ARKit Workshop

Michael Yagudaev
Nano 3 Labs

Schedule/Outline

- Part 1: 6:30-7:15pm (Basics with Primitives)
- Break 1: 7:15pm-7:20pm
- Part 2: 7:20-8:05pm (Loading 3D Model)
- Break 2: 8:05-8:10pm
- Part 3: 8:10-8:55pm (Responding to Gestures)

About Michael

- Co-Founder @ PictureThat & Nano 3 Labs
- Background as Web/Mobile Developer
- No prior knowledge of 3D/Game programming
- Been working with ARKit since July, 2017
- Worked on 3D Modeling & Texturing as a teenager
- Mentored at Lighthouse for 2-years
- Fun Fact: Snowboarding is my freedom

ARKit?

- ARKit is a framework to place virtual objects inside a camera frame

- It tracks the movement of the phone in space and observes feature points using multiple camera frames to create a ***point cloud*** also known as ***ARAnchors***

- This gives us the coordinates of a point in the real world (wall, floor, chair, table, etc) relative to the ***World Origin***

- We can then use a 3D engine like ***SceneKit***, ***Unity*** or ***Unreal***

SceneKit?

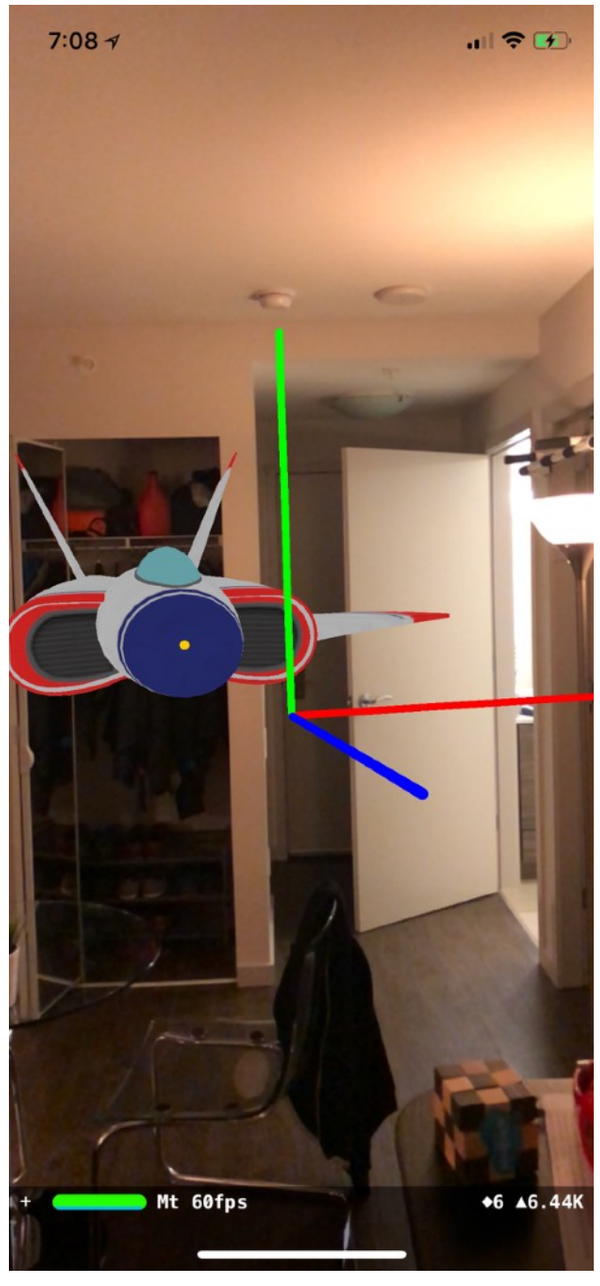
- SceneKit is a 3D and Physics engine
- Developed by Apple and can be used on iOS & Mac
- It will allow us to render 3D content in our AR scene
- Remember, AR is the creating the illusion a virtual object is part of a real-world scene

Understanding ARKit

- Let’s start by creating a new ARKit Project
- Download the starting repo & open it in xcode
  - https://github.com/nano3labs/arkit-workshop
- Add the debugging code and launch it

X, Y, Z Coordinates

- Y-upper world
- X - Red
- Y - Green
- Z - Blue
- World Origin is the (0, 0, 0) and resets every session. Everything is relative to it
- ARKit units are always in Meters



Scene Editor

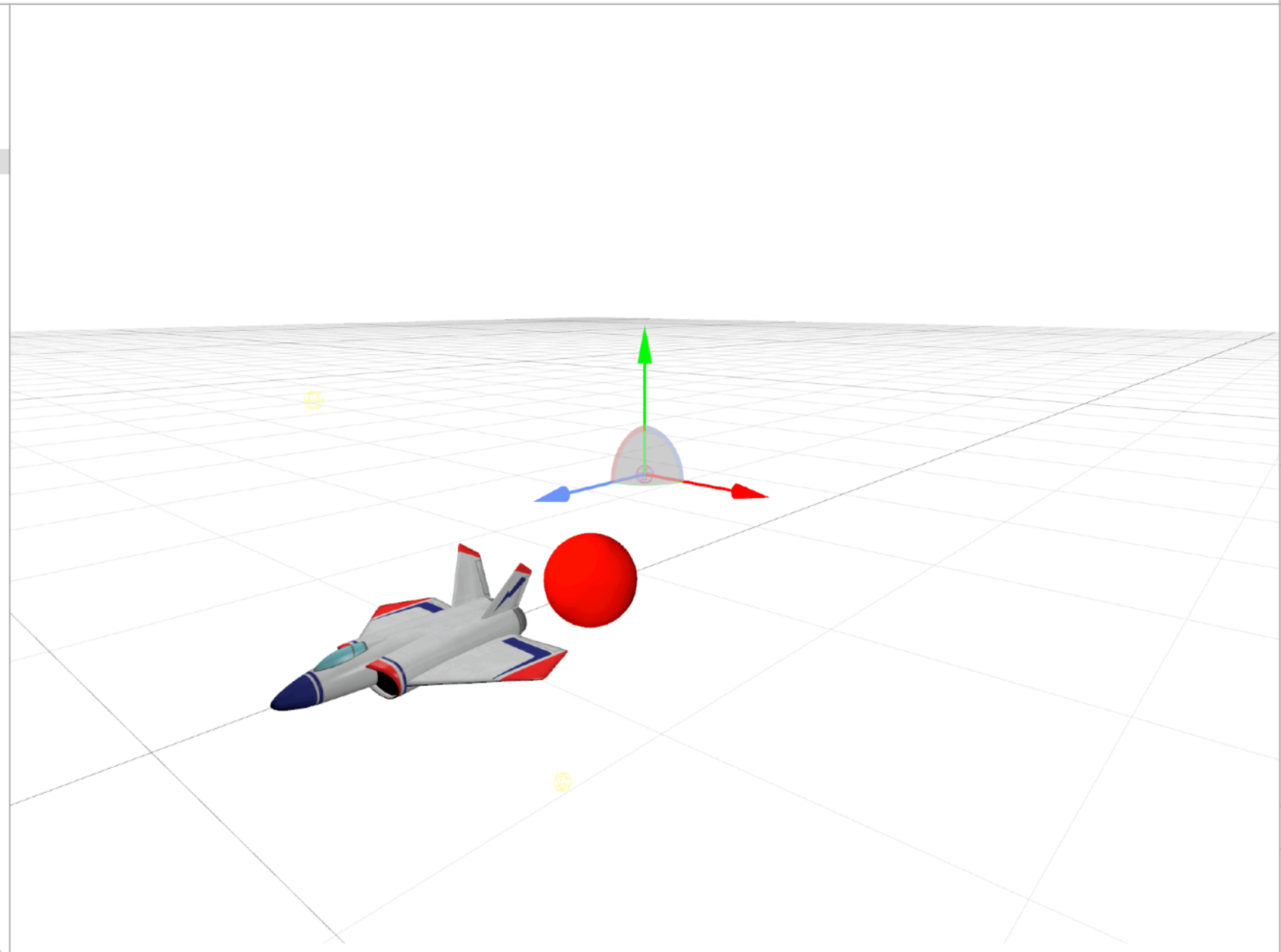
- View/Edit 3D objects
- View/Edit Node graph
- Add primitive geometry, lights, textures, etc
- Also useful for live debugging

LunarLanding 〉 Michael Yagudaev's iPhone | Running LunarLanding on Michael Yagudaev's iPhone | 1

LunarLanding 〉 LunarLanding 〉 art.scnassets 〉 ship.scn 〉 Scene graph 〉 right

Scene graph

- ship
- sphere
- left
- bottom
- right



Filter | local | Perspective | Turntable

Light

Name: omni
Identifier: --
Type: Omni
Mode: Dynamic
Color:

Intensity

1000

Attenuation

Start: 0
End: 0
Falloff: 2

Category bit...: -1

Empty Node - An empty SCNNode.

Omni light - Illuminates the scene from a point in every direction.

Directional light - Illuminates the scene in a specific direction.

LunarLanding

```
2018-01-20 21:12:43.233554-0800 LunarLanding[7964:3080190] [MC] System group
container for systemgroup.com.apple.configurationprofiles path is /private/var/
containers/Shared/SystemGroup/systemgroup.com.apple.configurationprofiles
2018-01-20 21:12:43.234576-0800 LunarLanding[7964:3080190] [MC] Reading from
public effective user settings.
```

Auto | Filter | All Output | Filter | Filter

Adding Nodes Programatically

- Editor is good for basic things
- To get the most out of ARKit we need to use code
- Let’s add the moon sphere using code
- We will use SCNNode, SCNSphere, SCNMaterial

Live Debugging

- We can pause the scene and inspect it

- Useful for finding the state of the scene graph

LunarLanding › Michael Yagudaev's iPhone | Running LunarLanding on Michael Yagudaev's iPhone | 1

LunarLanding › LunarLanding.ViewController › ARSCNView › SCNScene

LunarLanding PID 8062

CPU 15%

Memory 216.1 MB

Energy Impact

Disk Zero KB/s

Network Zero KB/s

FPS 60 FPS

UIWindow

LunarLanding.ViewController

ARSCNView

SCNScene

Scene graph

<untitled>

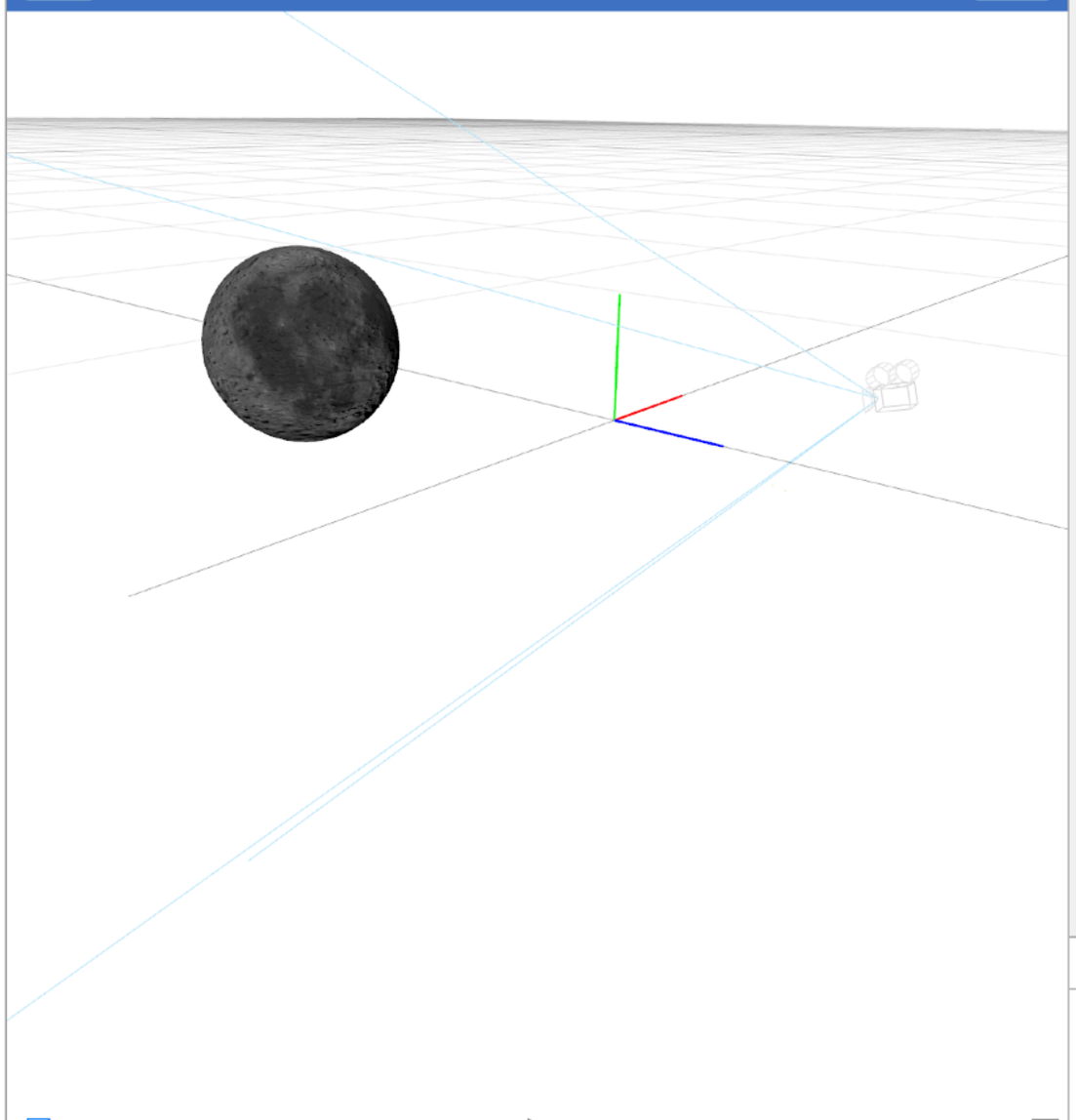
<untitled>

<untitled>

<untitled>

<untitled>

Reset | Runtime Scene | Export



Not Applicable

Filter

local | Perspective | Turnta...

LunarLanding › Thread 1 › 0 mach_msg_trap

[Technique] World tracking performance is being affected by resource constraints [0]
2018-01-20 22:06:29.873673-0800 LunarLanding[8062:3112305] [Technique] World tracking performance is being affected by resource constraints [0]
(lldb)

Auto | Filter | All Output | Filter | Filter

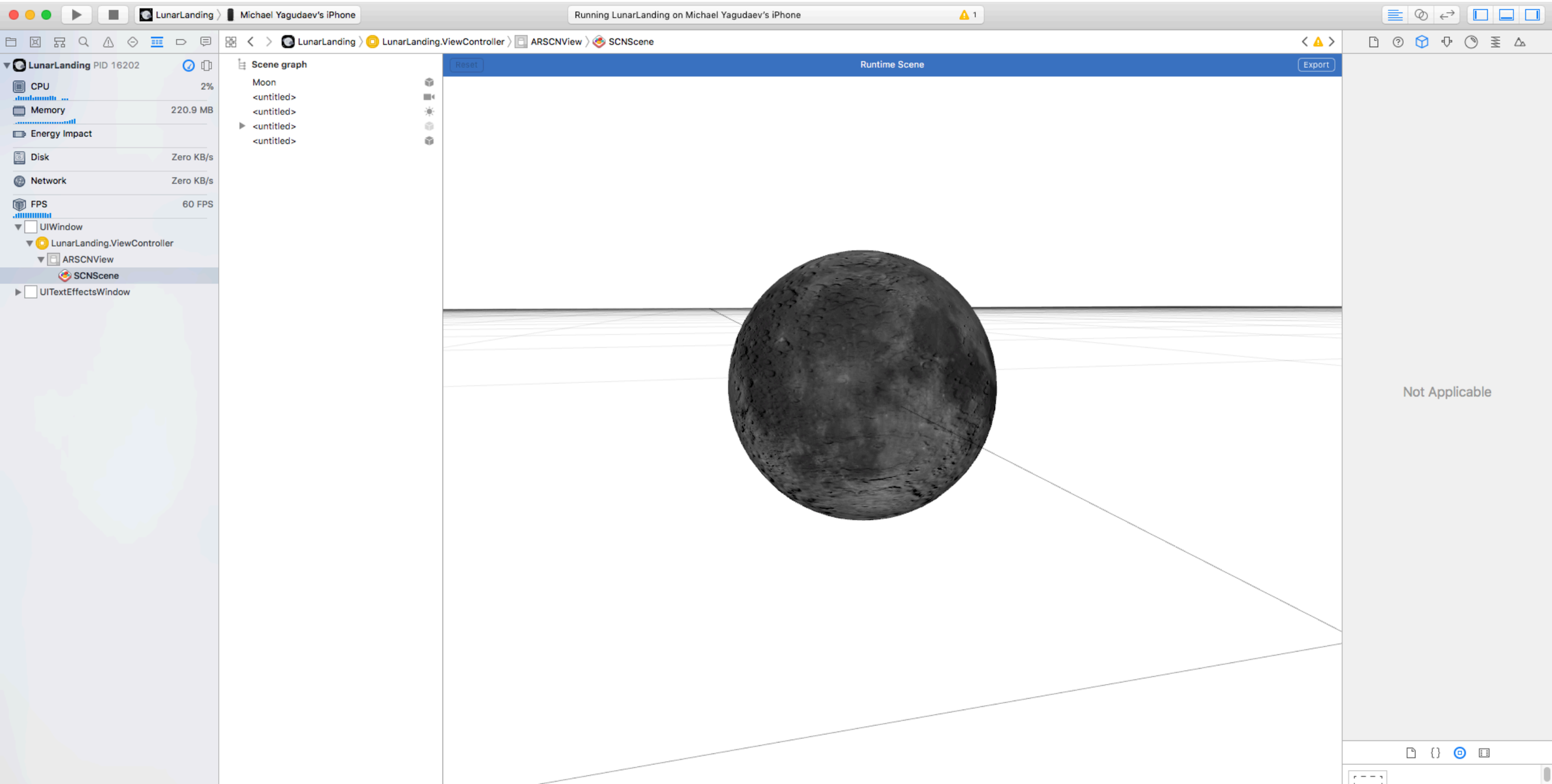
Empty Node - An empty SCNNode.

Omni light - Illuminates the scene from a point in every direction.

Directional light - Illuminates the scene in a specific direction.

Debugging Tip

- Add names to SCNNodes to make it easier



Tracking State

- ARKit initializes the scene and waits for user movement
- It tries to form a point cloud and to place objects where they should be (re-localization)
- Until then, it approximates the location using a few clever tricks, but the location is inaccurate
- Let’s display the tracking information
- use: func session(_ session: ARSession, cameraDidChangeTrackingState camera: ARCamera)

Plane Detection & ARAnchors

- As we place object in the scene ARKit automatically places ARAnchors at those positions

- ARAnchors tell the framework to track the location of that object/point as we move through the scene

- ARKit can also track horizontal planes for us upon request

- Everytime a horizontal plane is found ARKit will notify us that an ARPlaneAnchor has been added to the scene

- Let’s give it a try

- use: func renderer(_ renderer: SCNSceneRenderer, didAdd node: SCNNode, for anchor: ARAnchor)

- and func renderer(_ renderer: SCNSceneRenderer, didUpdate node: SCNNode, for anchor: ARAnchor)

User Interaction: Hit Testing

- In order to translate a 2D interaction to 3D space we use hit testing

- We send a ray out to our scene from a 2D point on the screen until it hits something in the scene

- Let’s try it out by adding a cube to the scene when a user touches a surface.

- We will use UITapGestureRecognizer + sceneView.hitTest

Import 3D Model

- Find model on 3D marketplace like SketchFab
- Xcode needs Colleda DAE file format and texture files
- Use open source Blender to export from formats like .obj to a .dae file
- Import into Xcode
- Convert to .scn file
- Load model into your scene

Physics Engine

- We can enable physics to bring the scene into life
- SCNPhysicsBody
- Three types of bodies:
- 1) Static - don’t move
- 2) Dynamic - move & can collide with things
- 3) Kinematic - like static but can be animated
- Let’s give it a try

Particle effects

- We can create particle effects like fire, smoke, sparks, explosions, etc

- Simply create the effect and adjust the values and import and attach to see

- Let’s see it in action

Animations

- Other than Physics forces, there are 3 other way to animate objects in SceneKit:
    - SCNActions - let use use predefined animations
    - SCNTransactions - let use specify the start and end value and the duration of the animation
    - Render loop - tying into the render loop gives unlimited flexibility
- Let’s add an animation to the moon using SCNActions

Moving Objects on Plane

- Lastly, we want to be able to move objects on a plane
- We will use another type of hitTest to find the plane coordinates
- We will also make use of UIPanGestureRecognizer
- Let’s try it out

References

- https://www.raywenderlich.com/146175/scene-kit-tutorial-swift-part-1-getting-started

- https://www.udemy.com/scene-kit/learn/v4/overview

- https://developer.apple.com/documentation/arkit/building_your_first_ar_experience

- https://developer.apple.com/ios/human-interface-guidelines/technologies/augmented-reality/

- http://www.idownloadblog.com/2018/01/25/arkit-1-5-features/