Introduction to ARKIT

WeAreDevelopers iOS days - April 2021

About me

- I am a full stack developer working in the industry for over 7 years
- Worked on: **Lufthansa**, **Coca Cola, Raiffeisen Bank...**
- My main focus in terms of development are iOS applications
- I co-founded a coworking space that got voted among top 100 in Europe
- Feel free to get in touch

Let’s talk about ARKit!

Few keywords first...

- AR stands for Augmented Reality (not to be confused with Virtual Reality)
- AR consists of: **gathering** input/information from device sensors, **evaluating** them and **augmenting** real world images with additional information

ARKit 1.0; 1.5; 2.0 ……

- Apple has released 4 versions of ARKit framework so far

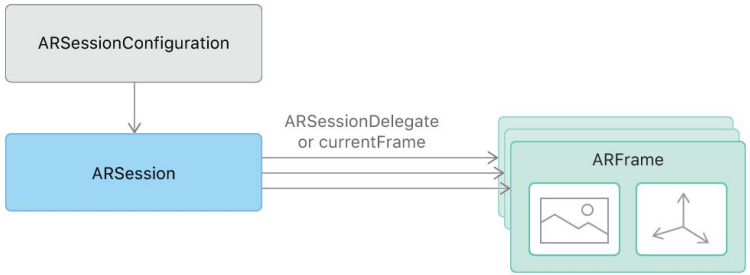
Version	**1.0 (iOS 11)**	**1.5 (iOS 11.3)**	**2.0 (iOS 12)**	**3.0(iOS 13)**
Detecting Horizontal planes	✓	✓	✓	✓
Detecting Vertical planes		✓	✓	✓
2D image/object recognition		✓	✓	✓
3D object recognition			✓	✓
Face tracking				✓

ARKit requirements

- A physical device is needed with an A9 chip (iPhone 6s or later and all iPads from 2017)
- Can’t test it on a simulator, so a long lightning cable or a WiFI debugger is needed
- iPhone or iPad needs to be on a version of iOS 12+
- Xcode minimum version is 10
- Solid amount of physical space in a room where the app will be tested

ARKit image rendering options

- ARKit offers three options for rendering images on top of real world (a scene you view through your camera)
- **SceneKit** - Rendering of 3D objects
- **SpriteKit** - Rendering of 2D objects
- **Metal** - Hardware accelerated 3D graphics and shaders rendering framework



Source: Apple documentation

Few thoughts about coordinate systems

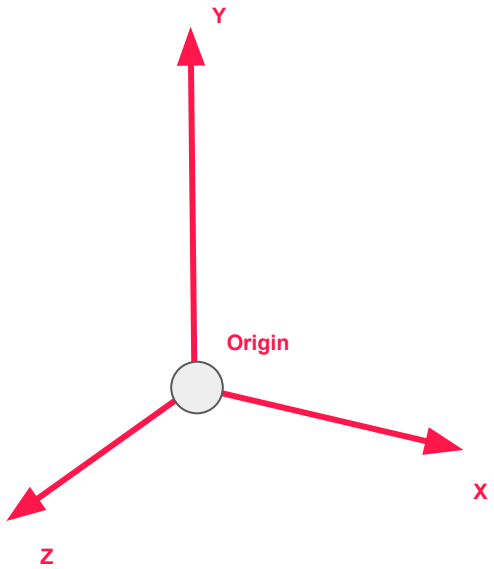
- In 2D environment, origin of a coordinate system for UIKit starts at top left corner of your phone
- X increases to the right hand side of the phone
- Y increases towards the bottom of the phone

Few thoughts about coordinate systems

- In 2D environment, origin of a coordinate system for SpriteKit or OpenGL starts at bottom left corner of your phone
- X increases to the right hand side of the phone
- Y increases towards the top of the phone

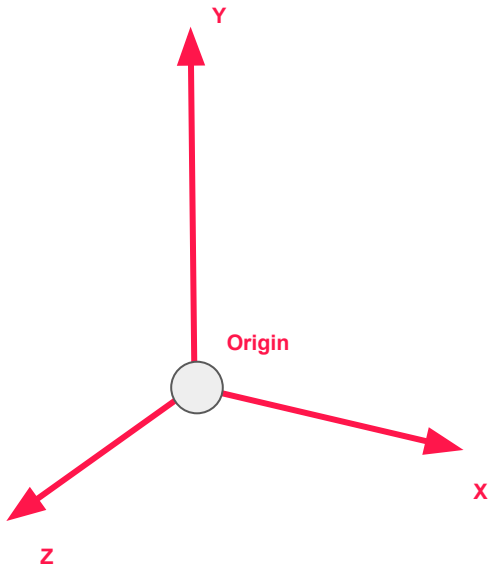
Few thoughts about coordinate systems

- In 3D space we need to deal with three axis
- The X axis still extends left and right from the origin and it's increasing as you go **right**
- The Y axis still extends up and down from the origin and is increasing as you go **up**
- The Z axis extends from and towards the origin and is increasing as it goes **"toward"** you
- Where is the origin in 3D coordinate system?



Few thoughts about coordinate systems

- ARKit origin is set at the precise physical location of the device when the ARKit session starts.
- But what defines where “left”, “right”, “up” and “down” is?
- Let’s talk about alignments…



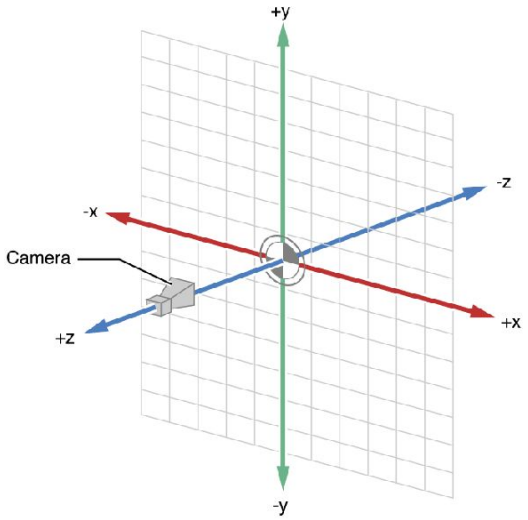
ARKit alignments

- We have three options on how to define the alignment of ARKit session
- **.gravity**, **.gravityAndHeading** and **.camera**
- Gravity alignment as the default alignment sets the origin (0, 0, 0) of ARKit according to gravity hence the Y axis decreases towards the bottom, Z decreases when you go away from the user, and X axis increases to the right side of the user.
- **.gravityAndHeading** also takes into account the compass of the device

Source:
https://developer.apple.com/documentation/arkit/arconfiguration/worldalignment/gravity

Objects in a scene

- Now that we know how to orient ourselves in 3D space let's talk about 3D objects in the scene
- All 3D objects that should be part of a scene are added to the SceneKit Graph
- SceneKit Graph is a Tree-like structure of nodes where each 3D object can either be the child of the root of the child of a child
- Scene's root node position is defined as **(0, 0, 0)**.
- The tree works much like a regular **view hierarchy** in UIKit.



Source: Apple documentation

Raycasting and how to transfer touch to coordinates

- **Raycasting in ARKit works** by casting "rays" to measure the distance to the nearest surface or plane, hence the term "**raycaster**". We “send” out rays starting from the camera, moving forward until it hits a detected surface, at which point it takes the distance it has traveled and returns the coordinates.

DEMO 1 - ARKit & SpriteKit Hello World.

DEMO 2 - ARKit & SpriteKit IKEA Place clone.

What can we improve?

- Add vertical plane detection so we can hang pictures on the walls
- Take into account different horizontal planes (so you can’t put a chair on a desk)
- Make objects draggable
- Improve light in the scene to make the objects more realistic
- Improve hit tests by adding a focus square instead of using just fingers

Thank you.