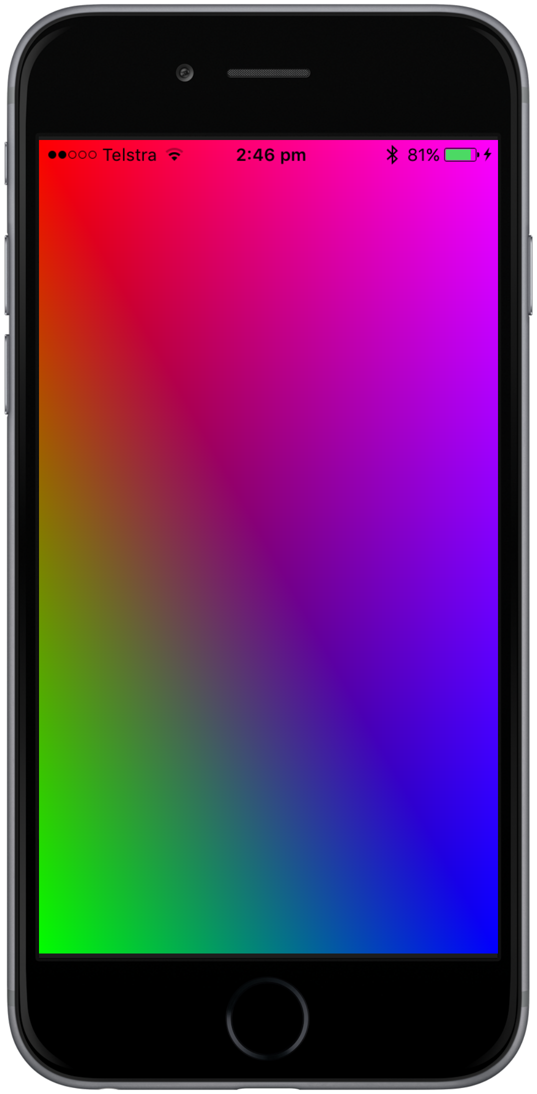
Beginning Metal

Part 5: Shaders


●●○○○ Telstra
2:46 pm
81%

Shaders

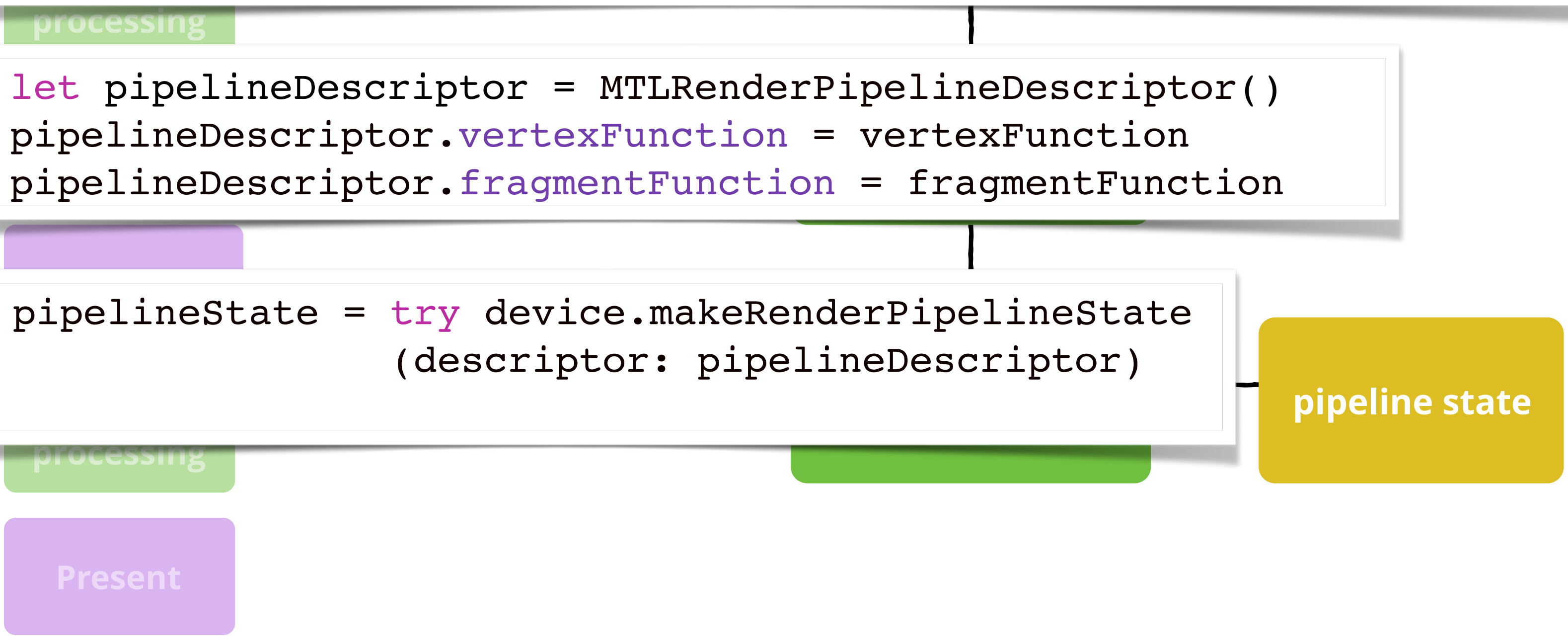
- Shader functions run on the GPU
- Three types of shader functions:
  - vertex - change vertex positions
  - fragment - change pixel colors
  - kernel - big data

The Pipeline State

```swift
private func buildPipelineState() {
  let library = device.newDefaultLibrary()
  let vertexFunction = library?.makeFunction(name: "vertex_shader")
  let fragmentFunction = library?.makeFunction(name: "fragment_shader")
```

```swift
  let pipelineDescriptor = MTLRenderPipelineDescriptor()
  pipelineDescriptor.vertexFunction = vertexFunction
  pipelineDescriptor.fragmentFunction = fragmentFunction
```

```swift
  pipelineState = try device.makeRenderPipelineState
                      (descriptor: pipelineDescriptor)
}
```



Vertex Function (1)

```
struct Constants {
  float animateBy;
};

vertex float4 vertex_shader(const device packed_float3 *vertices [[buffer(0)]],
                            constant Constants &constants [[buffer(1)]],
                            uint vertexId [[vertex_id]]) {
  float4 position = float4(vertices[vertexId], 1);
  position.x += constants.animateBy;
  return position;
}
```

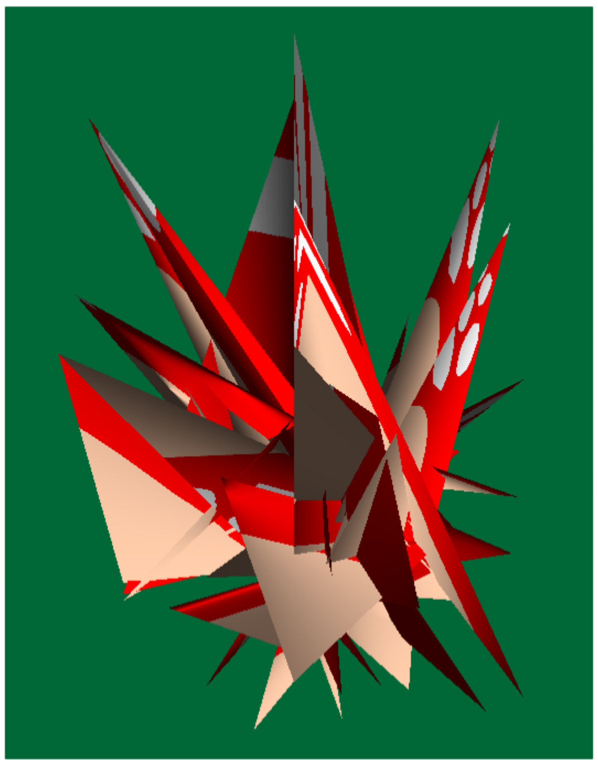
Vertex Descriptors

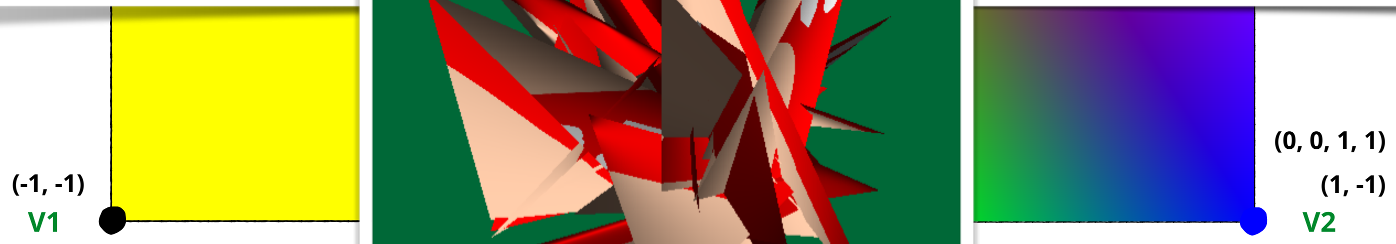
```
let vertexDescriptor = MTLVertexDescriptor()
                                              float3

                                              x = 0
                                              float4
                                             emoryLayout<float3>.stride
                                              x = 0
vertexDescripto

vertexDescripto                              ryLayout<Vertex>.stride
```

```
struct Vertex {
  var position:
  var color: fl
}
```





Vertex Function (2)

```
struct Vertex {
  var position: float3
  var color: float4
}
```

```
struct VertexOut {
  float4 position [[position]]; ];
  float4 color;
};
```

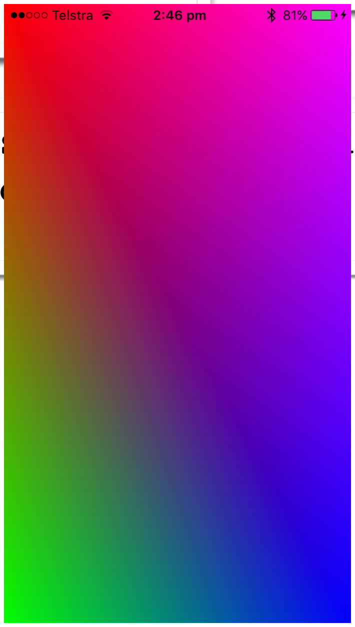
```
vertex VertexOut vertex_shader(const VertexIn vertexIn [[stage_in]]) {
  VertexOut vertexOut;
  vertexOut.position = vertexIn.position;
  vertexOut.color = vertexIn.color;
  return vertexOut;
}
```

The Fragment Function

```
struct VertexOut {
  float4 position [[position]];
  float4 color;
};
```

```
{
```

```
fragment half4 fragment_              t vertexIn [[ stage_in ]]) {
  return half4(vertexIn.
}
```



Demo

Challenge Time!