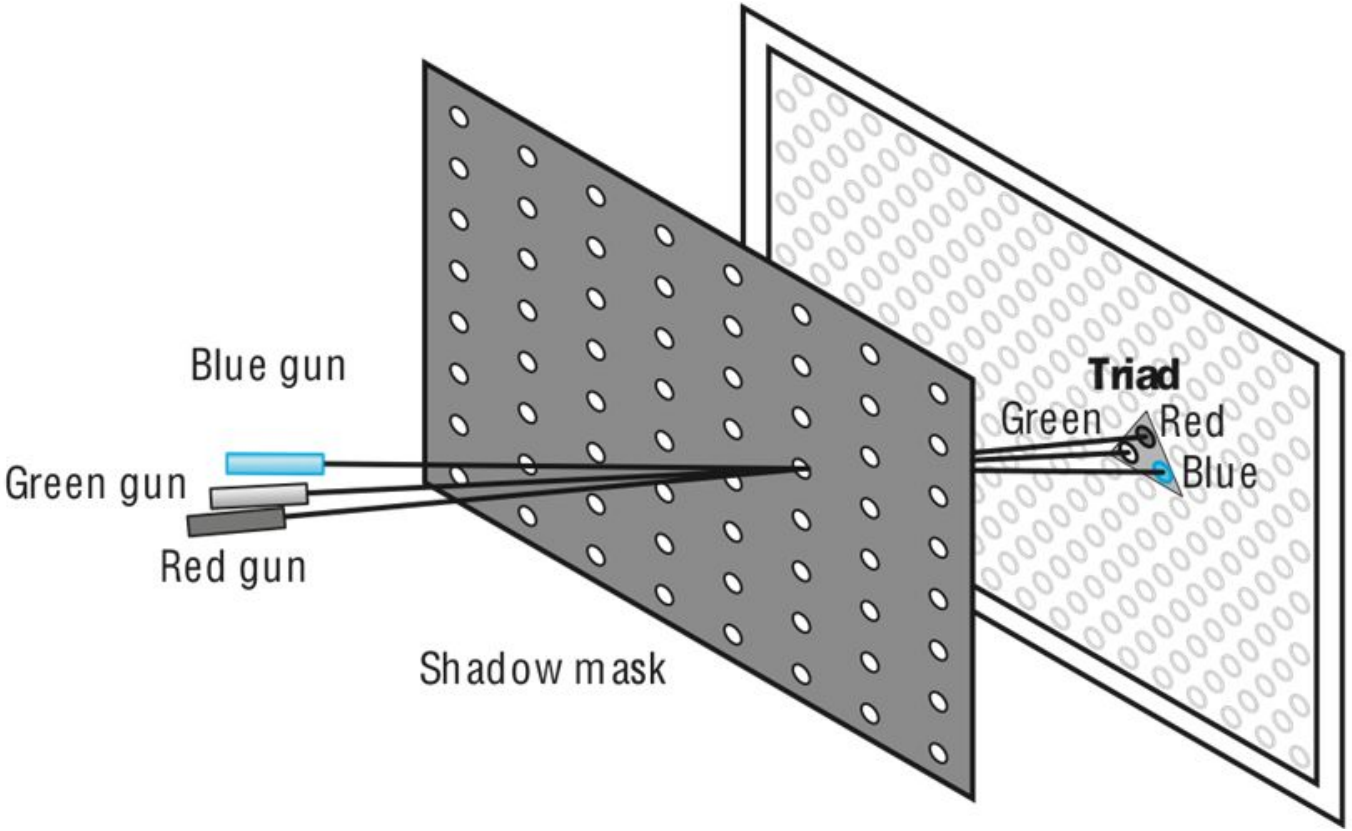
CRT

Modes:

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



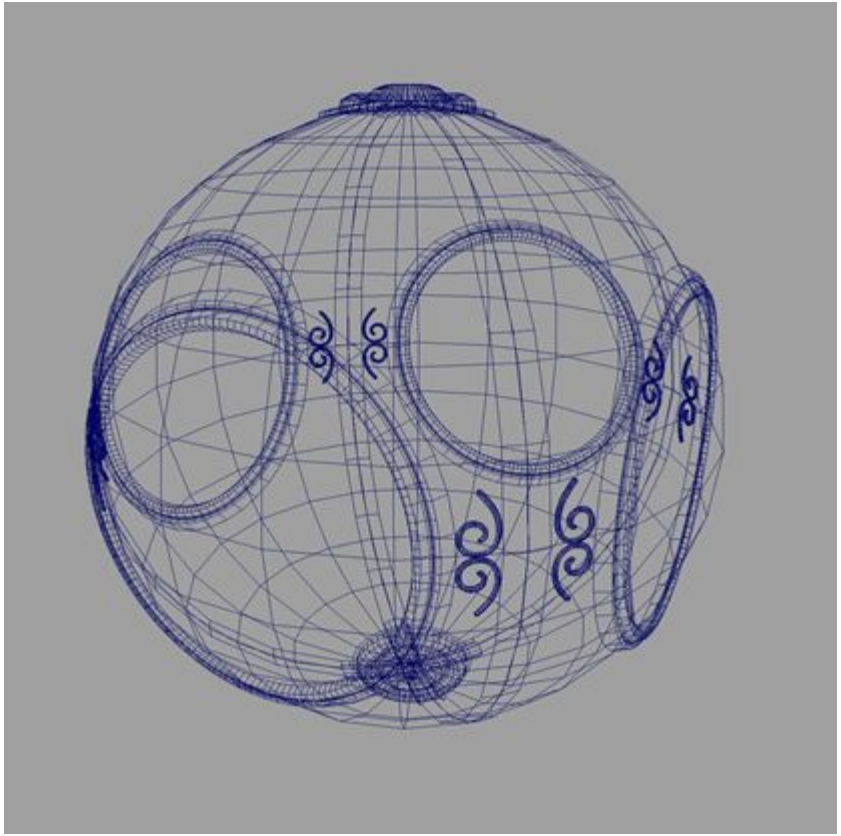
Shadow mask CRT



1960-1970

Wireframe graphics

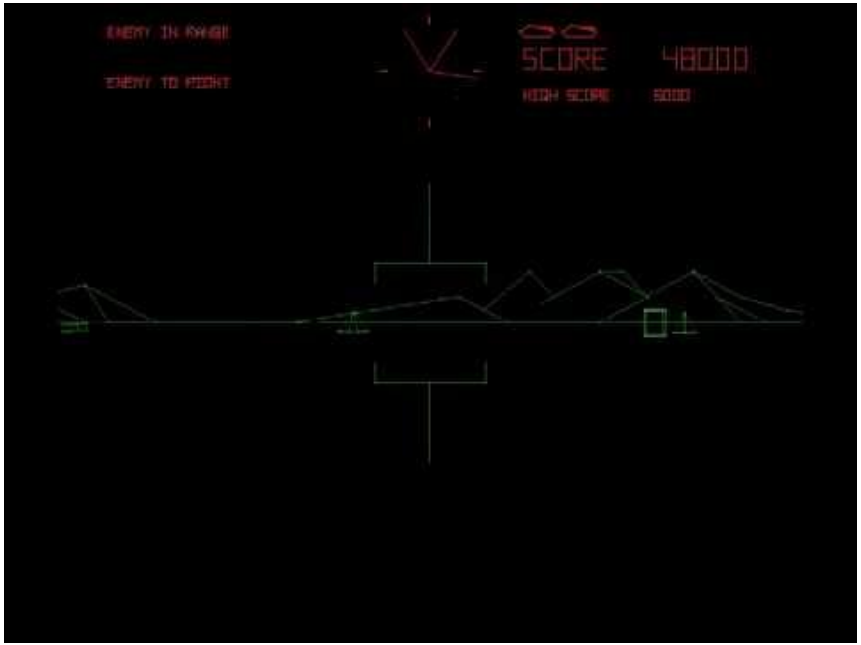
- Based on display processors and storage tubes



1970-1980

Raster graphics come to the forefront

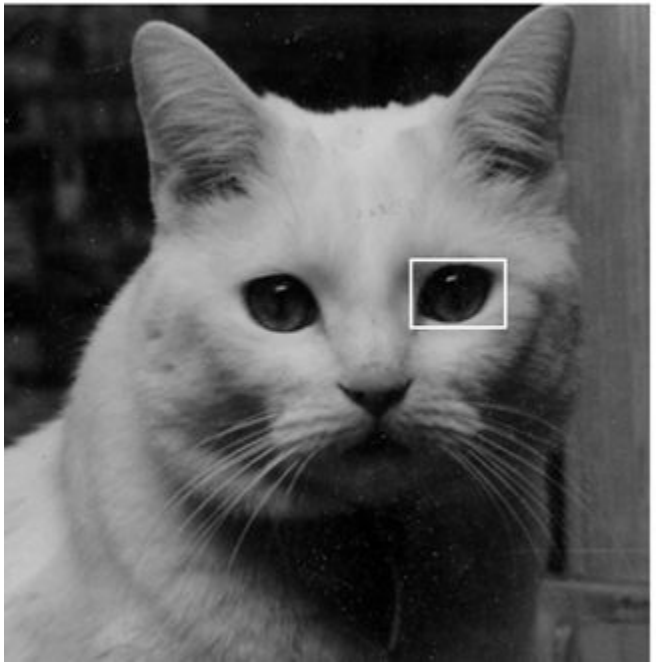
- Standards begin developing
- Workstations/PCs become common

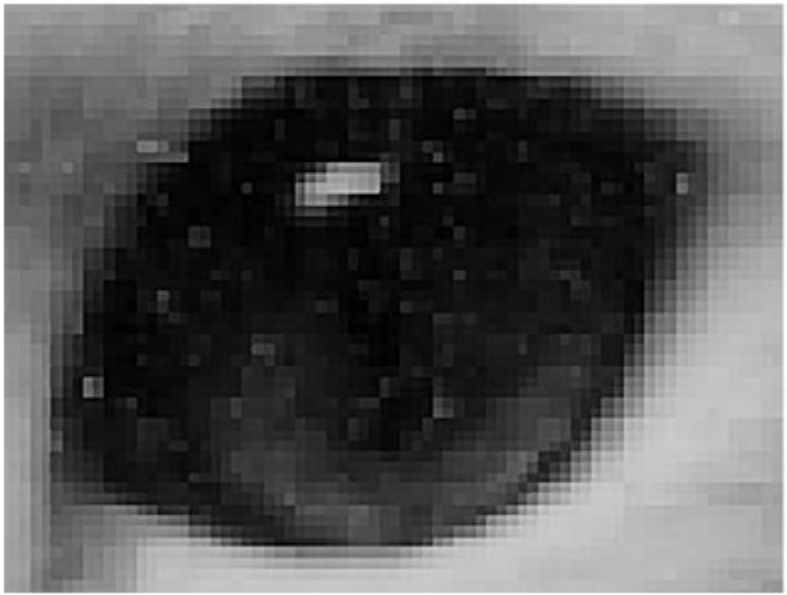


Atari Battlezone (1980)

Raster graphics?

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

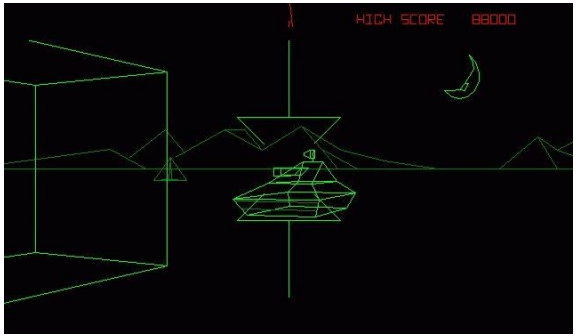


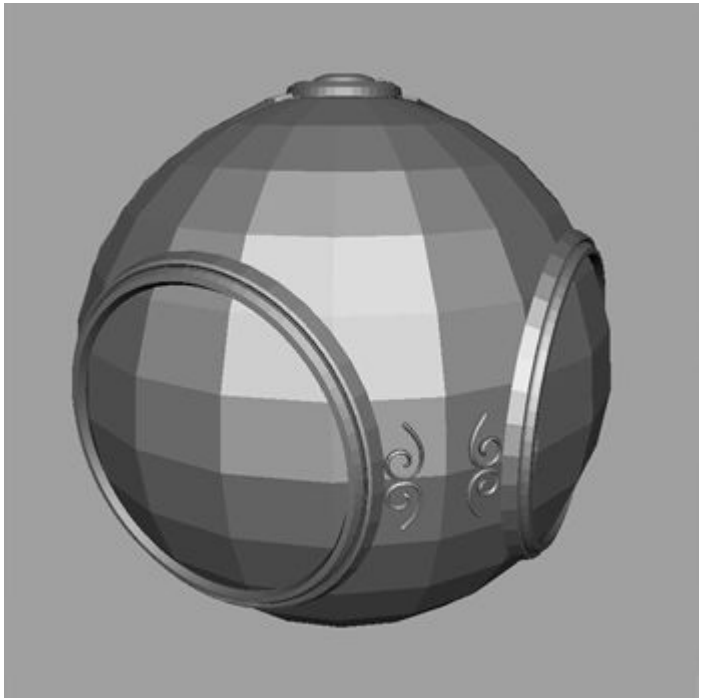


Raster graphics

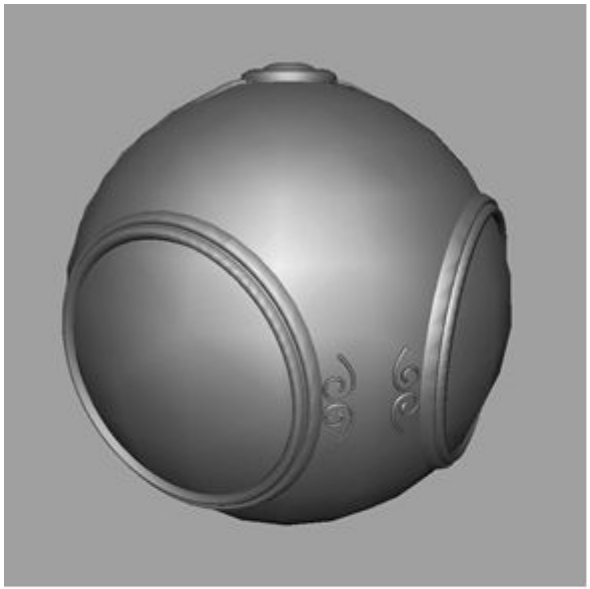
Takes us from wireframe to filled polygons

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



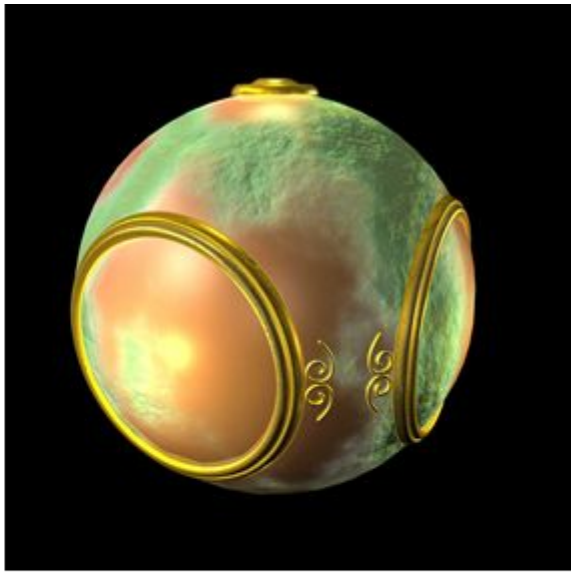


1980-1990



smooth shading

environment mapping



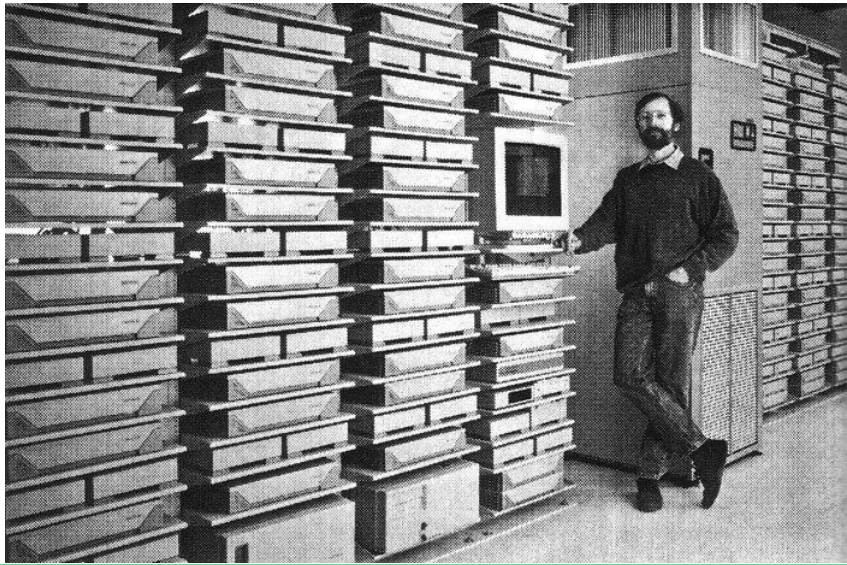
bump mapping

1990-2000

OpenGL

- Texture mapping
- Blending
- etc.

Pixar render farm (1995) ➜



2000-2010

Photorealism and graphics cards

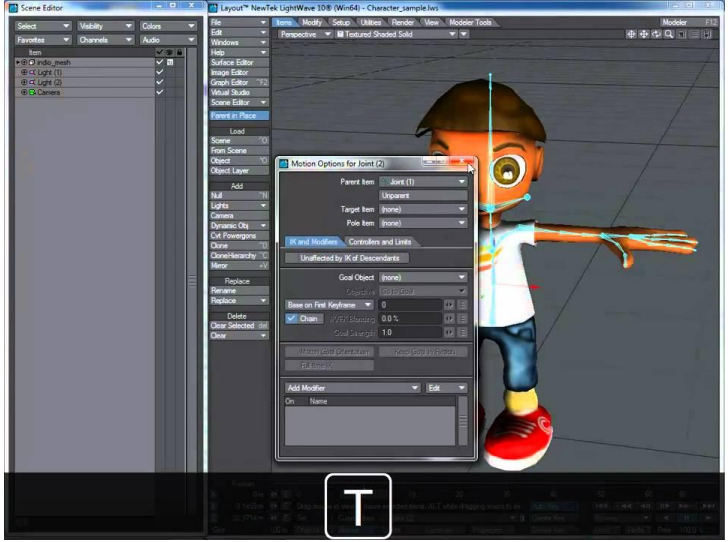
- Mainly Nvidia and ATI

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)



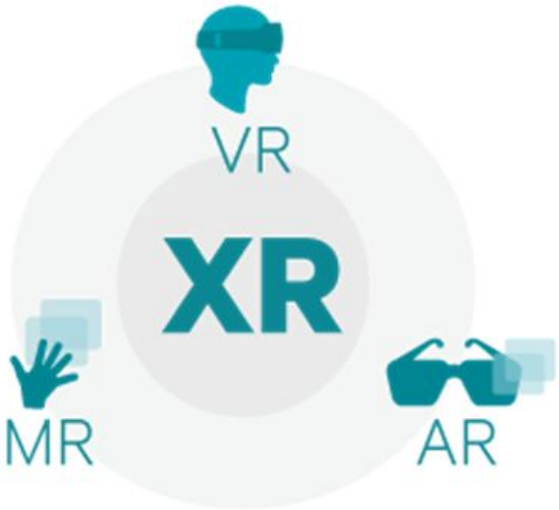
Lightwave character model

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
1
SCORE
0
LIVES
3
HEALTH
100%
AMMO
4

Image basics

We need an:

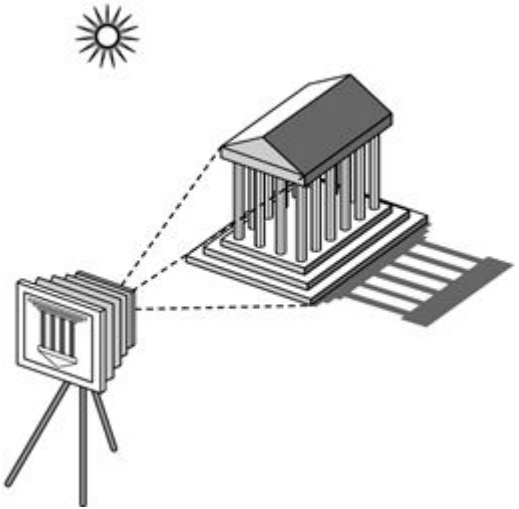
Object

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

Viewer

- Eyeball
- Camera
- etc.

Light source

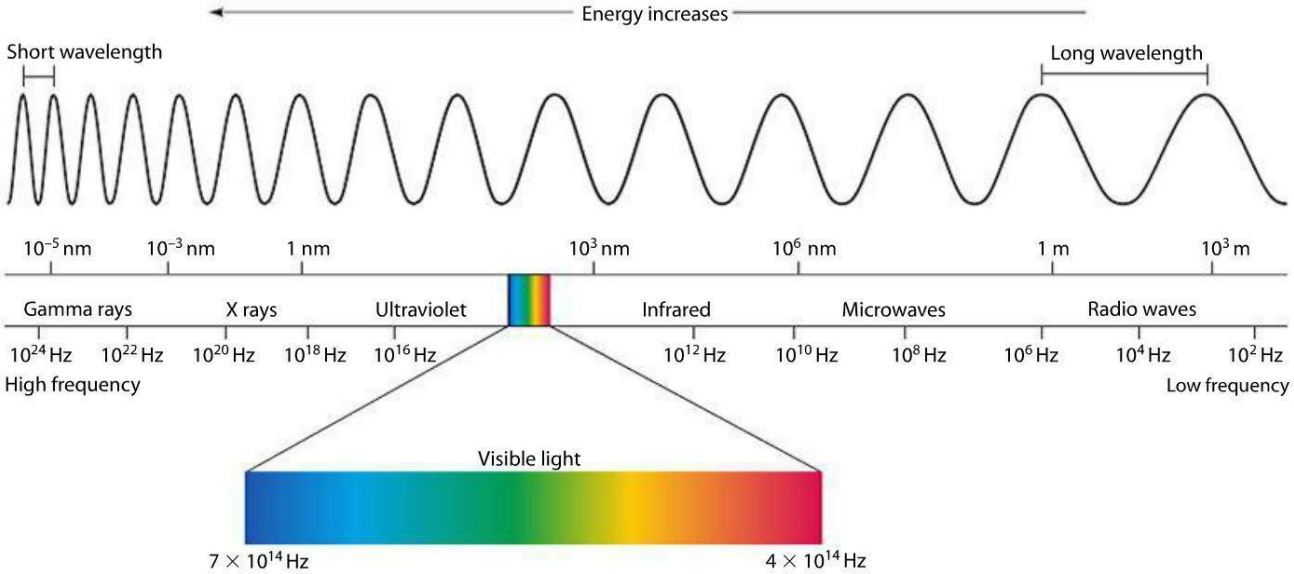


Light

Part of electromagnetic spectrum

- Wavelengths ~350-750nm

Longer are reddish, shorter are bluish



Imaging models

Multiple methods for creating images

- Including lighting

Two examples from Ch 1 (neither of which are super great at real-time, though we're getting there):

- Ray tracing (light rays)
- Radiosity (Ch 12)

Ray tracing

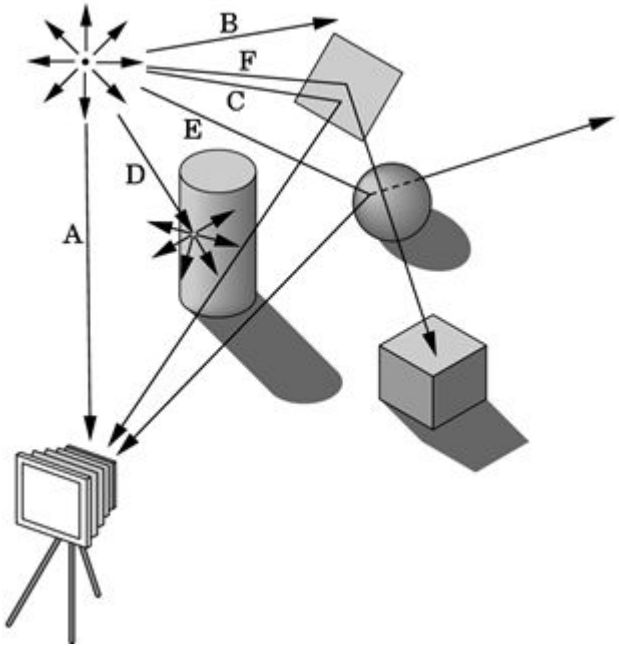
Follow rays of light from a source to destination

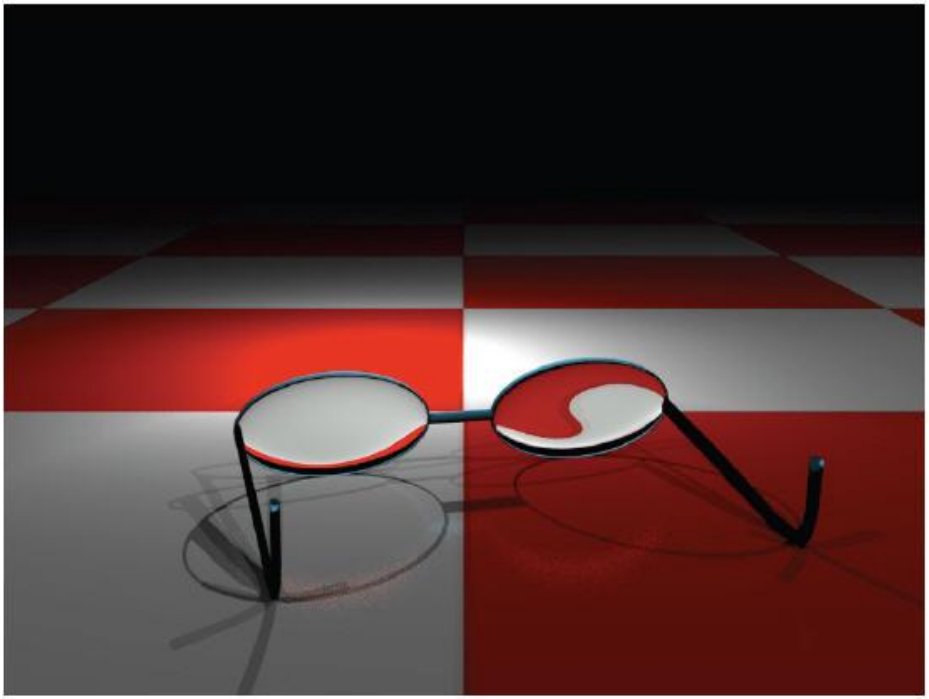
- Figuring out where rays enter camera
- Each ray may interact with scene differently

Some may hit objects

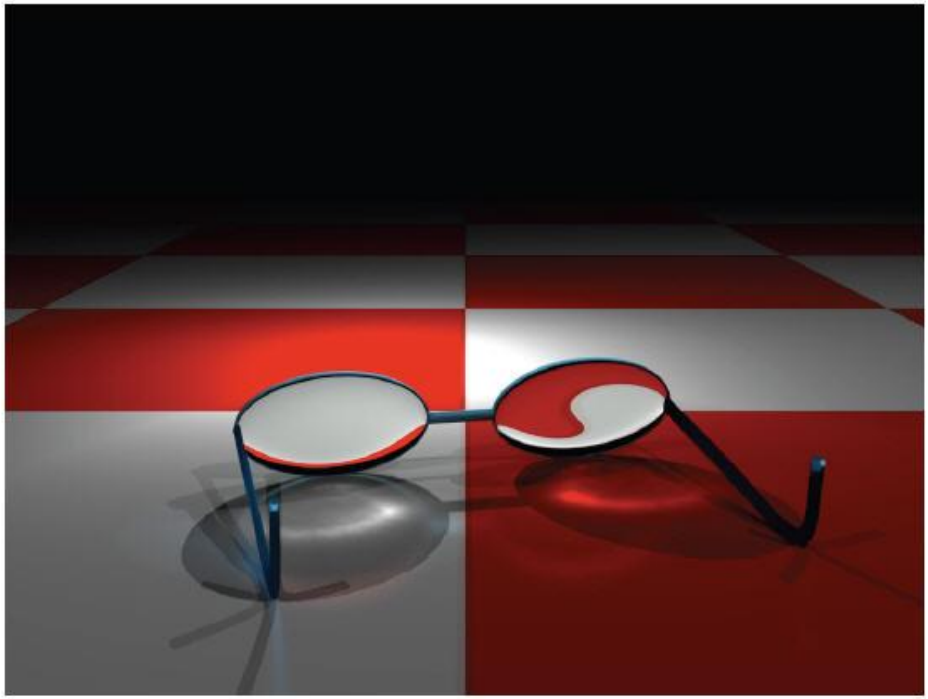
Some may go off into the ether and have **0** effect on the scene

What about mirrors?





Ray Tracing



Ray Tracing w/ Photon Map

Catherine Bendebury and Jonathan Michaels
CS 184 Spring 2005

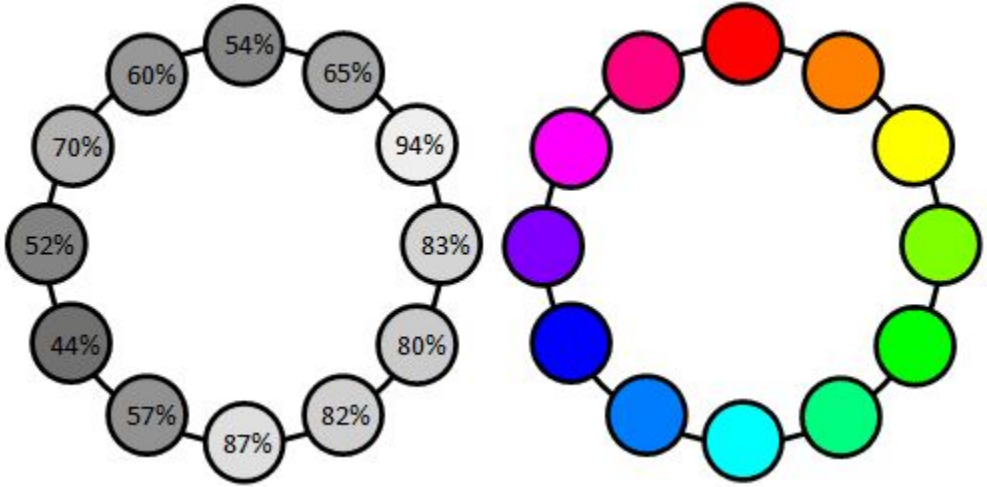
Colors

Luminance

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

Color

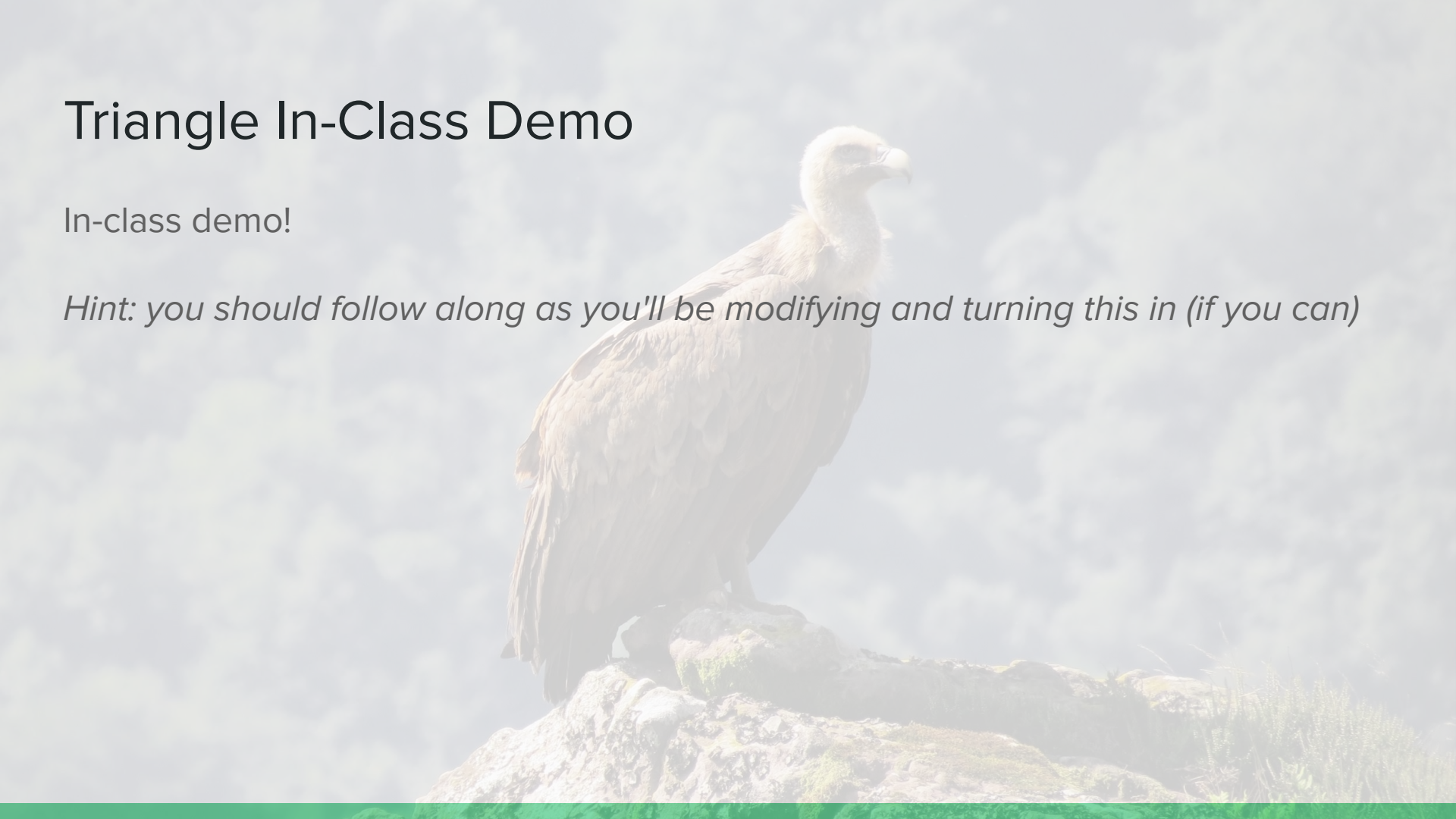
- Hue, saturation, lightness
- Red, green, blue
- etc.



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**

Barebones triangle.html

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

triangle.html

```
<head>
  ...
  <!-- define our shaders! -->
  <script id="vertex-shader" type="x-shader/x-vertex">
    attribute vec4 vPosition;
    void main() {
      gl_Position = vPosition;
    }
  </script>

  <script id="fragment-shader" type="x-shader/x-fragment">
    precision mediump float;
    void main() {
      gl_FragColor = vec4( 1.0, 0.0, 0.0, 1.0);
    }
  </script>
</head>
```

triangle.html

```
  ...
  <!-- include our libraries for WebGL -->
  <script src="../Common/webgl-utils.js"></script>
  <script src="../Common/initShaders.js"></script>
  <script src="../Common/MV.js"></script>

  <!-- the ACTUAL triangle code!! -->
  <script src="triangle.js"></script>
</head>
```

This assumes you have downloaded the GitHub repo! Also note the paths, it assumes that you have to go up a directory to get to Common

triangle.html (or, the end of the file)

```
<body>
  <canvas id="gl-canvas" width="640" height="480">
    Your browser doesn't support HTML5!
  </canvas>
</body>
```

And now, triangle.js

Save it to the same location as triangle.html

```
var gl;
var points;

window.onload = function init() {
};

function render() {
}
```

triangle.js

```
window.onload = function init() {
  // Setup our canvas and WebGL
  var canvas = document.getElementById('gl-canvas');
  gl = WebGLUtils.setupWebGL(canvas);
  if (!gl) { alert('WebGL unavailable'); }

  // Triangle vertices
  var vertices = [
    vec2(-1, -1),
    vec2(0, 1),
    vec2(1, -1)
  ];
};
```

triangle.js

```
...

  // configure WebGL
  gl.viewport(0, 0, canvas.width, canvas.height);
  gl.clearColor(1.0, 1.0, 1.0, 1.0);

  // load and initialize shaders
  var program = initShaders(gl, 'vertex-shader', 'fragment-shader');
  gl.useProgram(program);

};
```

triangle.js

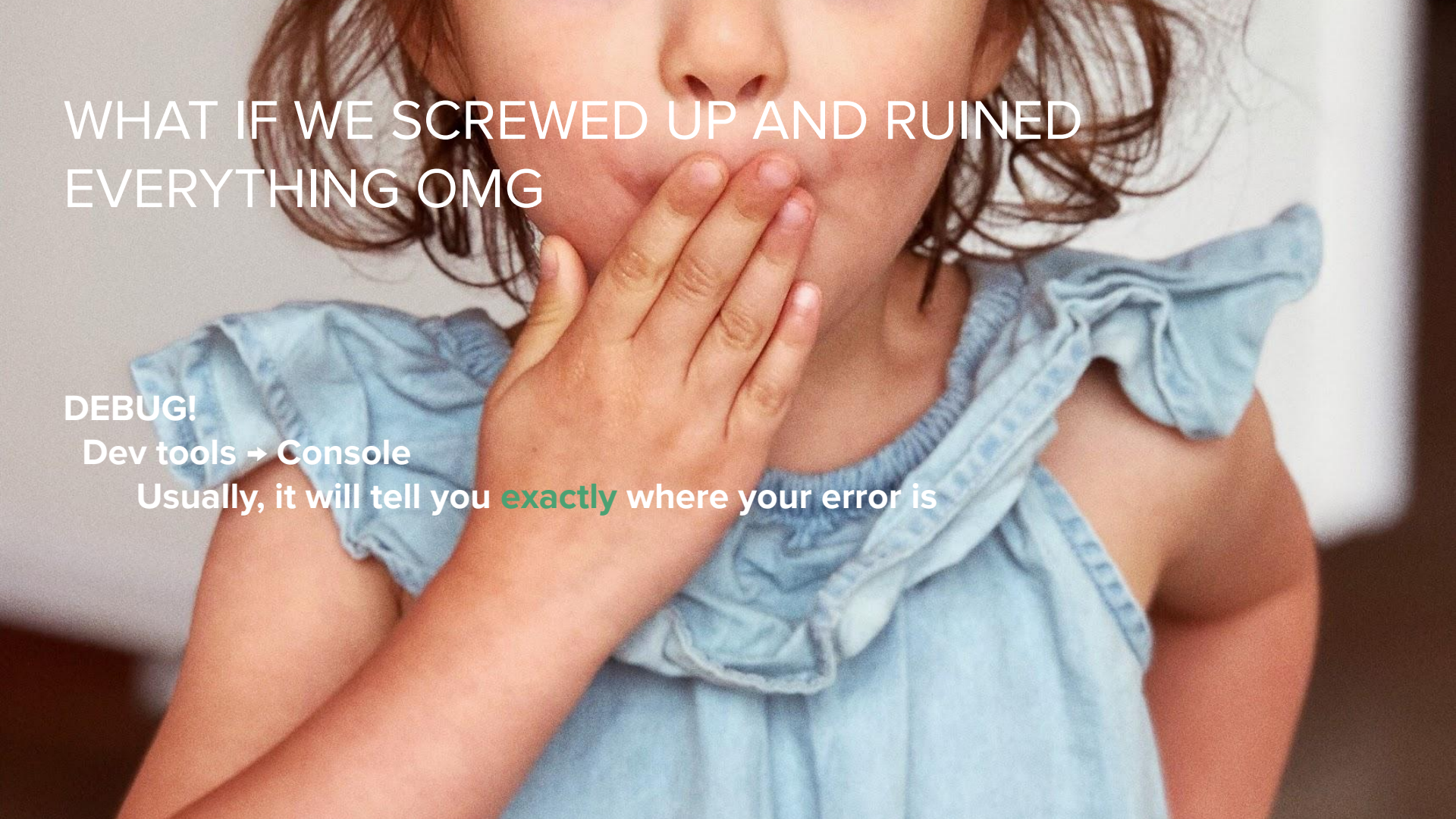
```
...

  // load data into GPU
  var bufferID = gl.createBuffer();
  gl.bindBuffer(gl.ARRAY_BUFFER, bufferID);
  gl.bufferData(gl.ARRAY_BUFFER, flatten(vertices), gl.STATIC_DRAW);

  // set position and render
  var vPosition = gl.getAttribLocation(program, 'vPosition');
  gl.vertexAttribPointer(vPosition, 2, gl.FLOAT, false, 0, 0);
  gl.enableVertexAttribArray(vPosition);
  render();
};
```

triangle.js

```
function render() {
  gl.clear(gl.COLOR_BUFFER_BIT);
  gl.drawArrays(gl.TRIANGLES, 0, 3);
}
```


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**

Handled by the **browser**

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

Handled by the **server**

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**

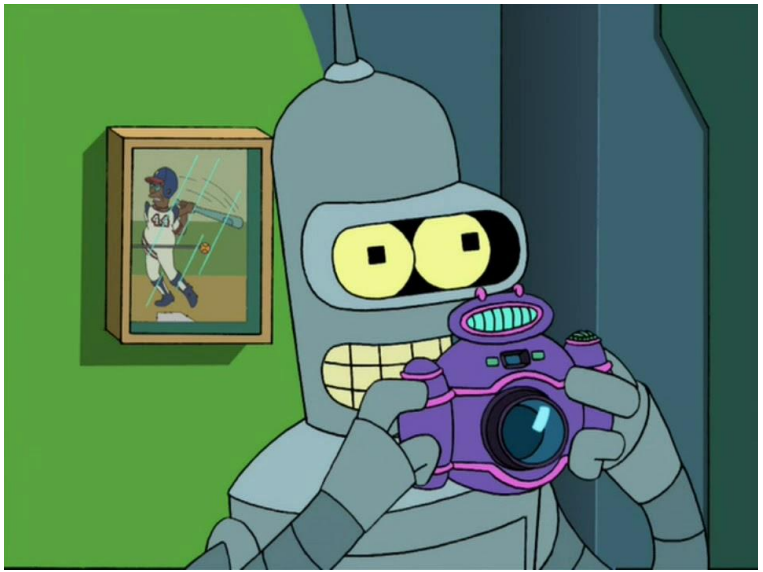
- Numbers
- Strings
- Booleans

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/demos/
  - Check out some of the examples

- Get the triangle program running on your computer. Change its color. Take a screenshot and submit to Blackboard.
  - (You can clone the repo or simply view it and check the source)
  - (Don't forget to grab the Angel libraries!)

Resources

Will create a living Blackboard list as well, but to get you started:

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

https://www.shadertoy.com/

WHOOO. ARE. YOUUUU?

Let's introduce ourselves

- Name
- Rank
- Serial number

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