Introduction to ARKit

(with Xamarin)

RYAN DAVIS

Queensland C# Mobile Developers Meetup

2018 11 27

whoami

- Ryan Davis
- Professional ~~Mobile~~ LINQPad Developer

 ryandavis.io

 rdavis_au

 rdavisau

- essential-interfaces – use DI/mocking with Xamarin.Essentials
- jsondatacontext-linqpad – json data context driver for LINQPad
- sockets-for-pcl, sockethelpers – socket comms in a PCL
  (today you should use netstandard sockets why are you all still installing this)

to cover

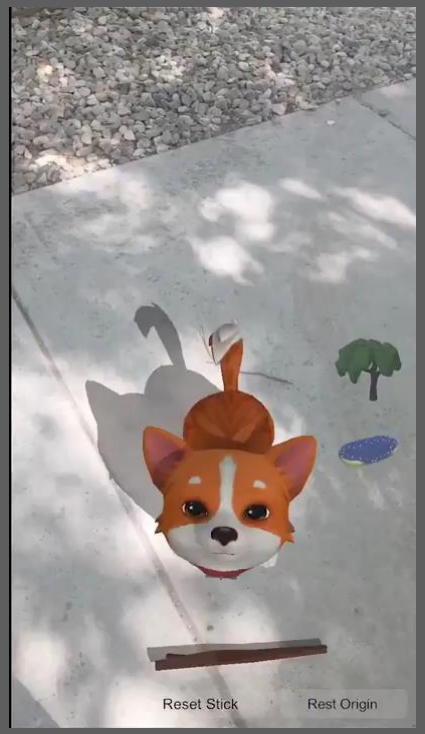
- AR in very brief, overview of ARKit
- Walk through the framework features
- Samples / demos
- Resources

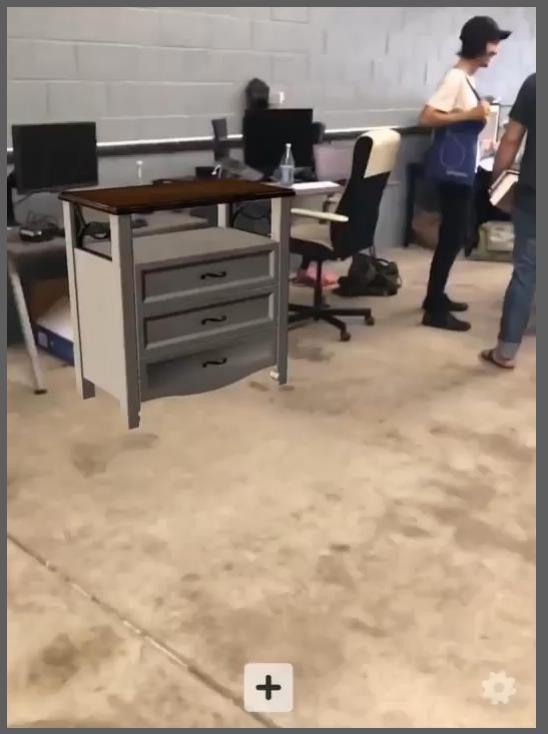
-= introduction to arkit =-

what is it?

augmented reality?

In Short: the convincing integration of virtual content into a real world space.





Videos from @madewitharkit

a high performance AR framework for iOS

"ARKit provides a platform for developing (AR) apps for iPhone and iPad, with sophisticated positional tracking and scene understanding."

(developer.apple.com, with most marketing fluff removed)

World Tracking

Scene Understanding

Rendering
(hooks)

- iOS devices with A9 processor or better (most devices since iPhone 6S)
- iOS11+ for ARKit 1.0, iOS12+ for ARKit 2.0

a laundry list of features for you to choose from

- **World Tracking:** use of "Visual Inertial Odometry" to correlate device movement and video input to precisely determine your position
- **Scene Understanding:** continuous interpretation of inputs to manage a world map while running
  - Topology: detecting planes (surfaces) and features
  - Objects: detecting 2D images or 3D objects in the real world
  - Light: detection of real-world lighting to influence brightness and rendering of virtual objects
- **Interaction:**
  - Hit-testing: allowing for interaction with 3D scene (real world) via 2D interface (your screen)
- **Persistence:**
  - Serialise a world map (including virtual additions) and deserialise it later
  - Send a world map to other devices for shared experience

Your app might use just a small number of these features

combine with frameworks for rich experiences

iOS:

- **SceneKit:** apply physics to virtual objects, animate, style and orchestrate scene contents
- **Vision framework:** detect objects, landmarks, text, barcodes/QR codes from frames captured by AR
- **CoreLocation, Beacons:** augment content based on location of user or proximity to beacons
- **AV:** Overlay audio and video in your virtual scene

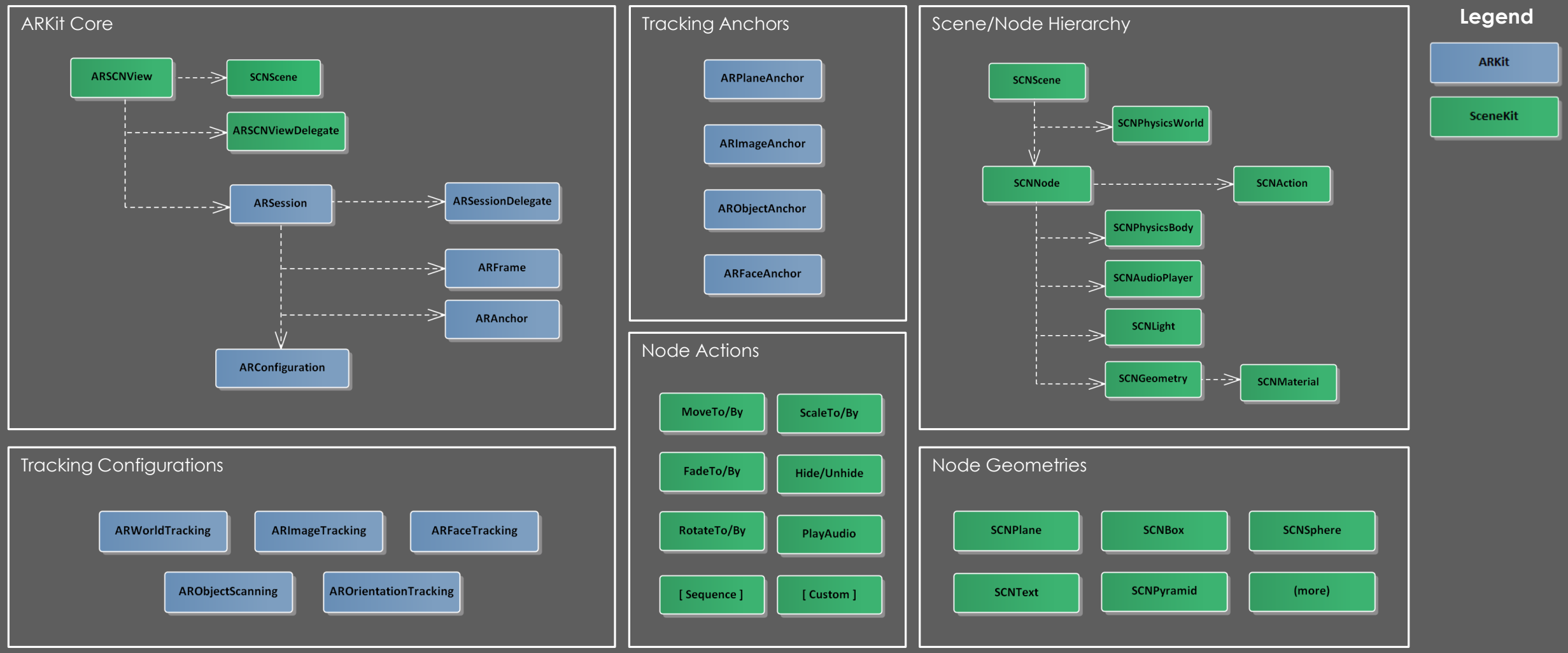
Xamarin/Microsoft:

- **SignalR:** realtime integration
- **Xamarin Forms:** EZ PZ UI
- **Cognitive Services:** Machine Learning, Artificial Intelligence

-= introduction to arkit =-

the framework

arkit+scenekit framework class structure (abridged)

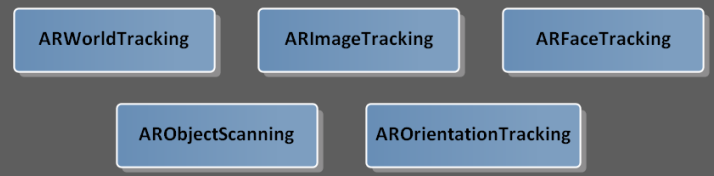


ARSession

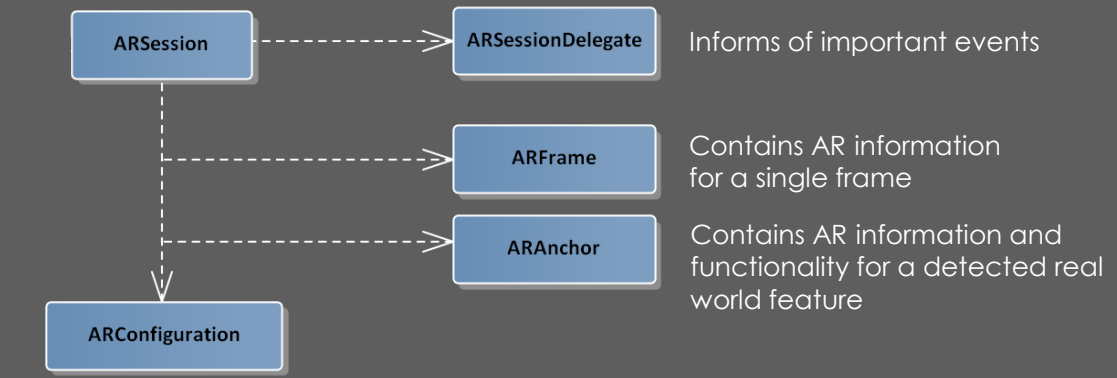
- ARKit is session based, and can be controlled using the `ARSession` class
- You begin an `ARSession` using the `Run` method, passing an `ARConfiguration`

- Once running, there are two primary ways to access session data:
  - Provide an `ARSessionDelegate` and implement `OnUpdate` to perform processing or every frame
  - At any time (e.g. in response to user input) access the `CurrentFrame` property of the `ARSession`

- **Xamarin mantra for health and happiness:** *"Touch an `ARFrame`, Dispose an `ARFrame`"*



Supported `ARConfigurations`



Core AR Class responsibilities

ARSession continued

- You can and should modify your session config while running to enable/disable features as needed. Calling **`Start`** again with an updated **`ARConfiguration`** while running will cause ARKit to use the new settings
  - e.g. disable surface detection once you have identified the ones you need – save battery and processing
  - e.g. swap out a set of reference images for another set during image detection based on the kind of activity the user is performing – reduce processing requirements

```
public void DisablePlaneTracking()
{
    // get the running config
    var config = SCNView.Session.Configuration as ARWorldTrackingConfiguration;

    // turn off plane detection
    config.PlaneDetection = ARPlaneDetection.None;

    // tell ARKit to use the new config
    SCNView.Session.Run(config);
}
```

- When AR tracking is interrupted, ARKit will call **`WasInterrupted`** on its **`ARSessionDelegate`**.

  If your app is sensitive to breaks in tracking, you can use this notice to reset the worldmap completely.

```
public override void WasInterrupted(ARSession session)
{
    // pause session
    session.Pause();

    // specify that existing tracking should be discarded
    var opts = ARSessionRunOptions.RemoveExistingAnchors
        | ARSessionRunOptions.ResetTracking;

    // restart session
    session.Run(session.Configuration, opts);
}
```

World Tracking

- World Tracking is configured by `ARWorldTrackingConfiguration` and using the magic of Visual Inertial Odometry gives you 6DOF positioning. Key configuration items:
  - `PlaneDetection` – Whether to detect surfaces that are horizontal, vertical, both or none
  - `WorldAlignment` – How co-ordinates will be determined/specified

Gravity
Y axis is up and down
X axis is left and right from origin
Z axis is forward and back from origin

Gravity + Heading
Y axis is up and down
X axis is east and west
Z axis is north and south

Camera
The camera transform (position and rotation) is considered to be the origin at all times.

- When running in world tracking mode:
  - ARKit will continuously monitor your position/heading based on device and video input
  - ARKit will detect features and surfaces according to configuration, and notify its delegate for key events
  - ARKit will let you know if it is unable to accurately track and why
- **World Tracking mantra for health and happiness**: "*ARKit in the dark would be boring anyway*" Accuracy suffers in poor lighting and when the video input contains mostly plain surfaces

Demo – World Tracking

- Basic setup of an ARKit session
- Observe how ARKit tracks us in an otherwise empty scene

Tips – World Tracking

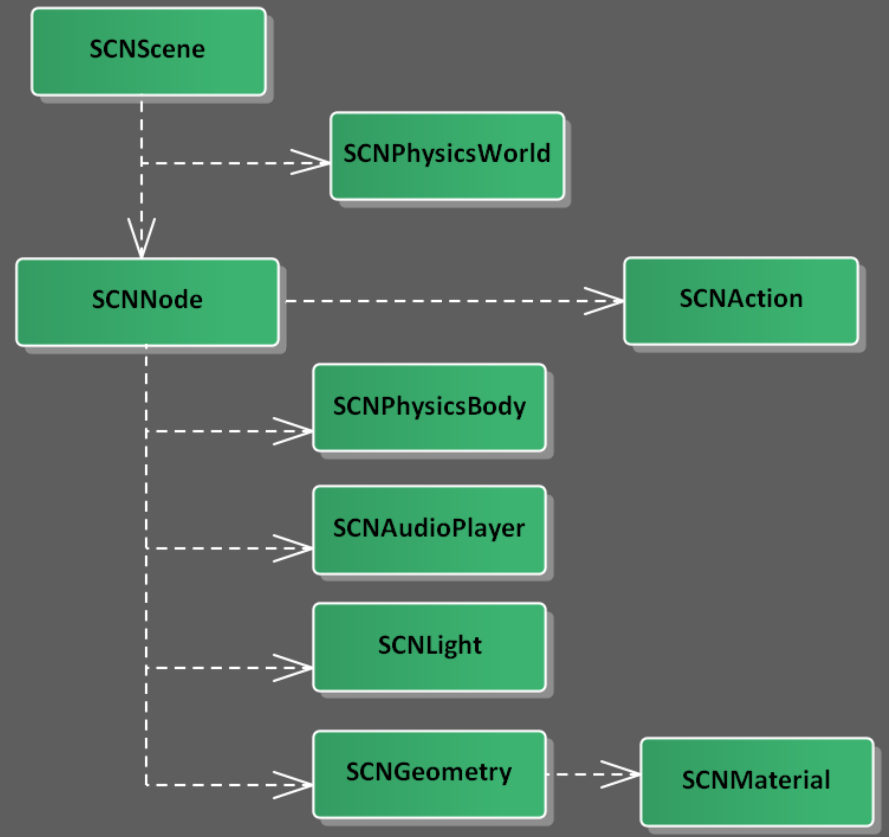
- **AR Presentation mantra for health and happiness**:
  "Always brush up on your 3D/Matrix math, if you can"

ARKit vs Renderers (SceneKit) vs Developer

- ARKit does not handle any virtual content – this is the responsibility of a renderer
- Features detected by ARKit are used automatically by the renderer:
  - As ARKit tracks your transform from origin, SceneKit adjusts the camera accordingly - `ARSCNView.PointOfView` property
  - As ARKit tracks lighting, SceneKit controls scene illumination
- Features detected by ARKit are used by you, the developer
  - Use surfaces detected by ARKit to constrain or orient your simulation
  - Use local co-ordinate systems of detected features to simplify positioning of virtual content

Virtual Content in SceneKit

- An `SCNScene` is the root of a 'world' and is viewed by an `SCNCamera`. When using `ARSCNView`, SceneKit uses AR inputs to control the camera
- An `SCNScene RootNode` is a hierarchy of `SCNNode`s,
  - A `SCNNode` has an name and a transform in 3D-space
  - A `SCNNode` may contain child `SCNNode`s
  - All child nodes are positioned relative to parent
- An `SCNNode` can contain components with behaviour
  - `SCNGeometry` – allows node to be rendered
  - `SCNPhysicsBody` – allows node to participate in physics
  - `SCNAudioPlayer` – allows the node to emit audio
  - `SCNLight` – allows the node to emit light

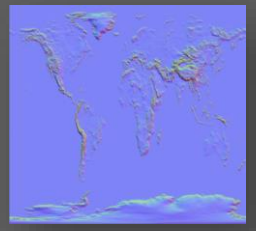


Virtual Content Rendering

- SceneKit has built in 3D shape primitives, or you can specify/load your own models
- Four lighting models of varying complexity, including Physical-Based Rendering (PBR)
- Sophisticated material subsystem that can be applied to get highly realistic results


diffuse

+

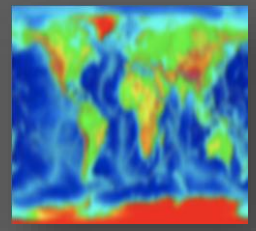

normal

+

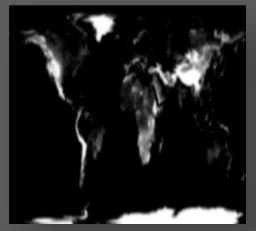

occlusion

+

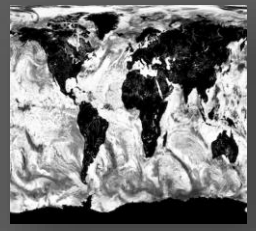

emission


self-illumination

+


displacement

+


metalness

+


roughness

=

Super realistic looking globe floating in the spare room

Virtual Content Logic

- Run an `SCNAction` on an `SCNNode` to declaratively specify behaviour and timing
  - Many actions included in the framework, or you can define your own
    - Translation: `.MoveTo/By, .RotateTo/By, .ScaleTo/By`
    - Appearance: `.FadeTo/By, .Hide/Unhide`
    - Audio: `.PlayAudio`
    - Control: `.Sequence, .Group, .Repeat, .Delay`
    - Abitrary: `.RunAction(n => )`
  - Control/Arbitrary actions allow you to compose complex behaviours
- You can perfom logic on a per-frame basis at the scene level
  - `ARSCNViewDelegate` calls `Update` and gives you the time since the last call
- Using `async` adds another dimension to logic, simplifying otherwise complex scenarios.

Demo – Virtual Content

- Load a SceneKit scene straight in, ~no code~
- Use SceneKit assets as ‘prefabs’
- Use SCNActions to animate virtual content in the 3D world
- Try and fail to place virtual objects realistically

Tips – Virtual Content

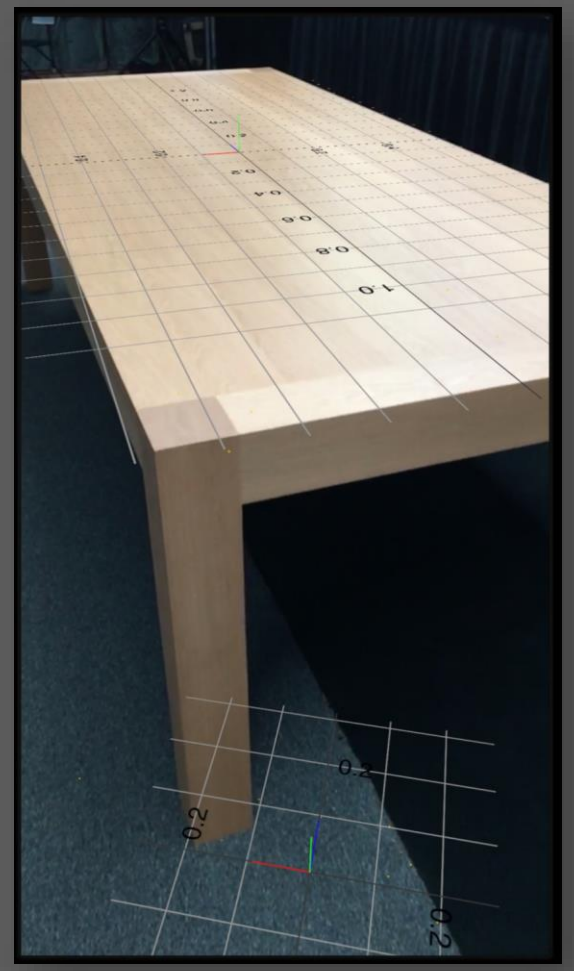
- **AR Presentation mantra for health and happiness**:
  "Always brush up on your 3D/Matrix math, if you can"

- Add objects as children of anchor nodes rather than the root node when it's easier to think in those terms

- Cloning nodes is super useful but note that clones share geometry/materials – so copy these individually if you want to modify a clone independently.

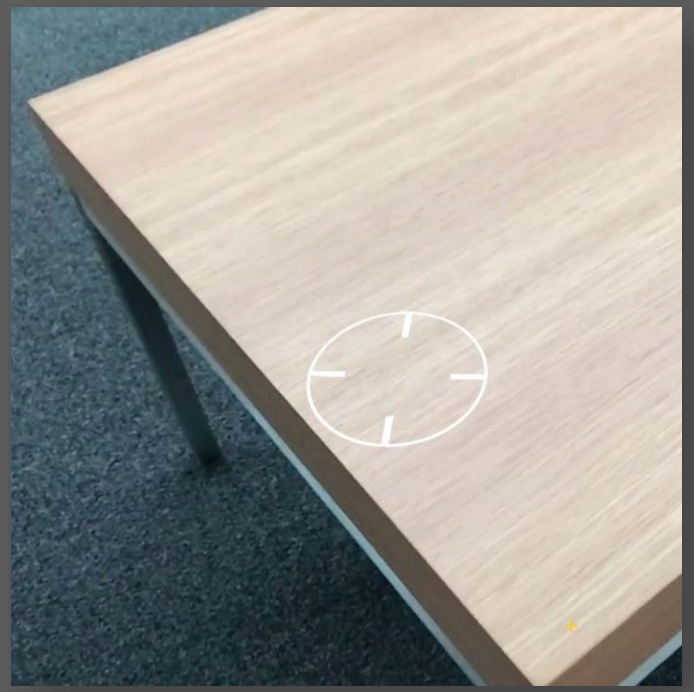
Plane Detection

- Convincingly placing virtual content in your AR world is aided by an understanding of physical surfaces that are present
- As of v1.5, ARKit detects both horizontal and vertical surfaces
- As plane detection runs, ARKit progressively builds up a world map and notifies you as its understanding increases using delegate callbacks:
  - `OnNodeAddedForAnchor` – new plane detected
  - `OnNodeUpdatedForAnchor` – understanding of existing plane improved
  - `OnNodeRemovedForAnchor` – existing plane no longer valid
- Detected surfaces are exposed to you as `ARPlaneAnchor` instances, providing information such as orientation and extents, and SceneKit provides a corresponding (invisible) `SCNNode`



AR Hit Testing

- ARKit provides support for hit testing **real world** features, exposed by the `ARFrame.HitTest(CGPoint, ARHitTestResultType)`
  - **Hit Testing mantra for health and happiness:**
    "ARKit hit tests using normalised co-ordinates (0.0-1.0), SceneKit hit tests using view co-ordinates"
- In my test scenes, performance seems to be good enough to hit test on every AR frame, YMMV
- ARKit can hit test against:
  - Detected plane bounds or geometry
  - Raw features
- Hit testing surfaces can be a good way to request input from the user (e.g. ask the user point to the surface you want to focus on, highlight the surface matching the hit test)



Demo – Plane Detection

- Detect horizontal and vertical planes
- Use hit testing to detect when the user points at detected planes
- Use planes to help place virtual content more realistically
  (but still fail kind of because I'm bad at math)

Tips – Plane Detection

- **AR Presentation mantra for health and happiness**:
  "Always brush up on your 3D/Matrix math, if you can"

- Encourage users to work under optimal conditions (good lighting, etc)

- Turn off plane detection once you have the surfaces you need
  - reduces processing time and battery drain,
  - more stable experience for users

- Use debug features, but sparingly
  (the internet says accuracy is worse while they're on)

Image Detection

- ARKit can detect of images by comparing input to a set of `ARReferenceImage`s that you provide
- Similar to plane detection, ARKit provides an `ARImageAnchor` when a reference image is detected
- ARKit also supports image **tracking**, which allows you to keep track of the movement of a detected image in 3D space.

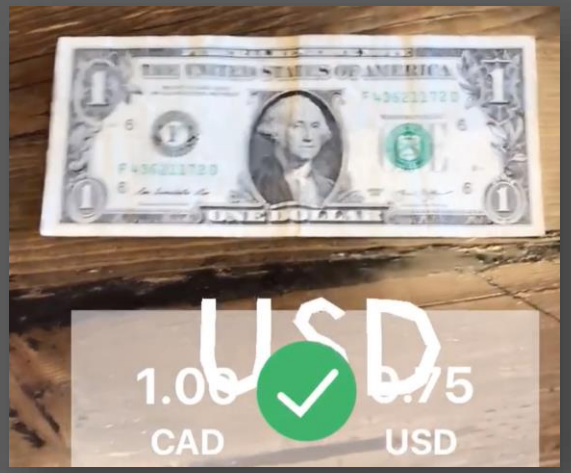


Image Detection
Recognised currency and provided additional information about it.

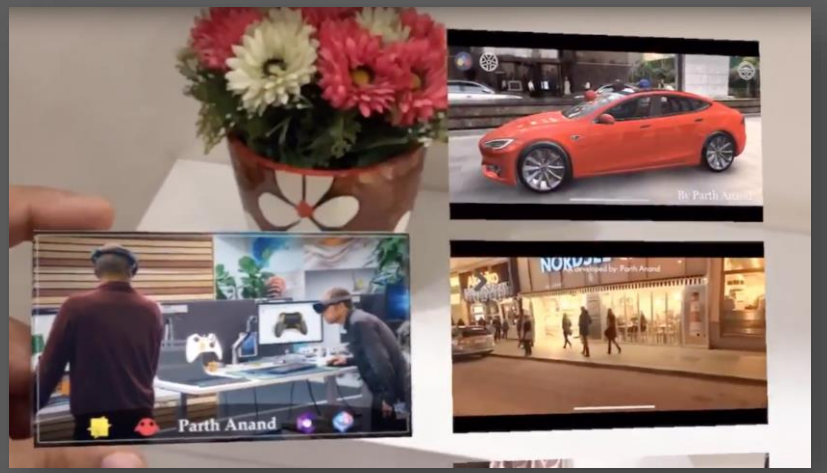


Image Tracking
Recognised business card and overlaid addition content to the right of it. As the business card or camera changes position, the additional content remains anchored to the business card.

Demo – Image Detection

- Detect reference images and place virtual content when we do
- Use *SceneKit* hit testing to detect when the user is pointing at the new content

Tips – Image Detection

- ARKit2 has a dedicated `ARImageTrackingConfiguration` configuration type that can be used for image detection if you don't need full world tracking powers – more efficient on battery life

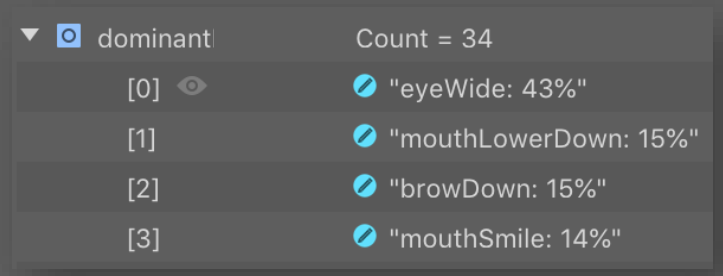
Face-based AR

- ARKit provides special features for performing face-oriented AR, initiated by using an `ARFaceTrackingConfiguration`:
  - Face detection
  - Face tracking
  - Expression tracking

- ARKit provides `ARFaceAnchor` s when a face is detected, which includes the detected geometry.

- `ARFaceAnchors` also include a `BlendShapes` property, which specifies the intensity of various aspects of face expression.

Detected face meshes are cool but kinda creepy if you display them raw



It's up to the developer to interpret the readings provided by `BlendShapes`

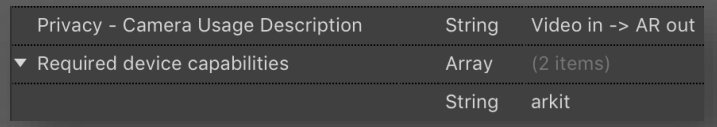
Demo – Face AR

- Detect a face
- Track a face
- Identify expressions

wrapping up

quick start ur Xamarin arkit+scenekit app

- Add Camera Usage reason and *arkit* device requirement to Info.plist
- Add an **`ARSCNView`** to a **`UIViewController`**'s root view
- Configure the appropriate **`ARSessionConfiguration`** for the kind of AR you want to perform
  - E.g. for **`ARWorldTrackingConfiguration`**, specify your world alignment and whether you are interested in horizontal and/or vertical plane detection
  - E.g. For **`ARImageTrackingConfiguration`**, specify the reference images that should be detected
- Perform any SceneKit setup on your **`SCNView`**'s **`Scene`**.
- When ready, call **`Run(config)`** on your **`SCNView`**'s **`Session`** to begin AR tracking.
- Respond to user input or callbacks raised by **`ARSCNViewDelegate`** and **`ARSessionDelegate`**



Prevent immediate crashes with this one crazy tip!

```
var config = new ARWorldTrackingConfiguration
{
    PlaneDetection = ARPlaneDetection.Horizontal
};

SCNView.Session.Run(config);
```

This ARKit session will continuously, automatically detect and track horizontal surfaces

```
public void OnFrameUpdate(ARSession session, ARFrame frame)
{
    // do something for every AR frame if you want
    // but don't forget to dispose it when you're done
    using (frame)
        DoStuff(frame);
}
```

You can receive callbacks for every AR frame received, or respond to specific events (e.g. plane detected/updated)

remember your mantras

- **Xamarin mantra for health and happiness:**
  "Touch an `ARFrame`*, Dispose an* `ARFrame`*"*

- **World Tracking mantra for health and happiness**:
  "AR in the dark would be boring anyway"

- **AR Presentation mantra for health and happiness**:
  "Always brush up on your 3D/Matrix math, if you can"

- **Hit Testing mantra for health and happiness:**
  "ARKit hit tests using normalised coordinates (0.0-1.0), SceneKit hit tests using view coordinates"

arkit things I didn't cover

- **3D Object Detection**
  Similar to the image detection demonstrated, it's possible to use ARKit to detect 3D objects in the real world based off reference models. Unlike 2D images, these cannot (currently?) be tracked

- **Scene Persistence**
  As of ARKit 2, it's possible to serialise AR session state – both the internal worldmap maintained by ARKIt and your virtual additions – and then restore it later

- **Multipeer/Shared Experience**
  As of ARKit 2, it's possible to initiate a shared AR session where multiple users can participate in a single virtual scene.

- **Probably a few other things..**

useful resources

- **Larry O'Brien's Xamarin/ARKit blog posts**
  https://blog.xamarin.com/augment-reality-xamarin-ios-11/
  https://blog.xamarin.com/exploring-new-ios-12-arkit-capabilities-with-xamarin/

- **Microsoft ARKit Docs**
  https://docs.microsoft.com/en-us/xamarin/ios/platform/introduction-to-ios11/arkit/
  https://docs.microsoft.com/en-us/xamarin/ios/platform/introduction-to-ios12/arkit2

- **Ray Wenderlich ARKit Articles, and "ARKit2 By Tutorials"** (~$40AUD)
  (these are all Swift-based but easy enough to apply)
  https://www.raywenderlich.com/library?domain_ids%5B%5D=1&q=arkit&sort_order=relevance
  https://store.raywenderlich.com/products/arkit-by-tutorials

- **#MadeWithARKit on Twitter**
  https://twitter.com/hashtag/madewitharkit?src=hash

questions / comments / concerns