• In AV and VE/VR the surrounding environment is virtual, in AR the surrounding environment is real

[Milgram & Kishino, 1994]
How to show it?

Display technologies

• Show virtual objects overlaying the real world in 3D space
• Head mounted
• Handheld

Where to show it?

Tracking (and registration) technologies

• To register virtual objects in 3D space and track user input
• Track the
  (a) scene
  (b) the user’s 6DOF viewpoint (head and/or eyes)
  (c) the user’s hands/body for input
  (d) input devices
Tracking

Simon Voelker, Philipp Wacker: iOS Application Development
World Tracking

• Back-facing camera

Face Tracking

• Front-facing camera
ARConfiguration & ARSession

- ARConfiguration defines which camera and tracking algorithms are being used
  - Configurations for: world, body, orientation, image, and face tracking
  - Object scanning

- ARSession manages the camera and motion processing

```swift
// Create a session configuration
let configuration = ARWorldTrackingConfiguration()

// Run the view's session
sceneView.session.run(configuration)
```

// Pause the view's session
sceneView.session.pause()
ARWorldTrackingConfiguration

• Define what to look for in the scene
  • Plane detection
  • Image detection
  • Object detection

• Create a world map
  • Persistence
  • Multiple viewers on the same scene
ARSceneView

- Automatically aligns SceneKit’s coordinate system with the world coordinate system & moves the “virtual” camera

- Show statistics
  
  ```javascript
  sceneView.showsStatistics = true
  ```

- Debug options
  
  ```javascript
  sceneView.debugOptions = [ARSCNDebugOptions.showWorldOrigin, ARSCNDebugOptions.showFeaturePoints]
  ```

- ARSCNViewDelegate
  
  - Notifications as features, such as planes, are detected
Feature Points
Finding flat surfaces

• Define what planes to look for

\[
\text{configuration.planeDetection = [.horizontal, .vertical]}
\]

• Implement ARSCNViewDelegate methods to respond to found planes

```swift
func renderer(_ renderer: SCNSceneRenderer, didAdd node: SCNNode, for anchor: ARAnchor) {}
func renderer(_ renderer: SCNSceneRenderer, didUpdate node: SCNNode, for anchor: ARAnchor) {}
func renderer(_ renderer: SCNSceneRenderer, didRemove node: SCNNode, for anchor: ARAnchor) {}

guard let planeAnchor = anchor as? ARPlaneAnchor else {return}
switch planeAnchor.alignment {
    case .horizontal://...
    case .vertical:  //...
}
```
Anchors

- Matching anchors for different feature tracking
  - ARPlaneAnchor, ARObjectAnchor, ARImageAnchor, ARFaceAnchor

- Feature specific properties
  - PlaneAnchor: alignment, center, extend, geometry, ...
  - ImageAnchor: referenceImage
Image Recognition with ARKit

- Define what images to look for:
  - Use `referenceImage` property of an `ARImageAnchor` to access the specific images’ name and `physicalSize`

```swift
configuration.detectionImages = referenceImages

// to enable continuous tracking
configuration.maximumNumberOfTrackedImages = 1
```
Interaction with AR

• Cast a ray into the scene and find intersections with the real world

```swift
// get the point from which to cast the ray
let touchLocation = sender.location(in: sceneView)
// perform the hit test
let hitTestResult = sceneView.hitTest(touchLocation, types: [.existingPlaneUsingExtent])
```

• Array of ARHitTestResult:
  - Type & Anchor
  - Distance & transforms

- .featurePoint
- .estimatedHorizontalPlane
- .estimatedVerticalPlane
- .existingPlane
- .existingPlaneUsingExtent
- .existingPlaneUsingGeometry
ARKit Demo