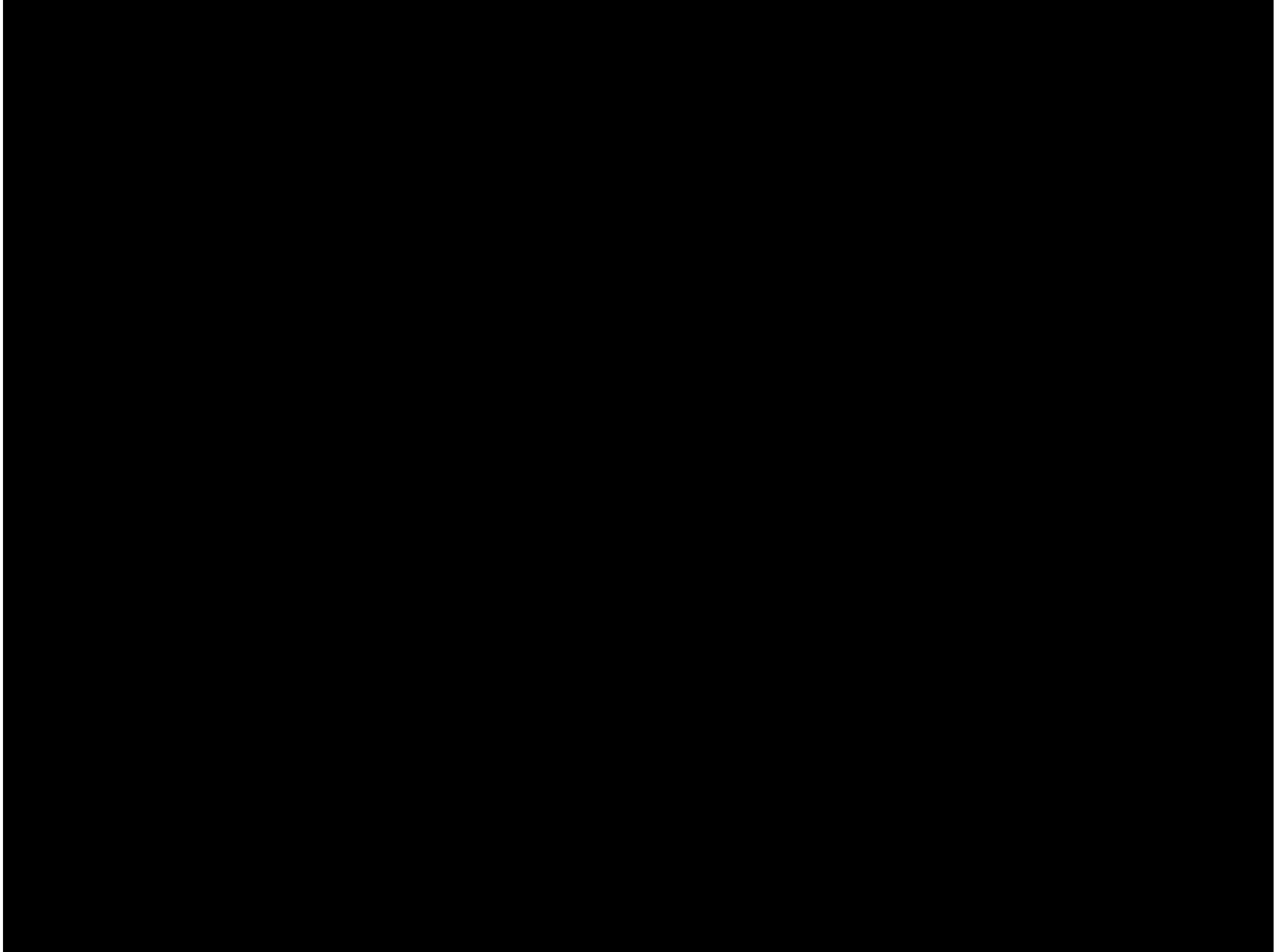
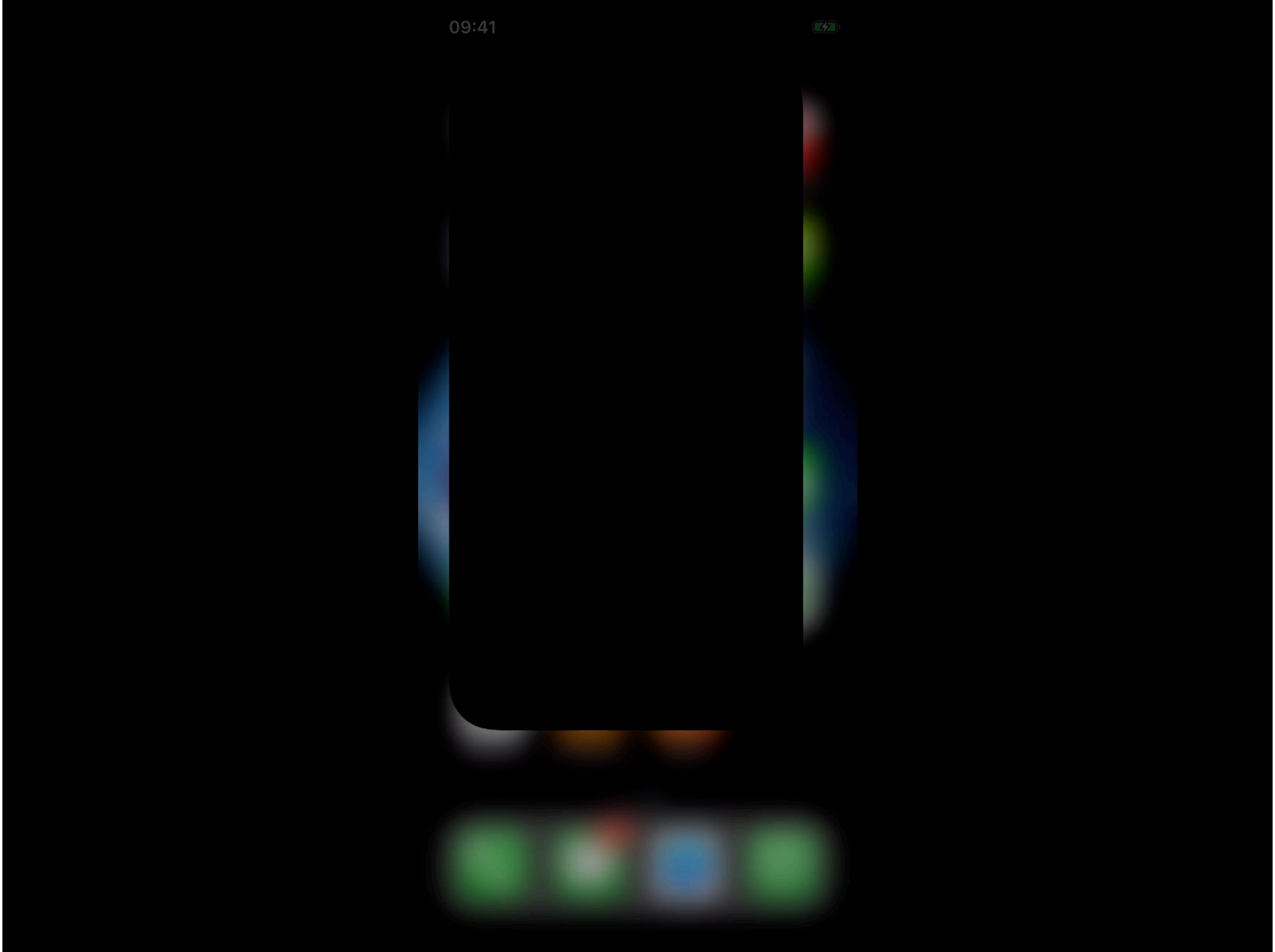



AR
ML
9:41 AM
100%
Mail
Tuesday
12
Calendar
Photos
Camera
Maps
Clock
Weather
News
Home
Notes
Stocks
Reminders
TV
App Store
iTunes Store
iBooks
Health
Wallet
Settings


AR
ML


09:41

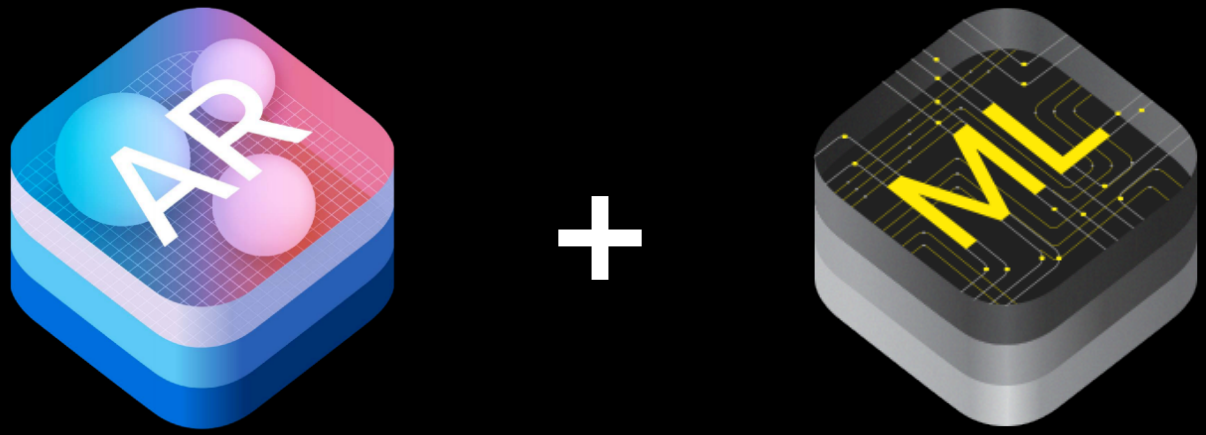
Agenda

- AR
  - What
  - Why
  - How - iOS ARKit
    - APIs
    - Advantages
    - Prerequisites
    - Structure
    - Classes and Relationships
    - Development Flow
    - Demo

Agenda

- ML
  - What
  - Why
  - How - iOS CoreML
    - APIs
    - Advantages
    - Prerequisites
    - Structure
    - Classes and Relationships
    - Development Flow
    - Demo

Agenda

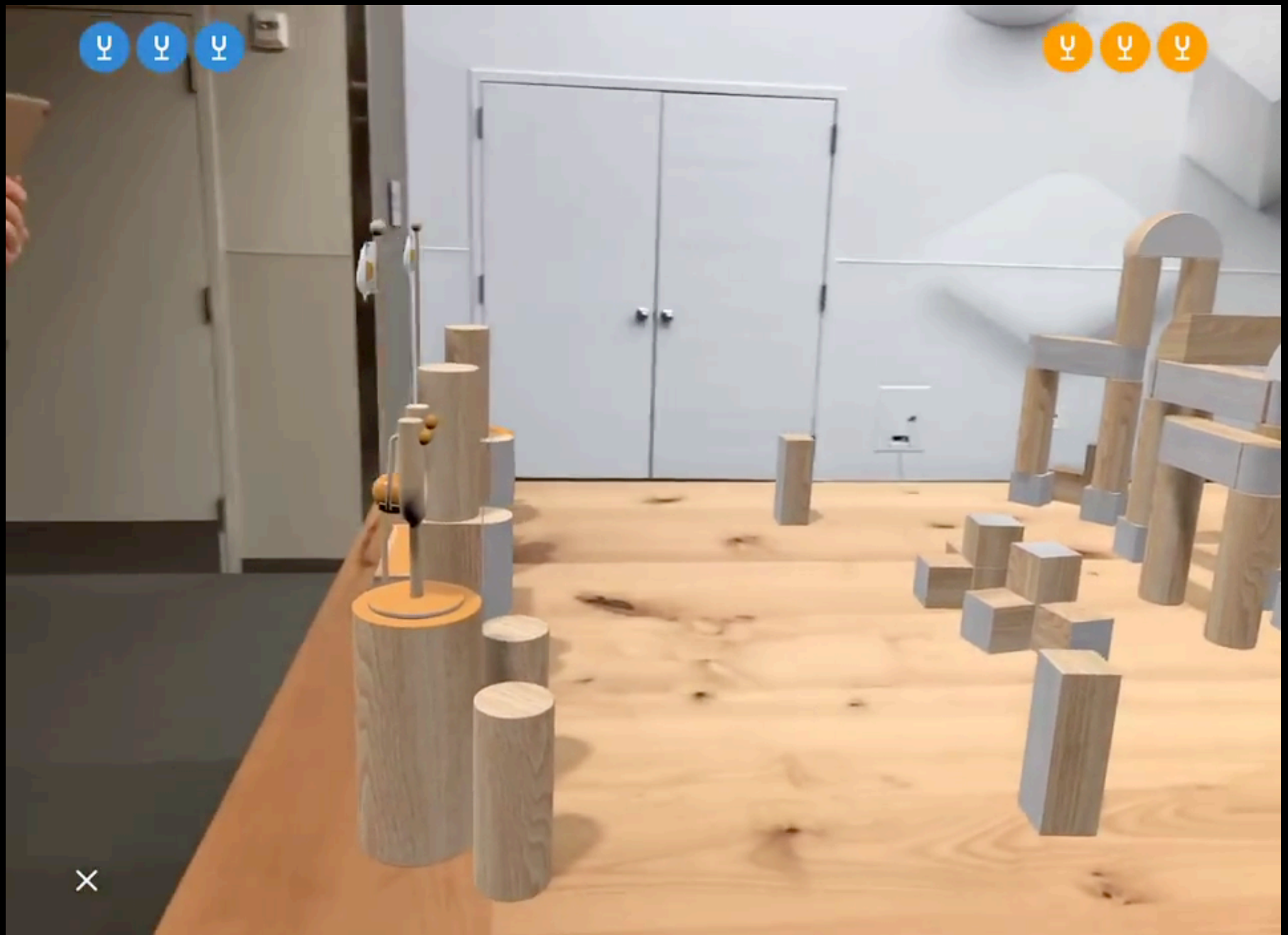


- Development Flow
- Demo


AR

AR - Why

AR - Why




AR

AR - How


AR


11


APL0598 33950176
GKBC40HI 1213
A9
K3PE7E700F - XGC2
3SC0091 1222

APIs

Computer Vision

Triangulation

Scene Understanding

Surface Estimation

SLAM

Feature Detection

Sensor Fusion

Augmented Reality

Camera Intrinsics

Camera Calibration

Light Estimation

Bundle Adjustment

Visual-inertial Navigation

Feature Matching

Nonlinear Optimization

Optimal Correction

Advantages

- Easy to use
- Mobile efficient
- Easy to customise

Prerequisites

Structure

Application

Processing

AVFoundation

CoreMotion

Rendering

Classes and Relationships

ARConfiguration

ARSession

ARSCNView → SCNScene → SCNNode

SCNGeometry

SCNAction

SCNMaterial

Classes and Relationships

ARConfiguration

ARSession

ARSCNView

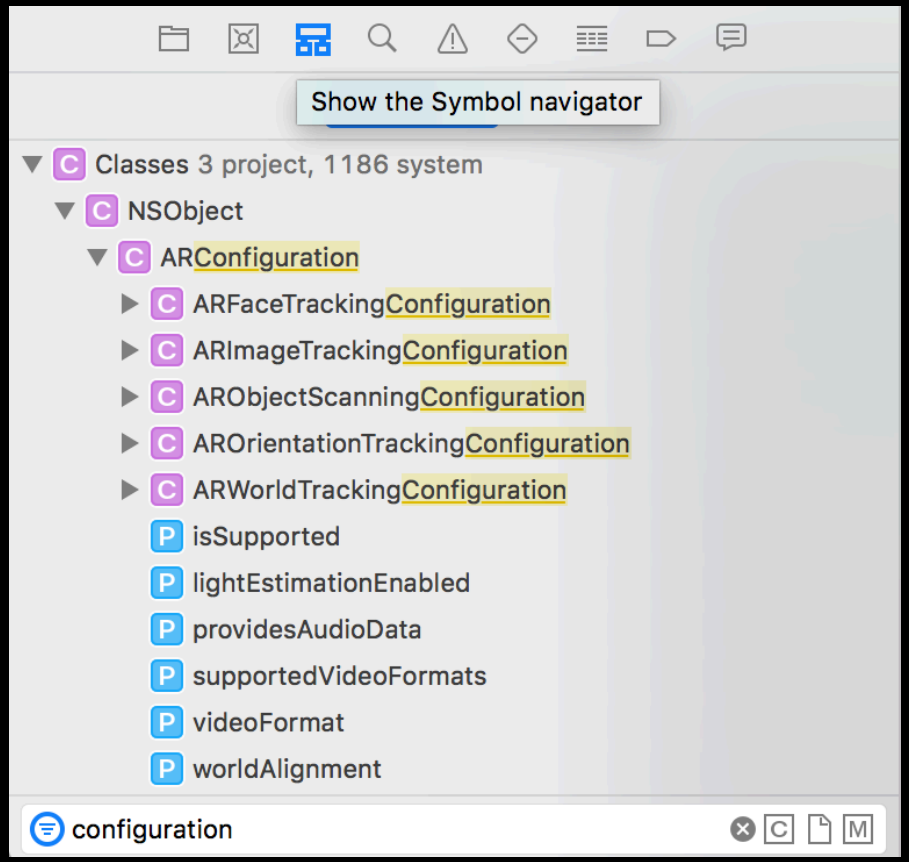
SCNScene

SCNNode

SCNGeometry

SCNAction

SCNMaterial



Classes and Relationships

ARConfiguration

ARSessionDelegate
Or currentFrame

ARSession

ARFrame

ARAnchor

ARSCNView

SCNScene

SCNNode

SCNGeometry

SCNAction

SCNMaterial


АЗБУКА

Development Flow

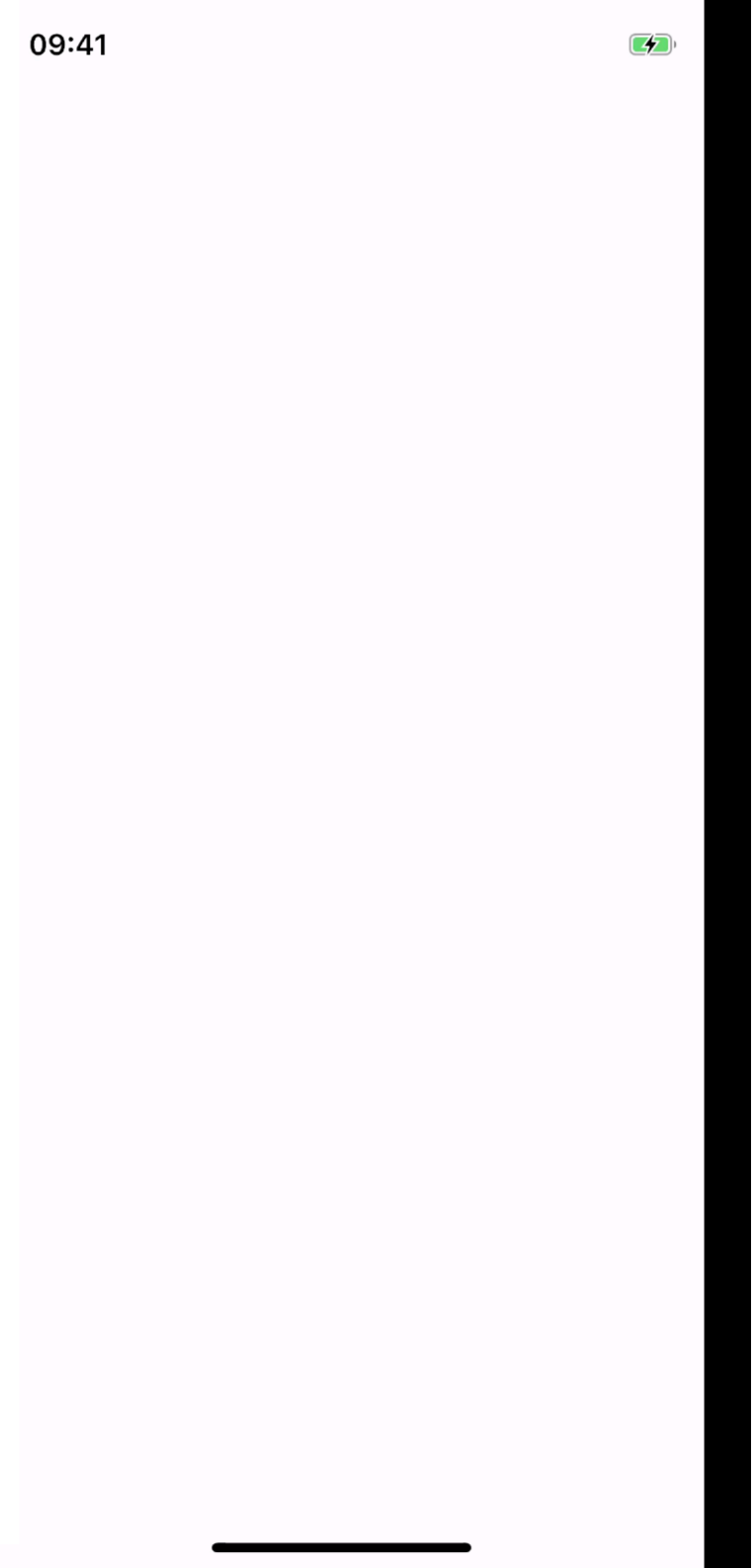
```
assert(ARWorldTrackingConfiguration.isSupported,
"ARKit is not available on this device. For apps
that require ARKit for core functionality, use
the `arkit` key in the key in the
`UIRequiredDeviceCapabilities` section of the
Info.plist to prevent the app from installing.
(If the app can't be installed, this error can't
be triggered in a production scenario.) In apps
where AR is an additive feature, use
`isSupported` to determine whether to show UI
for launching AR experiences.")
```

Development Flow

Key	Type	Value
▼ Required device capabilities	Array	(2 items)
Item 0	String	armv7
Item 1	String	arkit

Development Flow

```
@IBOutlet var sceneView: ARSCNView!
```



Development Flow

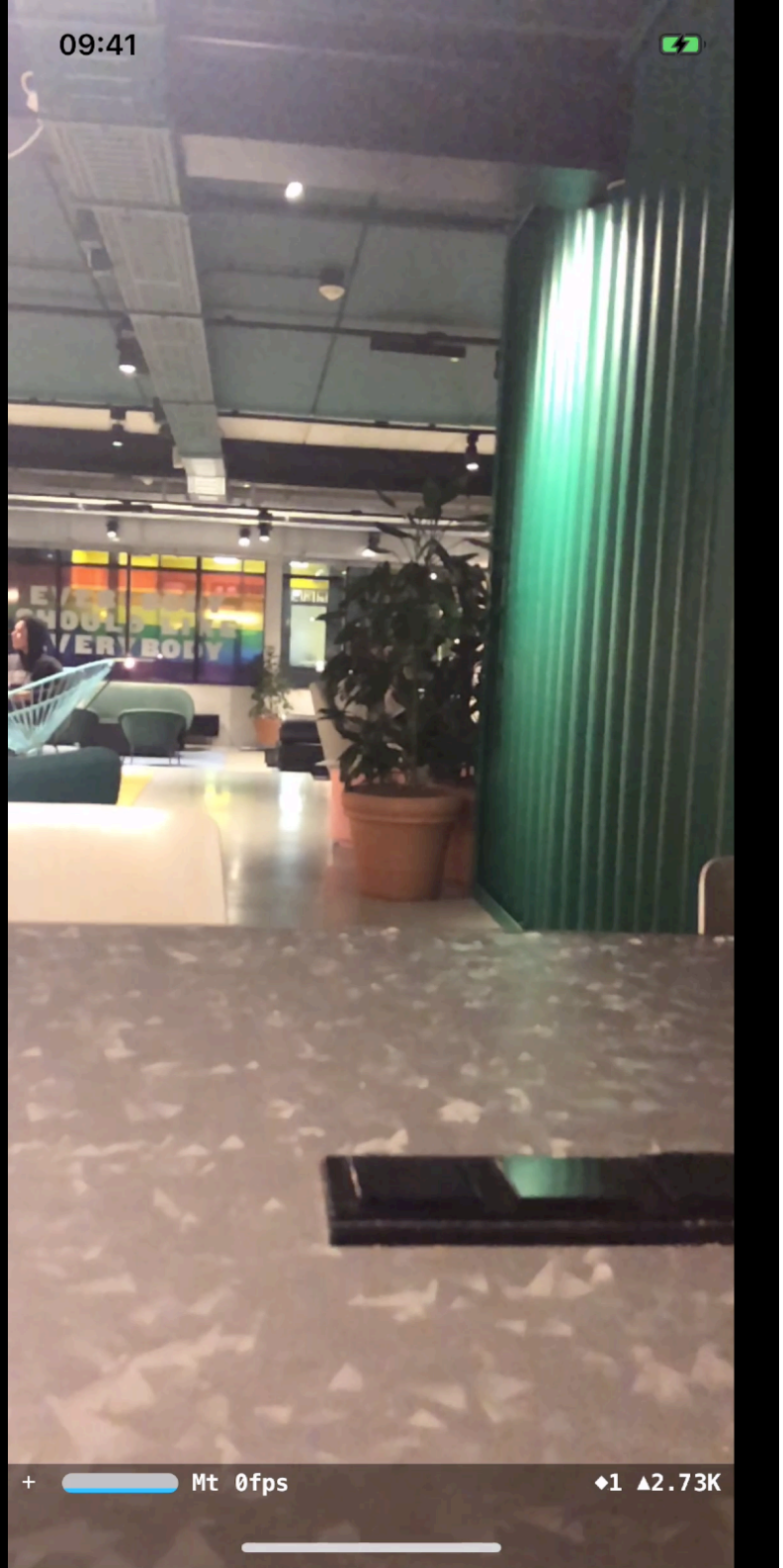
```
@IBOutlet var sceneView: ARSCNView!

let configuration =
ARWorldTrackingConfiguration()

sceneView.session.run(configuration)
```

Development Flow

```
@IBOutlet var sceneView: ARSCNView!

let configuration =
ARWorldTrackingConfiguration()
configuration.planeDetection = .horizontal

sceneView.session.run(configuration)
```

Development Flow

```
@IBOutlet var sceneView: ARSCNView!

let configuration =
ARWorldTrackingConfiguration()
configuration.planeDetection = .horizontal

sceneView.session.run(configuration)
sceneView.showsStatistics = true
sceneView.debugOptions  =
[ARSCNDebugOptions.showFeaturePoints]
```



Development Flow

```
@IBOutlet var sceneView: ARSCNView!

let configuration =
ARWorldTrackingConfiguration()
configuration.planeDetection = .horizontal

sceneView.session.run(configuration)
sceneView.showsStatistics = true
sceneView.debugOptions  =
[ARSCNDebugOptions.showFeaturePoints]

var chameleon = Chameleon()
sceneView.scene = chameleon
```

Development Flow

```
sceneView.delegate = self

func renderer(_ renderer:
SCNSceneRenderer, didAdd node:
SCNNode, for anchor: ARAnchor) {
  if anchor is ARPlaneAnchor {
    showToast("Plane anchor detected")
  }
}
```

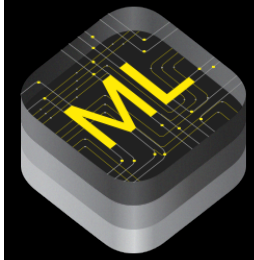
Development Flow

```
sceneView.delegate = self

func renderer(_ renderer:
SCNSceneRenderer, didAdd node:
SCNNode, for anchor: ARAnchor) {
  if chameleon.isVisible() { return }
  if anchor is ARPlaneAnchor {
    chameleon.setTransform(anchor.trans
    form)
    chameleon.show()
  }
}
```

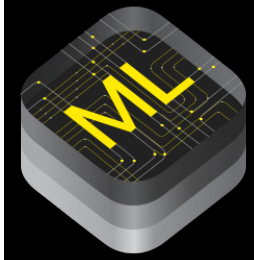
Development Flow

```
@IBAction func didPan(_ recognizer:
UIPanGestureRecognizer) {
  let location = recognizer.location(in:
  sceneView)
  let arHitTestResult =
  sceneView.hitTest(location,
  types: .existingPlane)
  if !arHitTestResult.isEmpty {
    let hit = arHitTestResult.first!
    chameleon.setTransform(hit.worldTransform)
    if recognizer.state == .ended {
      chameleon.reactToPositionChange(in:
      sceneView)
    }
  }
}
```


ML

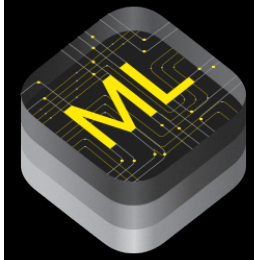


ML - Why

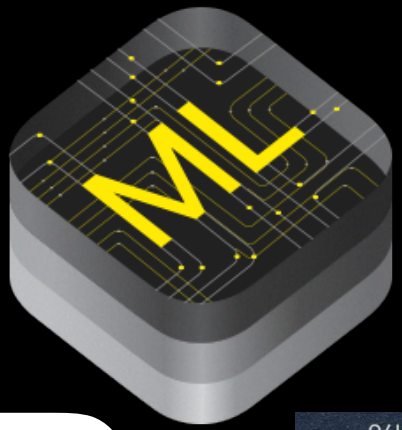

ML

ML - Why


9:41 AM
100%
Albums
People
Select
LIA
ALEX
Natalie
Andrew
Augusto
Beth
Dana
Kelly
Photos
Memories
Shared
Albums



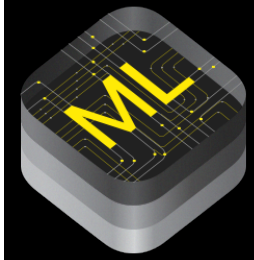
ML - How



10.13+

4.0+

11.0+



APIs

Real Time Image Recognition

Text Prediction

Entity Recognition

Sentiment Analysis

Handwriting Recognition

Style Transfer

Search Ranking

Machine Translation

Image Captioning

Personalization

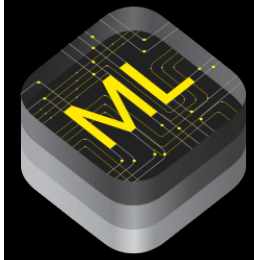
Face Detection

Emotion Detection

Speaker Identification

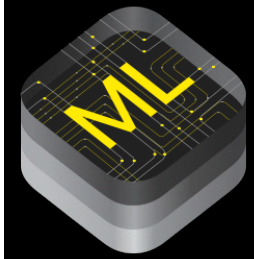
Music Tagging

Text Summarization



Advantages

- Run on device
  - Data privacy
  - Data and server cost
- Challenges
  - Correctness
  - Performance
  - Energy efficiency



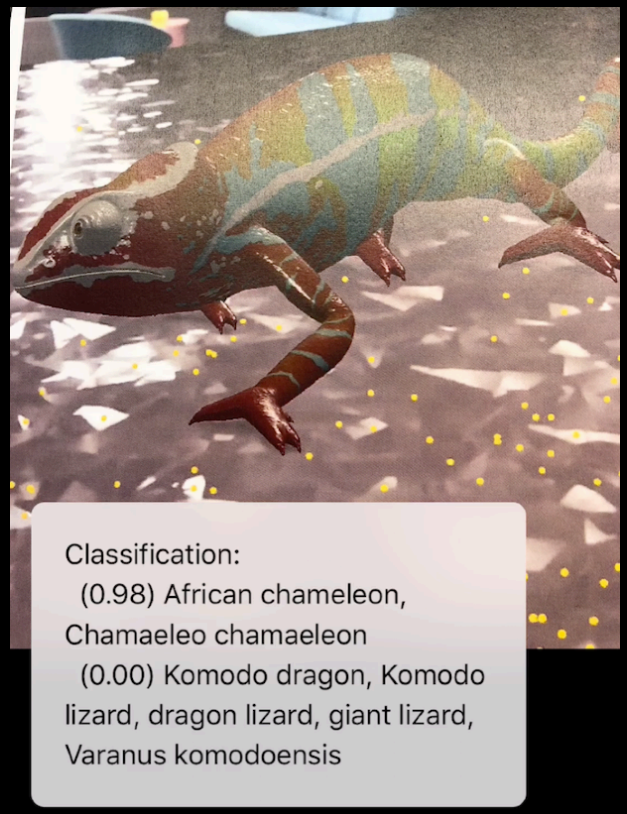
Prerequisites

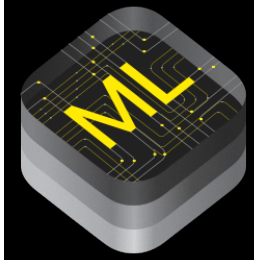
Live Data

Infer ↓

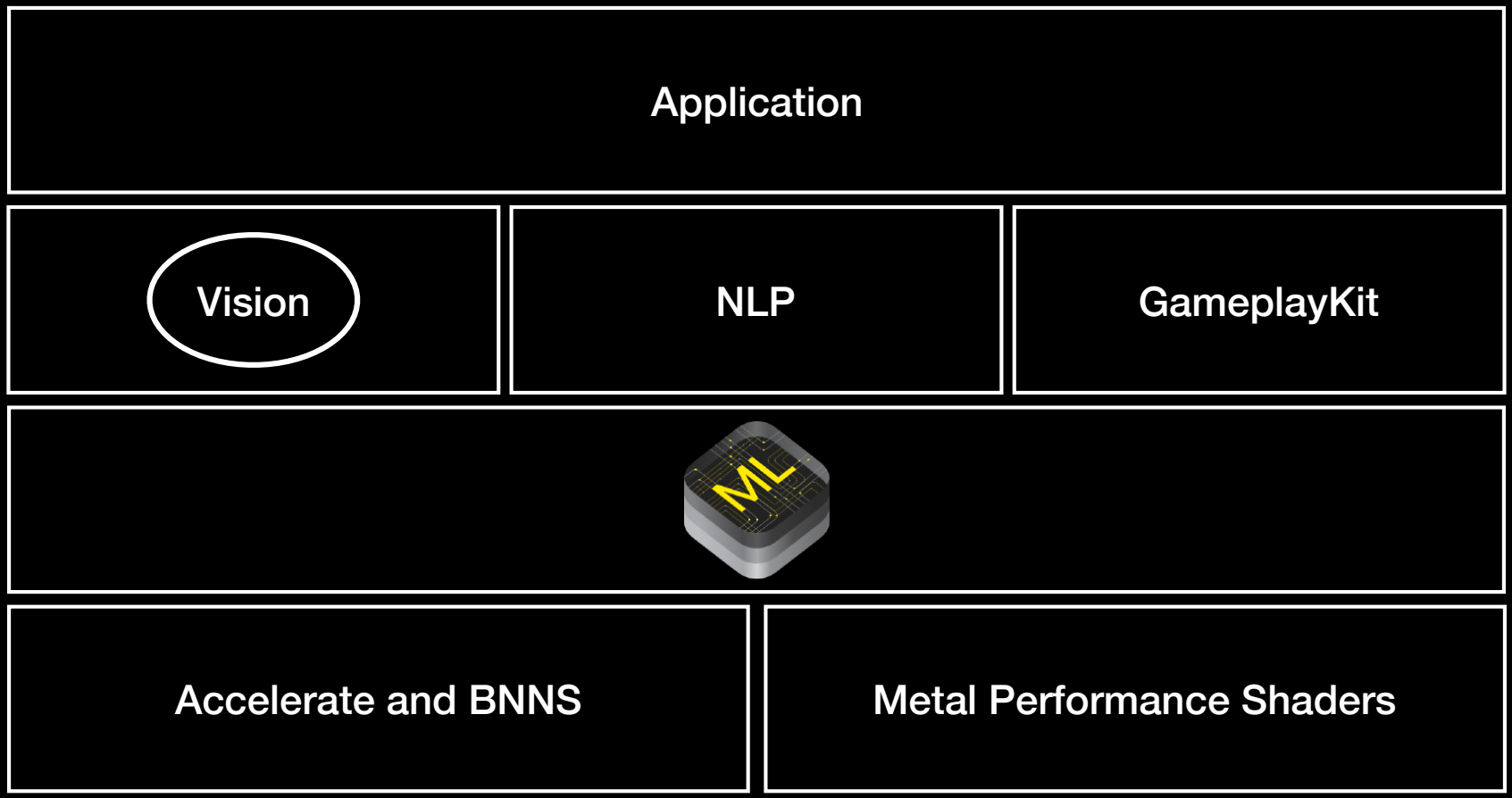
Training Data
(Gather data and decode it)

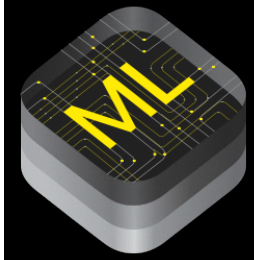
→



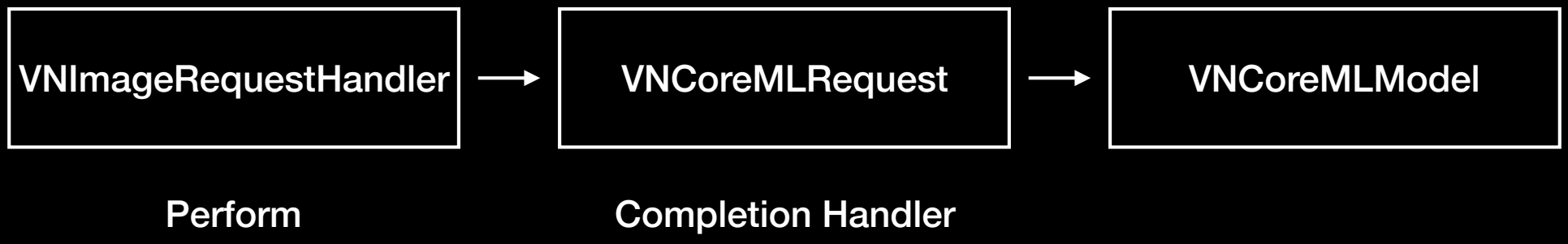


Structure

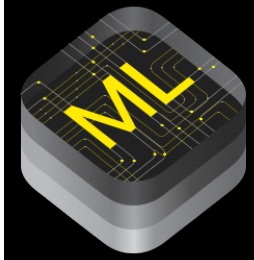

Application
Vision
NLP
GameplayKit
Accelerate and BNNS
Metal Performance Shaders



Classes and Relationships

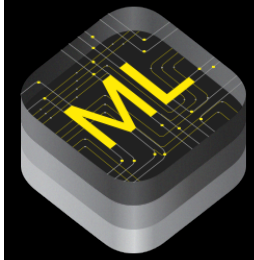



АЗБУКА



Development Flow

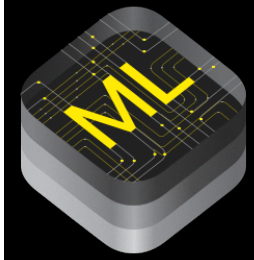
Drag and drop



Development Flow

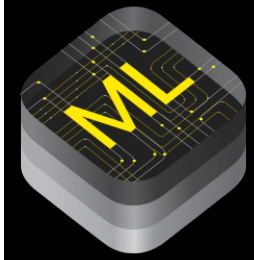
Machine Learning Model

Name	MobileNet
Type	Neural Network Classifier
Size	17,1 MB
Author	Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, Hartwig Adam
Description	MobileNets are based on a streamlined architecture that have depth-wise separable convolutions to build light weight deep neural networks. Trained on ImageNet with categories such as trees, animals, food, vehicles, person etc. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications https://github.com/shicai/MobileNet-Caffe
License	Apache License. Version 2.0 http://www.apache.org/licenses/LICENSE-2.0



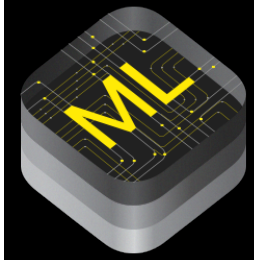
Development Flow

```
func updateClassifications(for image: UIImage) {
  let orientation =
  CGImagePropertyOrientation(image.imageOrientation)
  guard let ciImage = CIImage(image: image) else {
    fatalError("Unable to create \(CIImage.self) from \
    (image).")
  }
  let handler = VNImageRequestHandler(ciImage: ciImage,
  orientation: orientation)
  do {
    try handler.perform([self.classificationRequest])
  } catch {
    print("Failed to perform classification.\n\
    (error.localizedDescription)")
  }
}
```



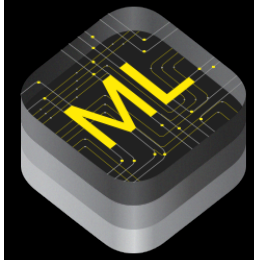
Development Flow

```
lazy var classificationRequest: VNCoreMLRequest = {
  do {
    let model = try VNCoreMLModel(for: MobileNet().model)
    let request = VNCoreMLRequest(model: model,
    completionHandler: { [weak self] request, error in
      self?.processClassifications(for: request, error: error)
    })
    return request
  } catch {
    fatalError("Failed to load Vision ML model: \(error)")
  }
}()
```

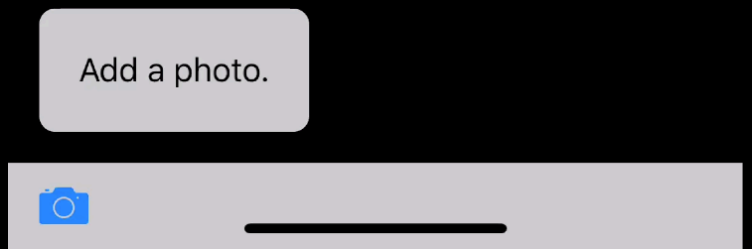


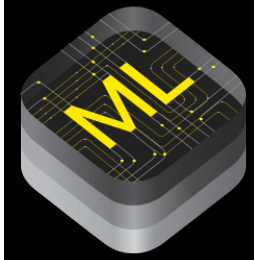
Development Flow

```
func processClassifications(for request: VNRequest, error: Error?) {
  guard let results = request.results else {
    self.classificationLabel.text = "Unable to classify image.\n\
    (error!.localizedDescription)"
    return
  }
  let classifications = results as! [VNClassificationObservation]
  if classifications.isEmpty {
    self.classificationLabel.text = "Nothing recognized."
  } else {
    let topClassifications = classifications.prefix(2)
    let descriptions = topClassifications.map { classification in
      return String(format: "  (%.2f) %@", classification.confidence,
      classification.identifier)
    }
    self.classificationLabel.text = "Classification:\n" +
    descriptions.joined(separator: "\n")
  }
}
```



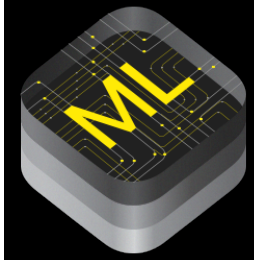
Development Flow





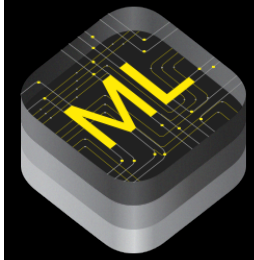
Development Flow

```
lazy var classificationRequest:
VNCoreMLRequest = {
  do {
    let model = try VNCoreMLModel(for:
    MobileNet().model)
    let request = VNCoreMLRequest(model:
    model, completionHandler: { [weak self]
    request, error in
      self?.processClassifications(for:
      request, error: error)
    })
    request.imageCropAndScaleOption
    = .centerCrop
    return request
  } catch {
    fatalError("Failed to load Vision ML
    model: \(error)")
  }
}()
```



Development Flow

```
lazy var classificationRequest:
VNCoreMLRequest = {
  do {
    let model = try VNCoreMLModel(for:
    Inceptionv3().model)
    let request = VNCoreMLRequest(model:
    model, completionHandler: { [weak self]
    request, error in
      self?.processClassifications(for:
      request, error: error)
    })
    request.imageCropAndScaleOption
    = .centerCrop
    return request
  } catch {
    fatalError("Failed to load Vision ML
    model: \(error)")
  }
}()
```



Development Flow

Machine Learning Model

Name	MobileNet
Type	Neural Network Classifier
Size	17,1 MB
Author	Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, Hartwig Adam
Description	MobileNets are based on a streamlined architecture that have depth-wise separable convolutions to build light weight deep neural networks. Trained on ImageNet with categories such as trees, animals, food, vehicles, person etc. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications https://github.com/shicai/MobileNet-Caffe
License	Apache License. Version 2.0 http://www.apache.org/licenses/LICENSE-2.0

Model Class

MobileNet

Automatically generated Swift model class

Model Evaluation Parameters

Name	Type
▼ Inputs	
image	Image (Color 224 x 224)
▼ Outputs	
classLabelProbs	Dictionary (String → Double)
classLabel	String

Machine Learning Model

Name	Inceptionv3
Type	Neural Network Classifier
Size	94,7 MB
Author	Original Paper: Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, Zbigniew Wojna. Keras Implementation: François Chollet
Description	Detects the dominant objects present in an image from a set of 1000 categories such as trees, animals, food, vehicles, person etc. The top-5 error from the original publication is 5.6%.
License	MIT License. More information available at https://github.com/fchollet/keras/blob/master/LICENSE

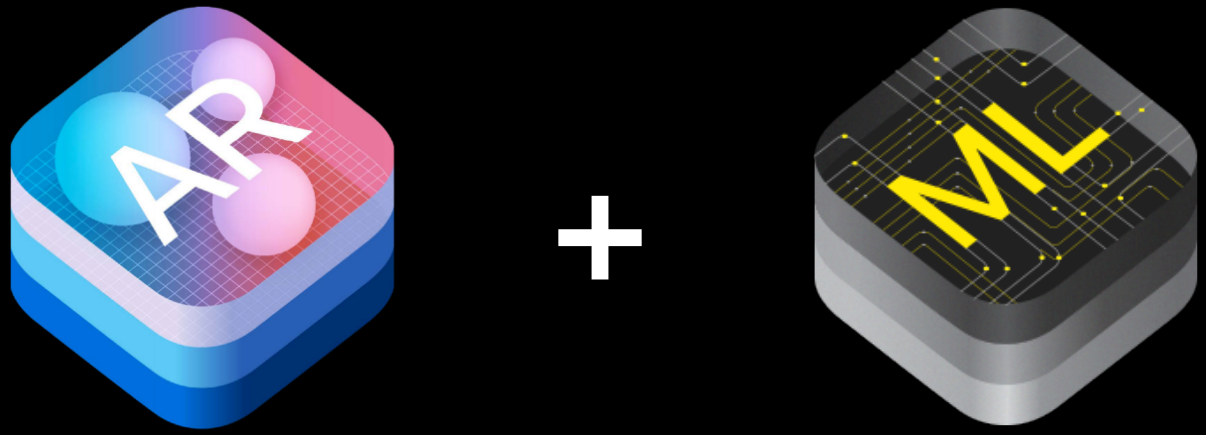
Model Class

Inceptionv3

Automatically generated Swift model class

Model Evaluation Parameters

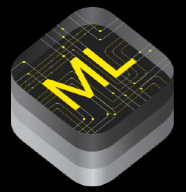
Name	Type
▼ Inputs	
image	Image (Color 299 x 299)
▼ Outputs	
classLabelProbs	Dictionary (String → Double)
classLabel	String


AR
ML


АЗБУКА

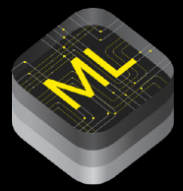
Development Flow

```
func updateCoreML() {
        let pixbuff : CVPixelBuffer? =
(sceneView.session.currentFrame?.capturedImage)
        if pixbuff == nil { return }
        let ciImage = CIImage(cvPixelBuffer: pixbuff!)
        let imageRequestHandler = VNImageRequestHandler(ciImage:
ciImage, options: [:])
        do {
            try imageRequestHandler.perform(self.visionRequests)
        } catch {
            print(error)
        }
    }
```



Development Flow

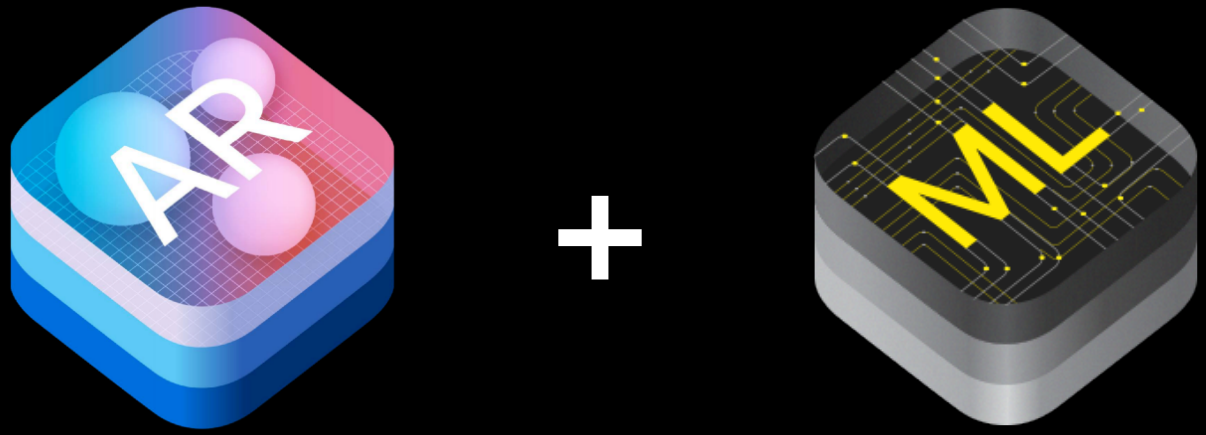
```
@objc func handleTap(gestureRecognizer: UITapGestureRecognizer) {
        let screenCenter : CGPoint = CGPoint(x:
self.sceneView.bounds.midX, y: self.sceneView.bounds.midY)
        let arHitTestResults : [ARHitTestResult] =
sceneView.hitTest(screenCenter, types: [.featurePoint])
       if let closestResult = arHitTestResults.first {
            let transform : matrix_float4x4 =
closestResult.worldTransform
            let node : SCNNode = createTextNode(latestPrediction)
            sceneView.scene.rootNode.addChildNode(node)
            node.transform = SCNMatrix4(transform)
        }
    }
```

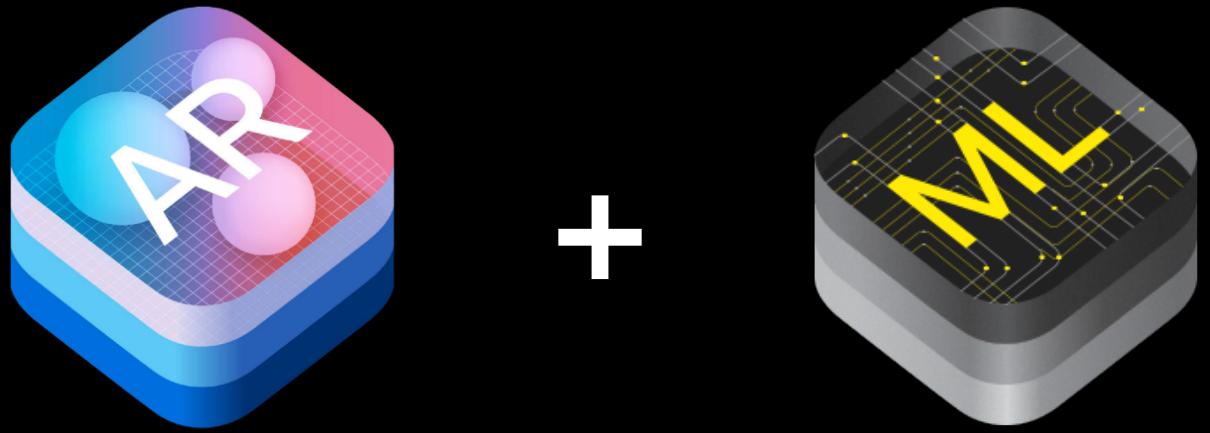



09:41
AR
ML

Recap

- AR
- ARKit
- ML
- CoreML
- ARKit + CoreML


AR
ML



MouradSidkyMourad@gmail.com
www.linkedin.com/in/mouradaly
www.linkedin.com/in/mouradsidky