

# Intro to Graphics Programming

## Using Apple's Metal API

Chris Feher

Shopify Waterloo

August 30, 2017

# What Is Metal

Similar to OpenGL. Metal gives you near direct access to the GPU on your iOS, TVOS, or MacOS device.

Useful for making games as well as performing complex calculations



# Why Do We Use It?

We needed a safe, reliable, and performant way to draw thousands of basic shapes on the screen every frame.

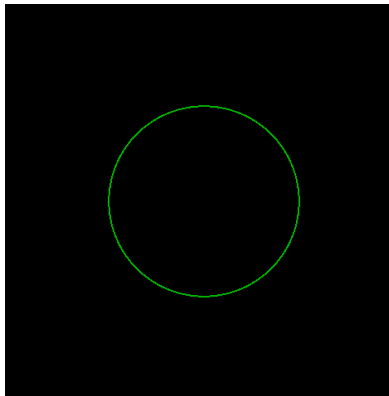
We ran into issues with SpriteKit so we decided to explore Metal



# What Are We Doing Here?

We will be working through the process of drawing a shockwave as seen on the lazerbeam map

This includes drawing shapes and animating them



# Code

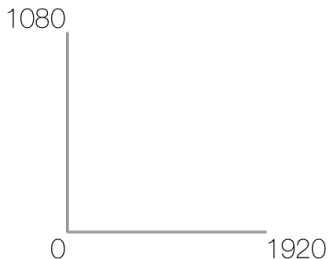
The code is separated into sections that correspond to each question. If there is no "start" folder, you must continue with the "finish" of the previous section

## Repository

<https://github.com/Shopify/metal-workshop>

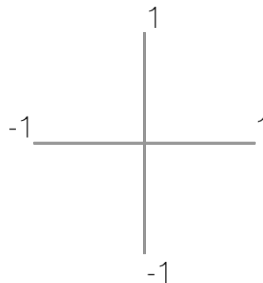
# Coordinate Systems

## Window Coordinate System



*Note: Window coordinate system in Metal is flipped on Y axis*

## Clip Coordinate System



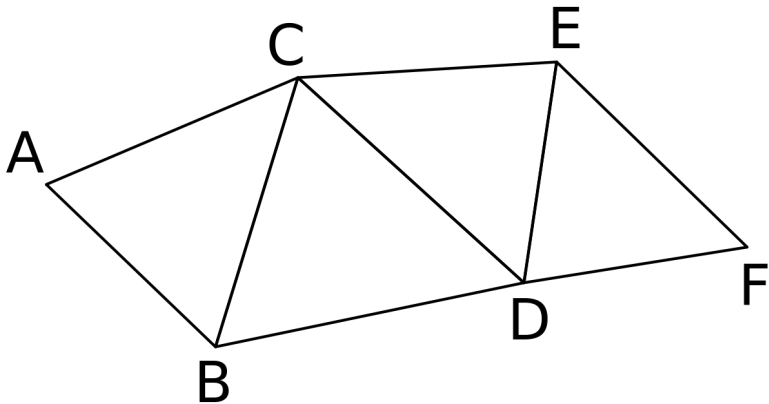
# Vertices

Triangles are drawn with vertices. All shapes on the screen are represented by groups of triangles

```
struct Vertex {  
    let position: vector_float4  
    let color: vector_float4  
}
```

## Vertices - Triangle Strip

A more memory efficient way to draw lots of triangles





# Buffers - Vertex Buffer

- Byte array containing structs with vertex information
- Order is important

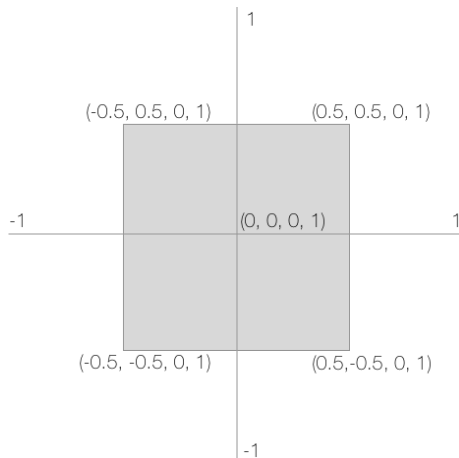
# Buffers - Index Buffer

- Byte array containing Ints
- Reference vertices in the Vertex Buffer
- Defines which order vertices are drawn

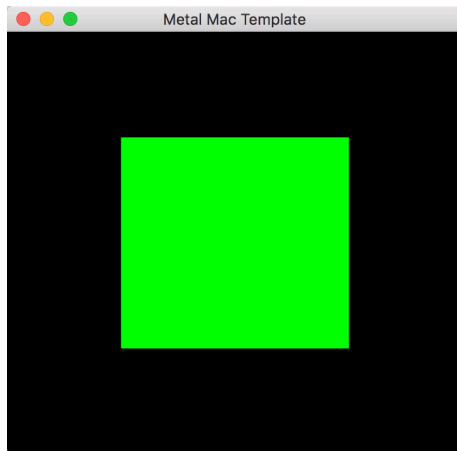
# Buffers - Uniform Buffer

- Byte array containing uniform structs
- Uniforms define transformations for each vertex
- Must align with the Vertex Buffer

## Exercise - Draw a Box



## Exercise - Expected Result



## Exercise - Vertex Buffer

```
[ Vertex{position: -0.5, 0.5, 0, 1, color: 0, 1, 0, 1},  
  Vertex{position: -0.5, -0.5, 0, 1, color: 0, 1, 0, 1},  
  Vertex{position: 0.5, 0.5, 0, 1, color: 0, 1, 0, 1},  
  Vertex{position: 0.5, -0.5, 0, 1, color: 0, 1, 0, 1} ]
```

## Exercise - Index Buffer

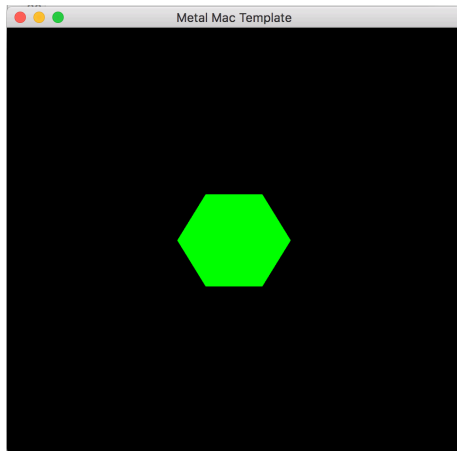
[0, 1, 3, 2]

## Exercise - Uniform Buffer

```
[ BoxUniform{ },  
  BoxUniform{ },  
  BoxUniform{ },  
  BoxUniform{ } ]
```



# Drawing a Circle using Polygons



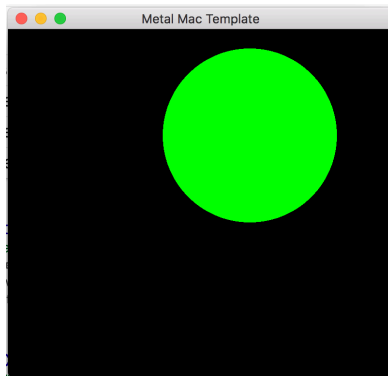
# Drawing a Circle using Polygons

The more facets, the smoother circle, right?

GPU

# Drawing a Circle More Efficiently

We can draw a *smooth* circle inside a box using the fragment shader and save memory at the same time!



# Fragment Shader

A function that takes a **Vertex** as input and provides a **color** as output

## Fragment Shader Function Signature

```
fragment float4 box_fragment_main(OutVertex outVertex [[stage_in]])
```

# Fragment Shader

Allows you to perform calculations on a per-pixel basis using the *window coordinate system*

# Fragment Shader

We would draw a circle by calculating an individual pixel's distance from the origin of the circle.

If it falls within some radius, return the vertex color, otherwise return clear

## Exercise - Draw a Circle

There is a function provided to you in Metal called **distance\_away** located in **Util.metal** that you need

### Distance Function Signature

```
float distance_away(float2 p1, float2 p2);
```



# Animations

Animations are made using Uniforms. Change a vertex's uniform every frame and you have an animation

Uniforms are applied in the Vertex Shader

# Animations

This will make the box appear to fade onto the screen

<b>Frame 1</b>	<code>BoxUniform{alpha: 0}</code>
<b>Frame 15</b>	<code>BoxUniform{alpha: 0.25}</code>
<b>Frame 30</b>	<code>BoxUniform{alpha: 0.5}</code>
<b>Frame 45</b>	<code>BoxUniform{alpha: 0.75}</code>
<b>Frame 60</b>	<code>BoxUniform{alpha: 1}</code>

## Exercise - Fade Circle In

Hint: use input values for *PulsingCircleUniform's animationTime* property in  $(0, \pi)$ .  $\text{abs}(\sin(x))$  has a period of  $\pi$ .

## Exercise - Shockwave

Make a small change to the fragment shader to draw a shockwave

# Exercise - Timing Function

## Cubic Bezier Curve

$$f(t) = (1 - t)^3 P_0 + 3(1 - t)^2 t P_1 + 3(1 - t) t^2 P_2 + t^3 P_3$$

*Hint: GPU's excel at performing dot products*