BEGINNING METAL

HANDS-ON CHALLENGES

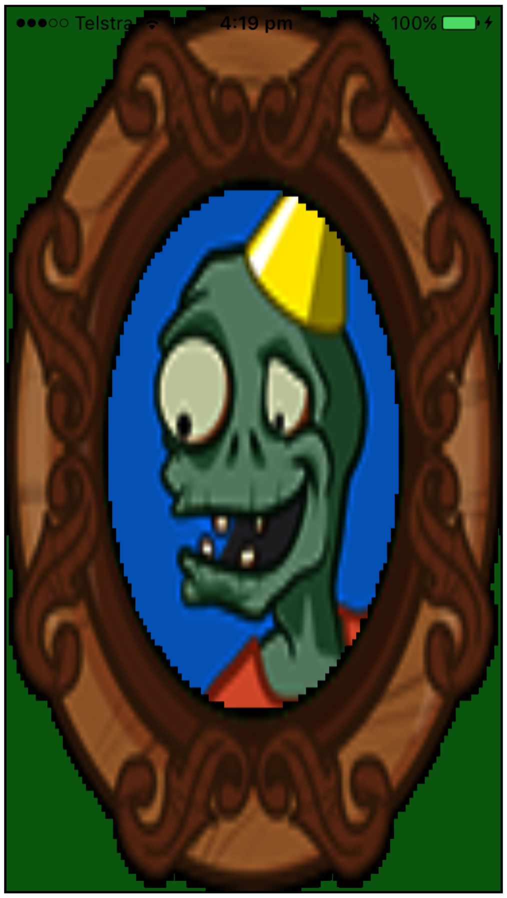
Beginning Metal

Caroline Begbie

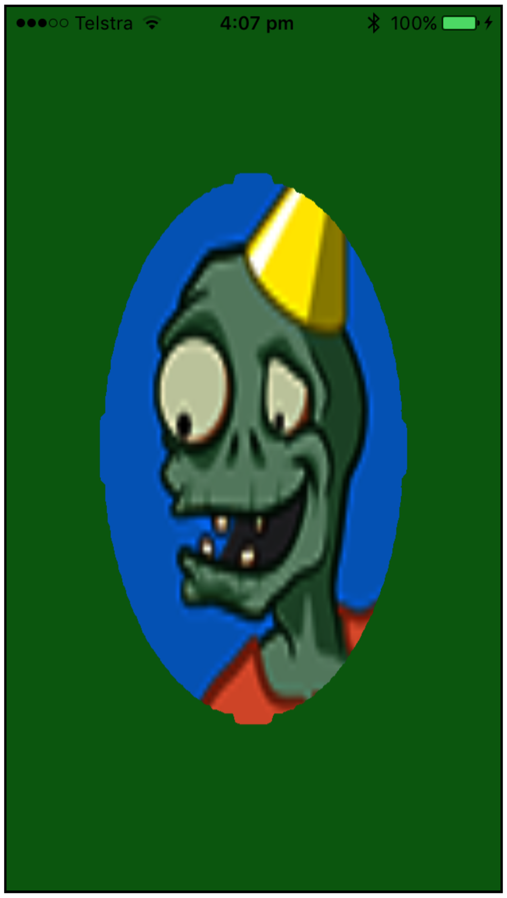
Challenge #6: Masking

By Caroline Begbie

Your challenge is to make the zombie appear in a picture frame. You'll first mask the zombie, and then add a second quad which will be the picture frame.



Take a look at picture-frame-mask.png. This is a white circle with all other pixels transparent. You'll check the zombie texture against this texture and only output the zombie texture where the mask pixels are white.



First create a mask texture property in `Plane`:

```
var maskTexture: MTLTexture?
```

In `Plane`, you'll create a third initializer to take a mask image name. Copy the previous initializer and change it to this:

```
init(device: MTLDevice, imageName: String, maskImageName: String) {
  super.init()
  buildBuffers(device: device)
  if let texture = setTexture(device: device,
                              imageName: imageName) {
    self.texture = texture
    fragmentFunctionName = "textured_fragment"
  }
  if let maskTexture = setTexture(device: device,
                                  imageName: maskImageName) {
    self.maskTexture = maskTexture
    fragmentFunctionName = "textured_mask_fragment"
  }
  pipelineState = buildPipelineState(device: device)
}
```

Here you accept the mask image name as a third parameter, and get the mask texture as a `MTLTexture` in the same way as the original texture.

If the mask texture is loaded correctly, you set the fragment function to be the fragment function that you'll create in a minute.

Change `GameScene` to call this initializer:

```
quad = Plane(device: device,
             imageName: "picture.png",
             maskImageName: "picture-frame-mask.png")
```

In `Plane`, in `render(commandEncoder:deltaTime:)`, after setting the original fragment texture, set the mask texture's fragment buffer to buffer index `1`:

```
commandEncoder.setFragmentTexture(maskTexture, at: 1)
```

Now create a new fragment function. In **Shader.metal** add this new function at the end of the file:

```
fragment half4 textured_mask_fragment(VertexOut vertexIn [[ stage_in ]],
                              texture2d<float> texture [[ texture(0)]],
                              texture2d<float> maskTexture [[ texture(1) ]],
                              sampler sampler2d [[sampler(0)]]) {
  float4 color = texture.sample(sampler2d, vertexIn.textureCoordinates);
}
```

So far, this is the same as the previous texture function, except for the extra mask texture parameter. The mask texture is in fragment buffer `1`.

To test whether the fragment is masked, first get the fragment color from the mask texture:

```
float4 maskColor =
    maskTexture.sample(sampler2d, vertexIn.textureCoordinates);
```

Then check the opacity:

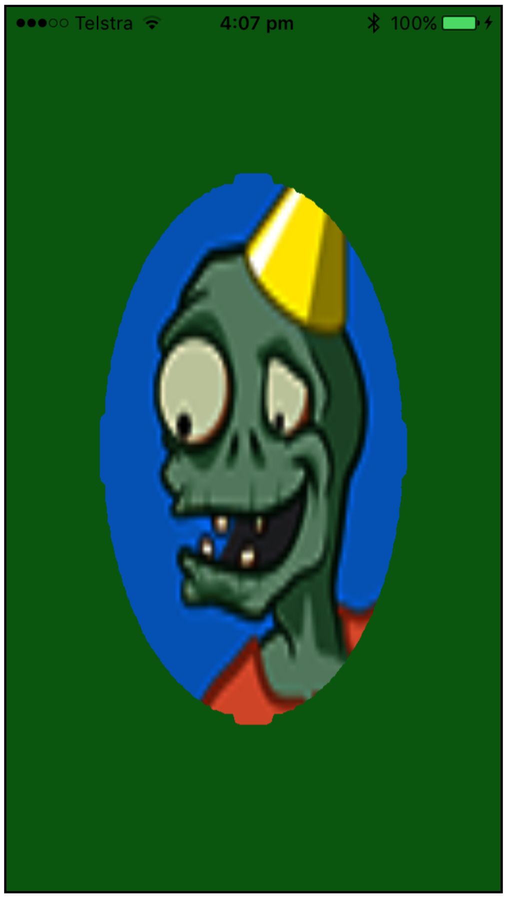
```
float maskOpacity = maskColor.a;
if (maskOpacity < 0.5)
  discard_fragment();
```

If the opacity is less than 0.5, discard the fragment. This means that the fragment will be empty when rendered.

Return the fragment color for the fragments that aren't discarded:

```
return half4(color.r, color.g, color.b, 1);
```

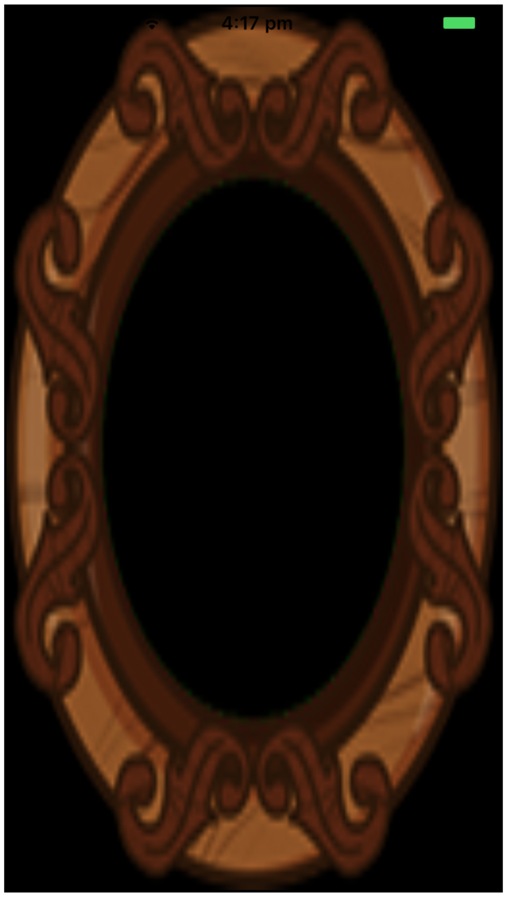
Build and run, and see your zombie masked out.



The next part of the challenge is to put a picture frame on top of this texture. To do this, you'll add a second textured quad to the scene. In `GameScene`, at the end of `init(device:size:)`, add this:

```
let pictureFrame = Plane(device: device,
                         imageName: "picture-frame.png")
add(childNode: pictureFrame)
```

Build and run, and you'll see your picture frame.



Unfortunately, where the picture frame pixels are transparent, you're just seeing black. The fragment function you're using for rendering this quad is the previous textured fragment function without the mask, so it doesn't check whether pixels are transparent or not.

> **Note**: You can check which fragment function is being used for each draw call with the GPU debugger.

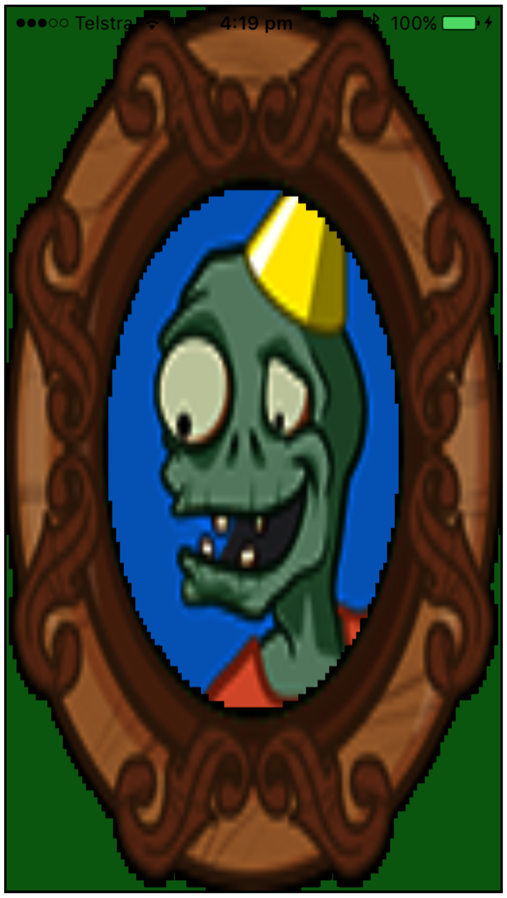
In **Shader.metal**, find the fragment function called `textured_fragment(...)`

Before returning the color, add this:

```
if (color.a == 0.0)
  discard_fragment();
```

Here you discard the fragment if the color's alpha property is zero.

Build and run and you'll see your party zombie happily framed against the view's green background.



You'll have noticed that the textures are stretched to fit the quad. In the next video we'll go from 2d to 3d, and in the process we'll be able to make our quad square as it should be.