BEGINNING METAL

HANDS-ON CHALLENGES

Beginning Metal

Caroline Begbie

Notice of Rights

Notice of Liability

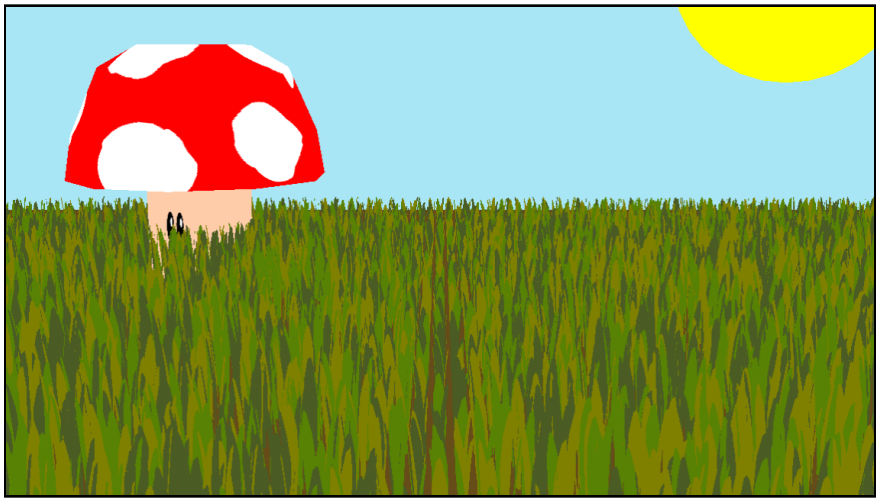
Trademarks

Challenge #10: A 3D Scene

By Caroline Begbie

In this challenge you're going to build a 3d landscape scene adding onto the scene that you created in the previous challenge.

You're going to create a ground plane, and use a grass model that you'll find in the Resources folder for this video. You'll load the blades of grass onto the ground plane, and also add a mushroom character.



Firstly, drag the grass model into your project from the Resources folder.

In your Landscape scene, add the properties you'll need for ground plane, grass and mushroom:

```
let ground: Plane
let grass: Instance
let mushroom: Model
```

Initialize them in the initializer before calling `super`

```
ground = Plane(device: device)
grass = Instance(device: device, modelName: "grass", instances: 10000)
mushroom = Model(device: device, modelName: "mushroom")
```

You'll be creating 10,000 blades of grass.

> **Note**: I wouldn't recommend making a game with 10000 blades of grass without further optimization. We're currently unnecessarily updating all the instances every frame and the CPU and battery usage will be extrememly high.

Create a new method to set up the scene:

```
func setupScene() {
}
```

In `setupScene()`, color the ground plane:

```
ground.materialColor = float4(0.4, 0.3, 0.1, 1) // brown
```

Add the three models to the scene:

```
add(childNode: ground)
add(childNode: grass)
add(childNode: mushroom)
```

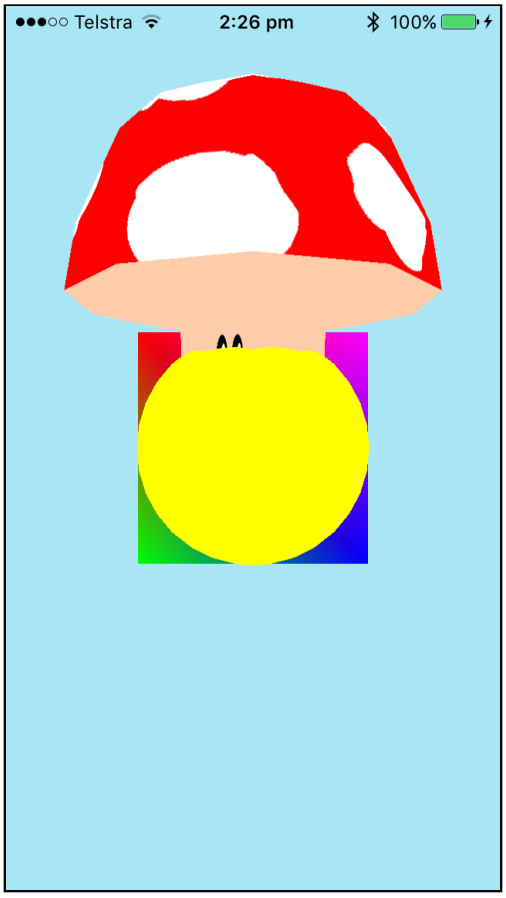
And call `setupScene()` at the end of `init()`:

```
setupScene()
```

Make sure you set `LandscapeScene` as the main scene in `ViewController`:

```
renderer?.scene = LandscapeScene(device: device, size: view.bounds.size)
```

Build and run, and you'll see that you've loaded the models, but the scene still has a little way to go.



The plane is multi-colored because we set up that models can have a material color, but we didn't do the same for `Primitives`.

So in `Primitive`, change the fragment function so that the color comes from the material color instead of each vertex:

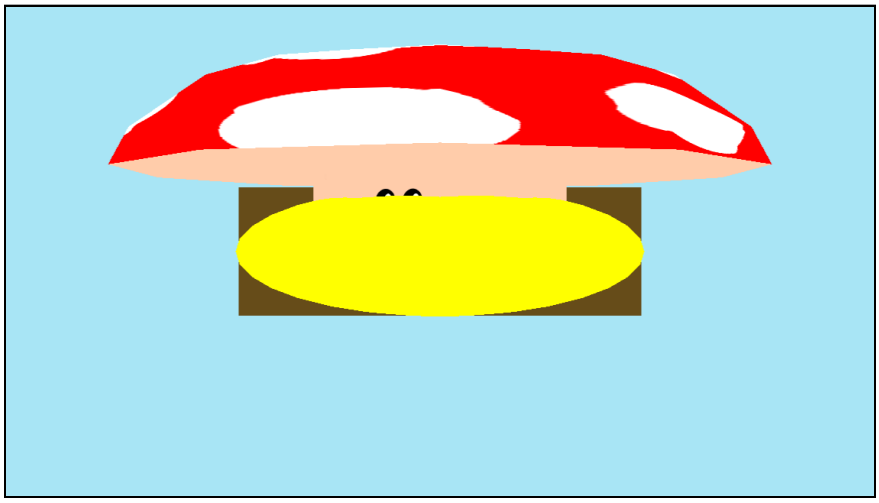
```
var fragmentFunctionName: String = "fragment_color"
```

In `Primitive`'s `doRender(commandEncoder:modelViewMatrix:)`, add the material color to the model constants struct:

```
modelConstants.materialColor = materialColor
```

Build and run, and your ground plane will now be brown.

You may have noticed that when you rotate your phone, the models don't resize properly.



Now's the time to fix that.

The camera's aspect ratio and scene's projection matrix needs to be updated every time the screen rotates.

Add a new method to `Scene`:

```
func sceneSizeWillChange(to size: CGSize) {
  camera.aspect = Float(size.width / size.height)
}
```

Remove this from `init(device:size:)`:

```
camera.aspect = Float(size.width / size.height)
```

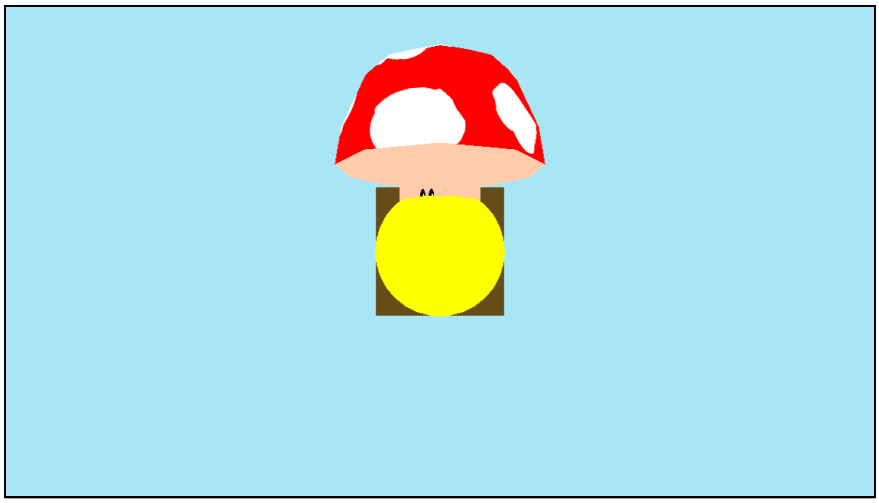
`sceneSizeWillChange(to:)` must be called every time the scene size changes. The place that gets notified of this change is the Metal View's delegate, `Renderer`. In `Renderer`, in the delegate extension, you'll find the method that so far we've ignored - `mtkView(_:drawableSizeWillChange:)`

Add this to the method:

```
scene?.sceneSizeWillChange(to: size)
```

Every time the device rotates or the metal view changes size, the scene's method will be called and the scene camera's aspect ratio will update accordingly.

Build and run, and your scene should resize properly.



Now to rearrange the objects in the scene.

Firstly, scale the ground plane to be really large.

In `LandscapeScene`, in `setupScene()` add this:

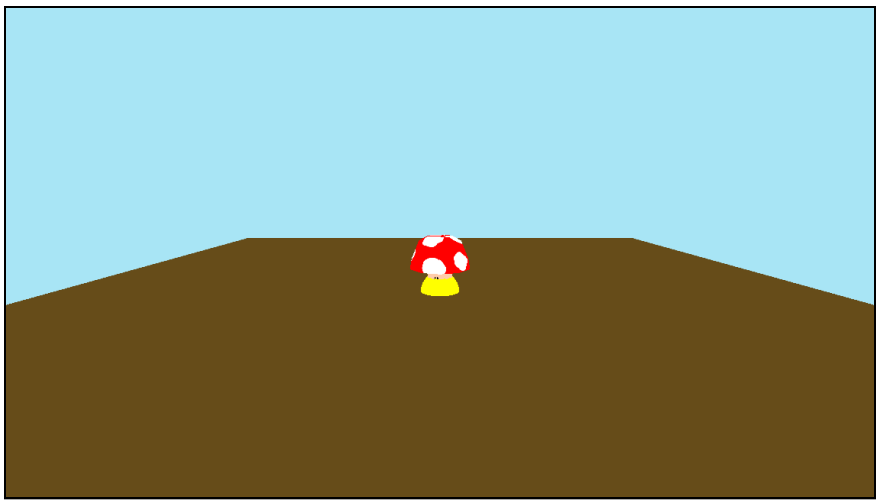
```
ground.scale = float3(20)
ground.rotation.x = radians(fromDegrees: 90)
```

This rotates the ground plane so that it will look like ground.

Rotate and move the camera so that you're looking down on the scene:

```
camera.rotation.x = radians(fromDegrees: -10)
camera.position.z = -20
camera.position.y = -2
```

I got these camera settings by a bit of trial and error to get the best position.



Continue adding to `setupScene()`. You'll color your grass one of 3 different greens:

```
let greens = [
  float4(0.34, 0.51, 0.01, 1),
  float4(0.5, 0.5, 0, 1),
  float4(0.29, 0.36, 0.14, 1)
]
```

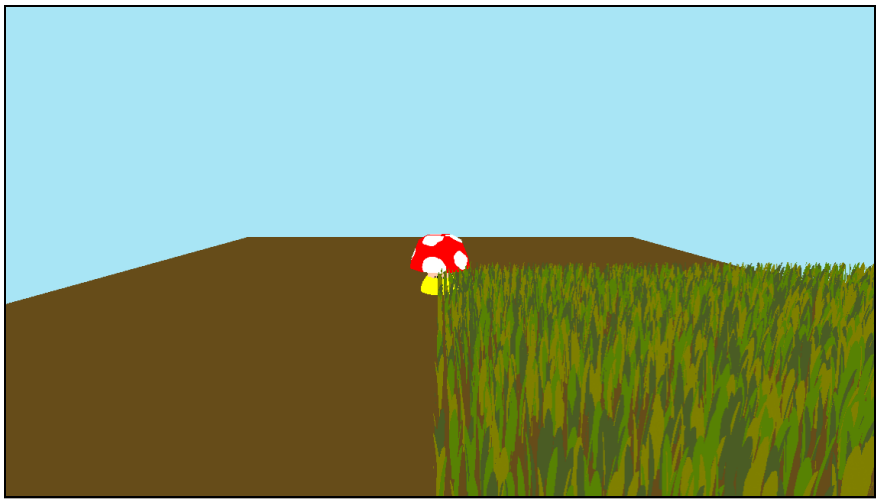
Set up each blade of grass like this:

```
for row in 0..<100 {
  for column in 0..<100 {
    var position = float3(0)
    position.x = (Float(row))/4
    position.z = (Float(column))/4

    let blade = grass.nodes[row * 100 + column]
    blade.scale = float3(0.5)
    blade.position = position

    blade.materialColor = greens[Int(arc4random_uniform(3))]
    blade.rotation.y =
        radians(fromDegrees: Float(arc4random_uniform(360)))
  }
}
```

Here you're looping through 10000 blades of grass in rows and columns and setting their x and y positions. You're scaling each blade of grass because I made it a bit big, and setting the grass to a random green color. You're also randomly rotating the blade so it doesn't look too uniform.



The grass has started from the origin which is at the middle. Because all the blades of grass are in one `Instance`, and that `Instance` is a `Node` subclass, we can position the instance node individually. At the end of `setupScene()`, move the whole instance back and left in the scene.

```
grass.position.x = -12
grass.position.z = -12
```

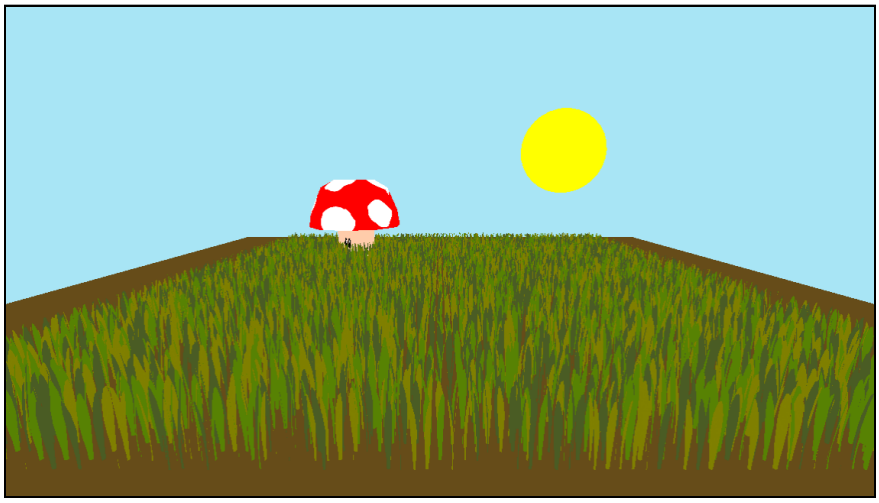
Position the mushroom at the back of the scene:

```
mushroom.position.x = -6
mushroom.position.z = -8
mushroom.scale = float3(2)
```

and position the sun:

```
sun.position.y = 7
sun.position.x = 6
sun.scale = float3(2)
```

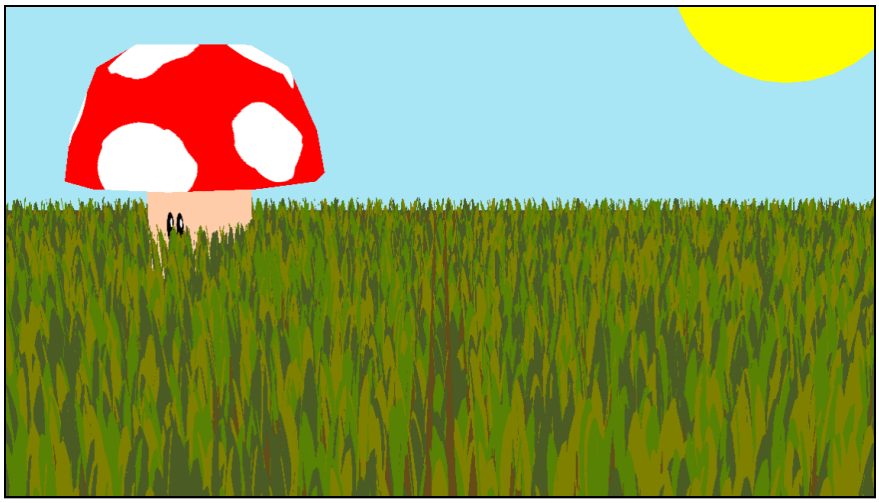
Now when you run the app, all the models are in place.



The field of view of the camera is rather wide, so change it at the end of `setupScene()`:

```
camera.fovDegrees = 25
```

And build and run and here's your wonderful pastoral scene:



Congratulations :]. You can now add models to your scene and you're well on the way to have a completed mini-game engine.