



Praktikum – iOS-Entwicklung

Wintersemester 2018/19

Prof. Dr. Linnhoff-Popien

Markus Friedrich, Christoph Roch

Games

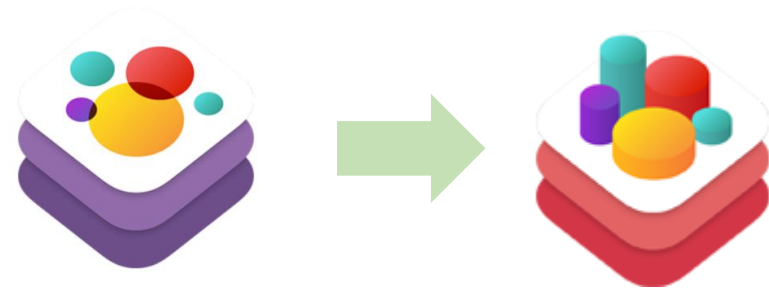
SceneKit

SceneKit

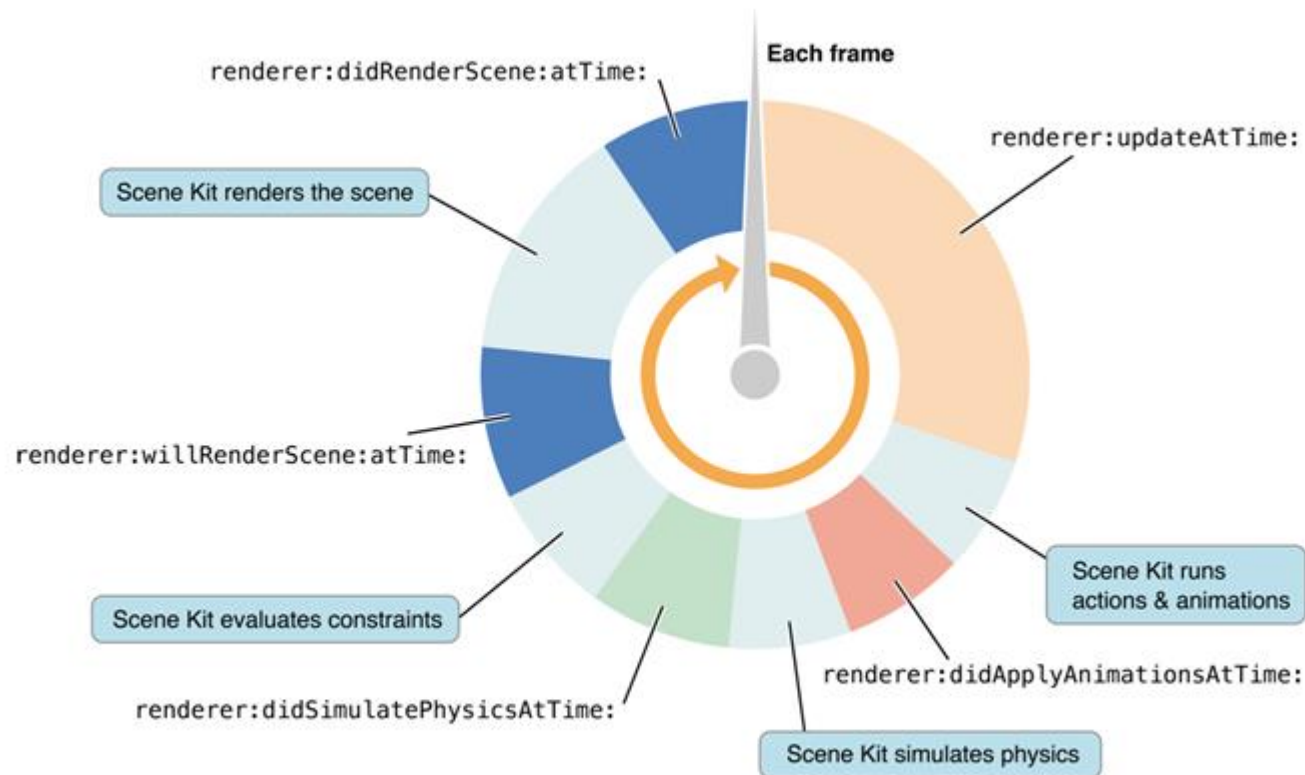
SceneKit ist ein Framework zur Erstellung von **3D** Spielen basierend auf OpenGL.

Es stellt Funktionalität für die folgenden Bereiche zur Verfügung:

- Darstellen von texturierten und animierten 3D Meshes
- 3D Partikeleffekte
- 3D Physikengine
- Audioeffekte
- Unterstützte Plattform: iOS, macOS, tvOS

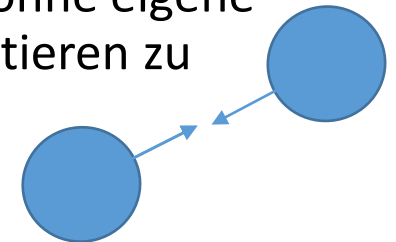


SceneKit – Game Loop



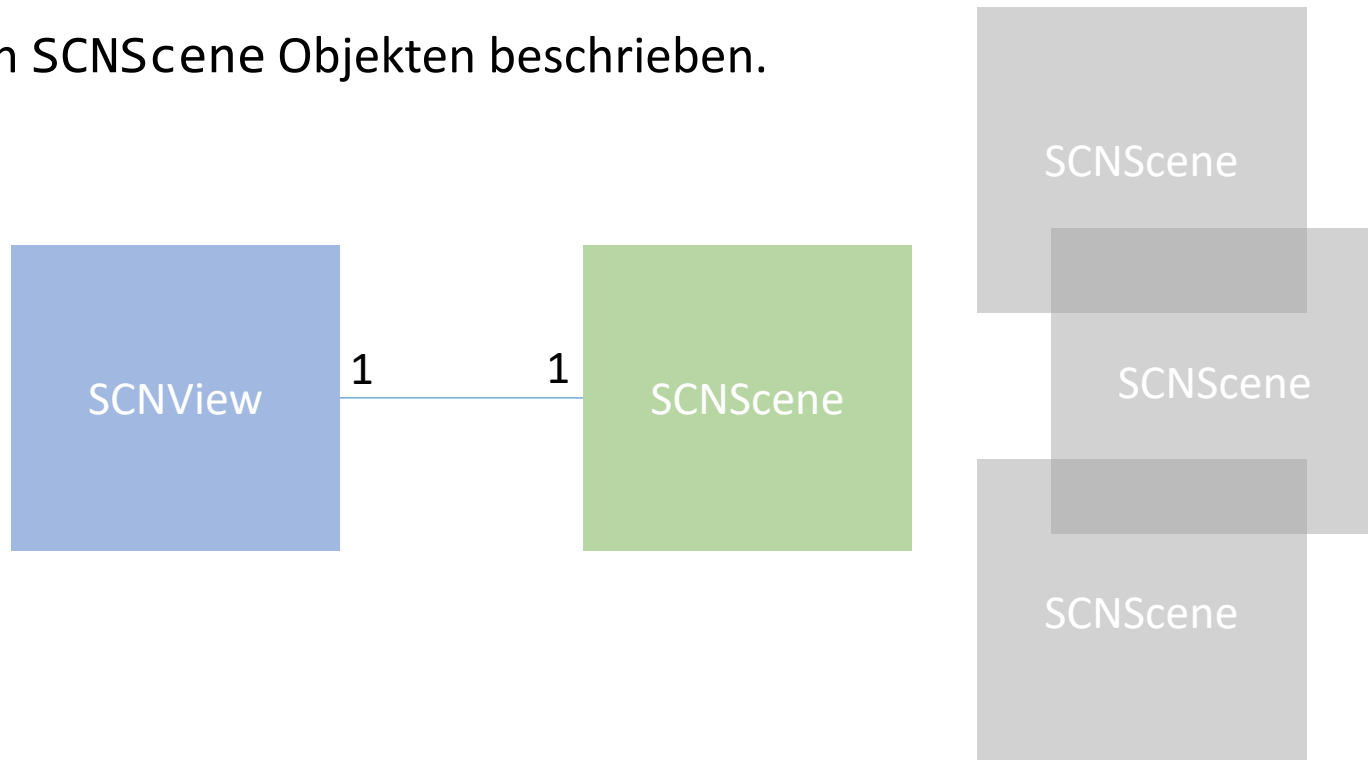
update: Wird einmal pro Frame aufgerufen. Kann für Spiellogik verwendet werden.

didApplyConstraints: Wird aufgerufen, nachdem alle Constraints angewandt wurden. Constraints (SKConstraint) werden verwendet, um Beziehungen zwischen Szenenelementen auszudrücken (bzgl. Rotation, oder Position), ohne eigene Logik implementieren zu müssen.



SceneKit – SCNView und SCNScene

- Ein SCNView Objekt bietet den Rahmen, in dem der Spielinhalt dargestellt wird.
- Die Spielinhalte werden von SCNScene Objekten beschrieben.



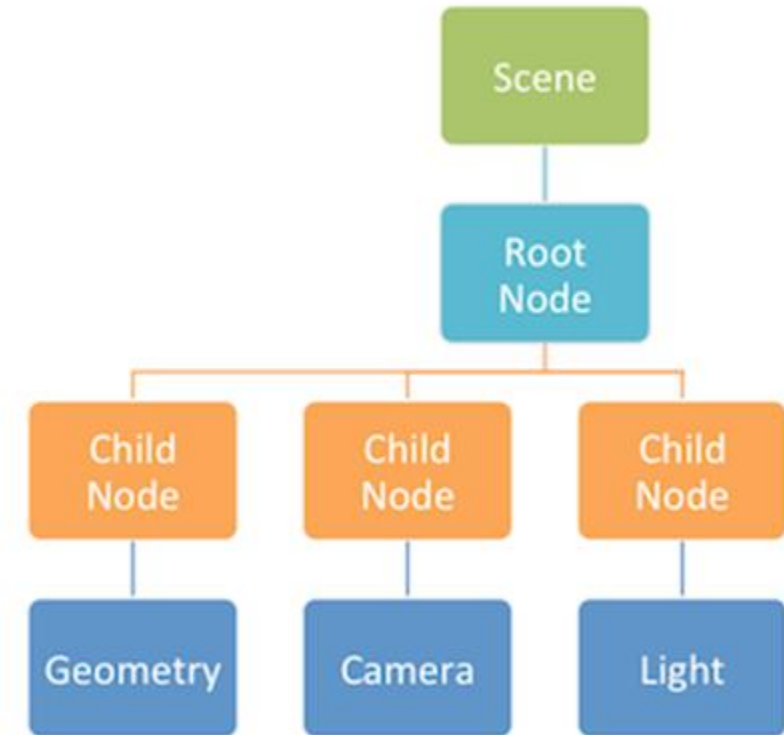
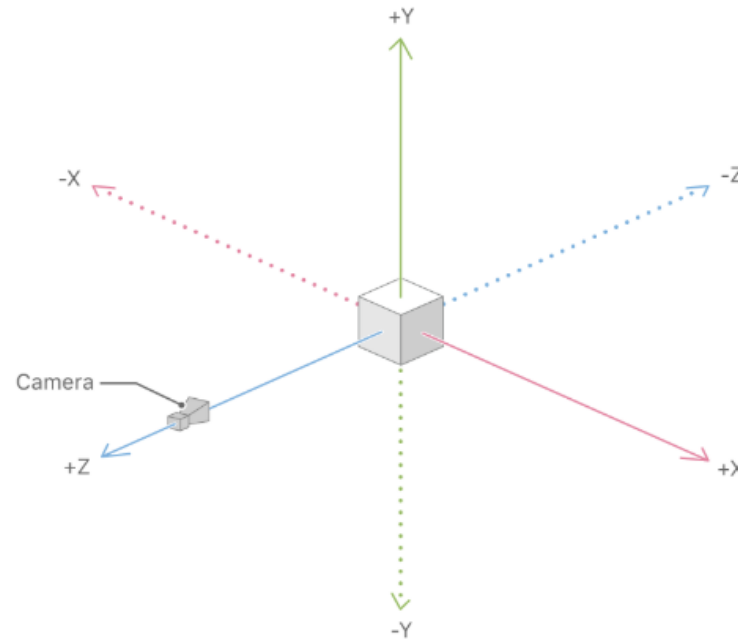
SceneKit – SCNView

- Ein SCNView Objekt kann für folgende Dinge verwendet werden:
 - Konfiguration: Bildwiederholrate, Anti-Aliasing Modus, Kontinuierliches Render ja/nein
 - Steuerung der Szenenabspielung (pause(), play(), stop())
 - Rendern von Szeneninhalten in eine Textur (snapshot())
 - Konfiguration der zu verwendenden Kamera
- Weitere Konfigurationsoptionen können per dictionary (options) an die init Methode übergeben werden.
- Die Szene, die dargestellt werden soll, ist im scene Property abgelegt.

Wichtig: SCNView implementiert das SCNSceneRenderer Protokoll. Weitere Möglichkeiten: SCNLayer (Core Animation Layer), SCNSceneRenderer

SceneKit – SCNScene

- Eine Szene wird in einem sog. **Scene Graph** organisiert.
- Koordinatensystem:



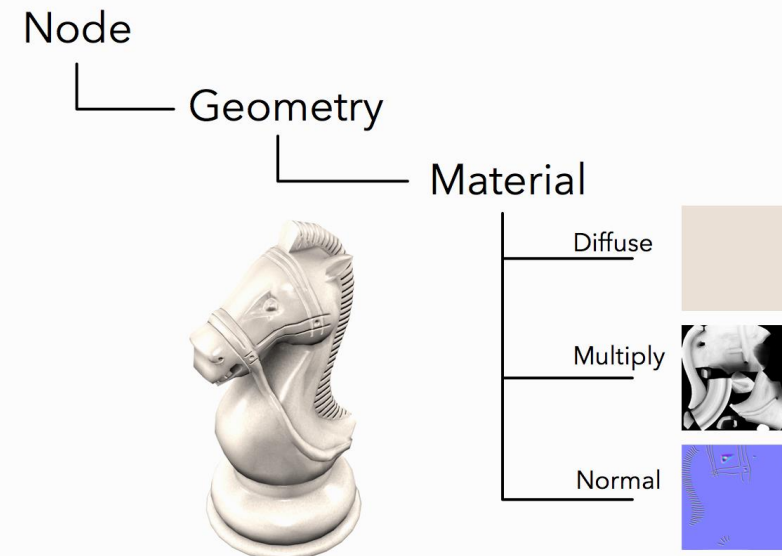
<https://docs.microsoft.com/en-us/xamarin/ios/platform/gaming/scenekit>

- SCNScene Objekte können Inhalte persistieren und laden.
- Scenes lassen sich per Dictionary Property konfigurieren (z.B. Animationsstartzeit und Bildrate).

SceneKit – SCNNode

- SCNNode Objekte Beschreiben die Struktur der Szene.
- Wichtige Properties: `position`, `rotation`, `scale`
- SCNNode Objekte können hierarchisch organisiert werden.
- Attachment Properties: `light`, `camera`, `geometry`, `morpher`, `skinner`.

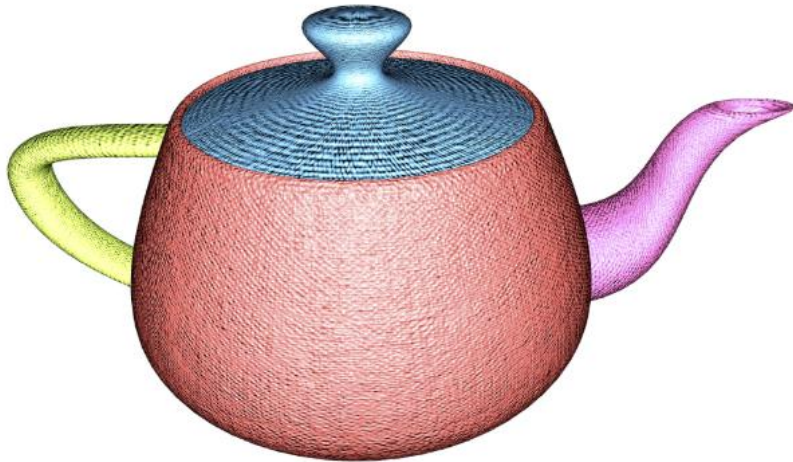
Wichtig: != SpriteKit: Particle System, Audio, Licht sind keine abgeleiteten Klassen, sondern „Attachments“.



<https://www.objc.io/issues/18-games/scenekit/>

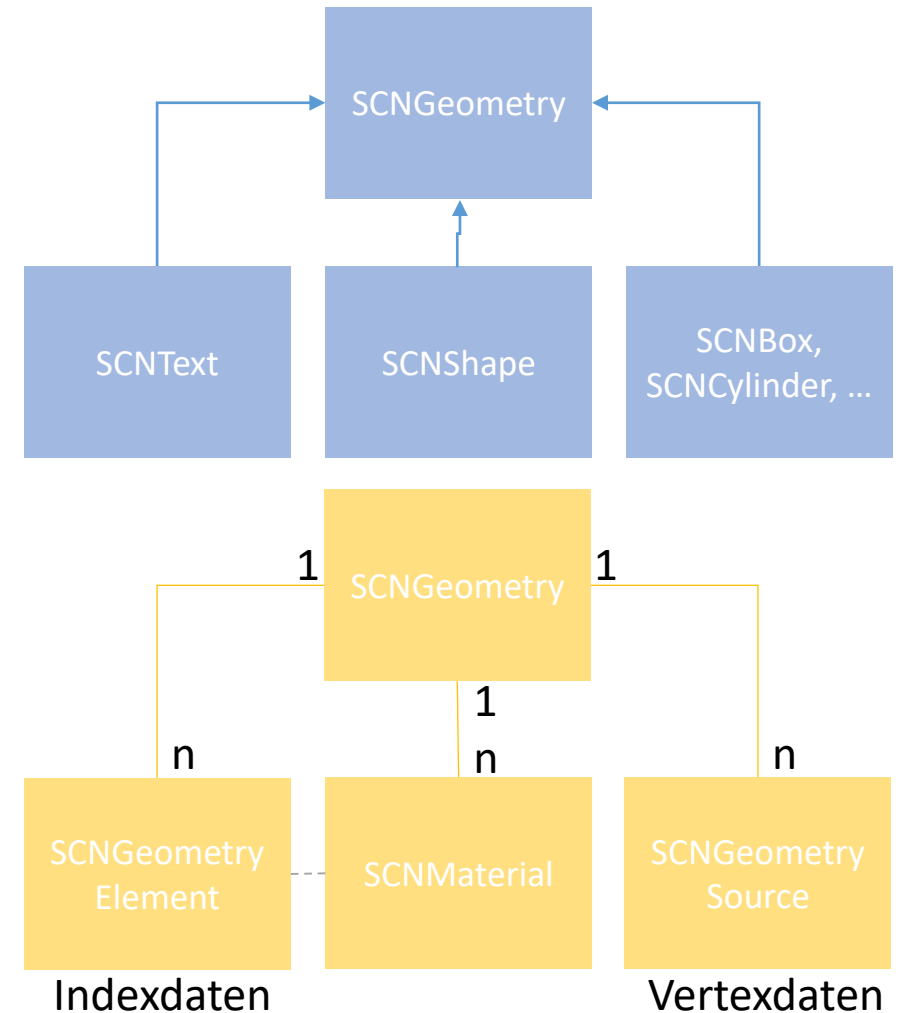
SceneKit – SCNNode Attachments - Geometrie

- Die SCNGeometry Klasse verwaltet Geometriedaten.



<https://developer.apple.com/documentation/scenekit/scngeometryelement?language=objc>

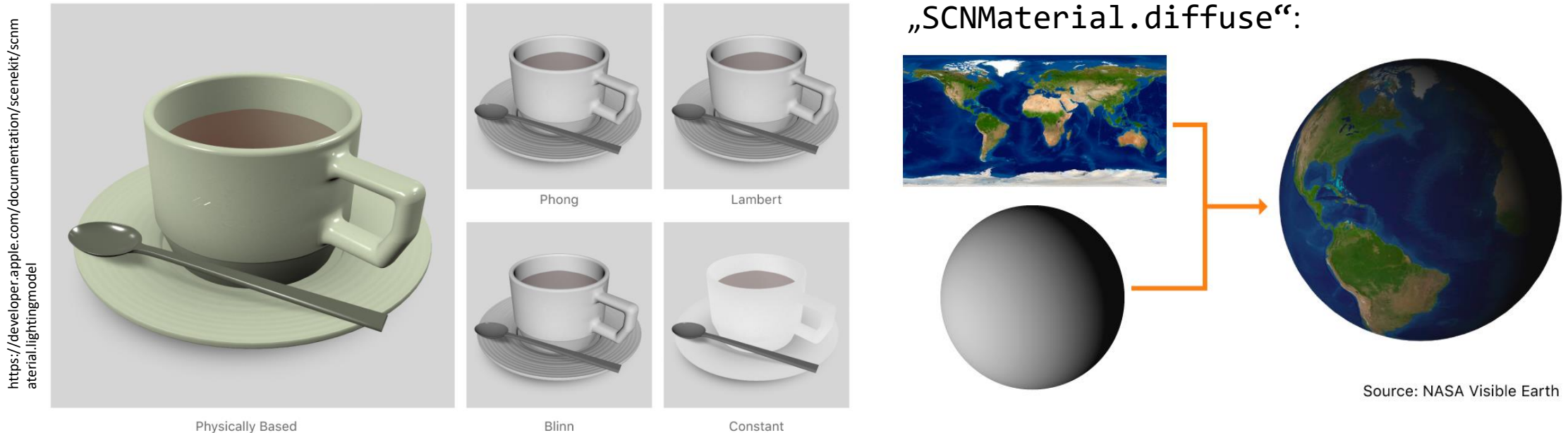
- Kopie von SCNGeometry Objekten: `copy()` (Teilt Geometriedaten, Material dann pro Kopie)
- Geometrie-Animation: `SCNMorpher`, `SCNSkinner`



SceneKit – SCNNode Attachments - Material

- Beschreibung der Oberflächenerscheinung eines Nodes (`SCNMaterial`).
- Lichtmodell (`lightingModel` Property) kombiniert Material- und Beleuchtungseigenschaften:

**Beispiel `SCNMaterialProperty`
„`SCNMaterial.diffuse`“:**



- Texturen werden z.B. in `SCNMaterialProperty.contents` abgelegt.

SceneKit – SCNNode Attachments - Material

- Wichtig für „**physicallyBased**“ Lichtmodell:

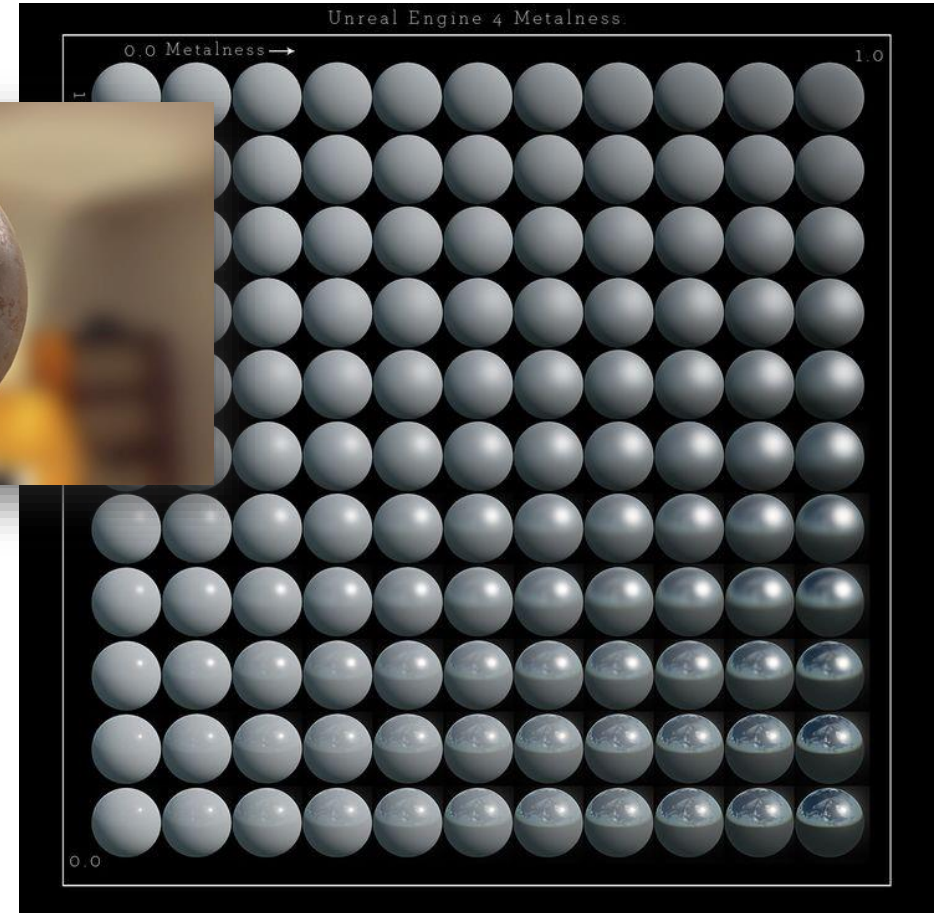
SCNMaterialProperty

- **diffuse**
- **roughness**
- **metalness**
- (normal, ambientOcclusion, displacement)

- Zusätzlich: Umgebungsbeleuchtung (scene.lightingEnvironment):



<https://blog.markdaws.net/arkit-by-example-part-4-realism-lighting-pbr-b9a0bedb013e>



<https://3delight.atlassian.net/wiki/spaces/3DFM/pages/16252931/Image+Based+Lighting>

<https://medium.com/@avihay/amazing-physically-based-rendering-using-the-new-ios-10-scenekit-2489e43f7021>

SceneKit – SCNNode Attachments - Licht

- Lichtquellen werden mit `SCNLight` Objekten definiert.



Ambient



Directional



Omni



Spot

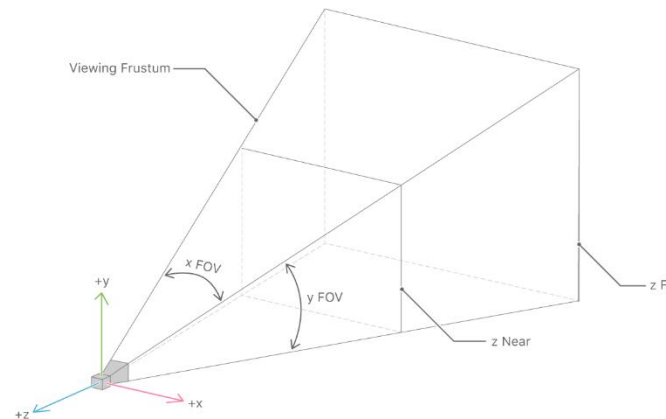
<https://developer.apple.com/documentation/scenekit/scnlight.lighttype>

Wichtig: Es werden auch kompliziertere Lichtquellentypen unterstützt.

- Schattenwurf kann mit `castsShadow` eingeschaltet und dann konfiguriert werden (z.B. Radius). `castsShadow` muss auch im `SCNNode` gesetzt werden.
- Festlegung der Lichtkategorie: `categoryBitMask` (Node: Wenn `categoryBitMask` auf die Lichtkategorie gesetzt ist => Ausschluss).

SceneKit – SCNNode Attachments - Kamera

- Klassische Kamera als camera property eines Nodes (SCNCamera).
- Zusatzeffekte:
 - Depth-of-Field
 - Motion Blur
 - HDR
 - Bloom
 - Screen-Space AO



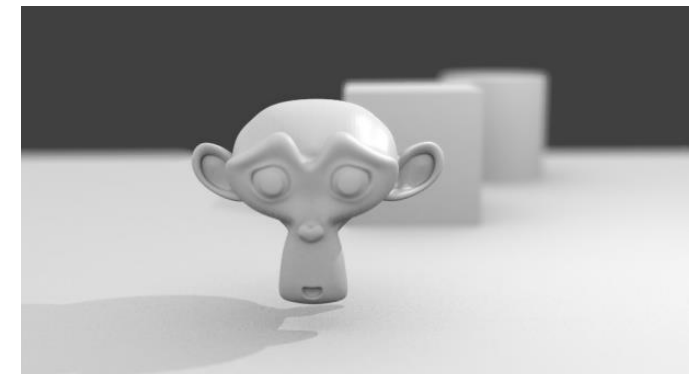
<https://www.youtube.com/watch?v=Y16jJflxcHc>



https://de.wikipedia.org/wiki/Screen_Space_Ambient_Occlusion#/media/File:Screen_space_ambient_occlusion.jpg

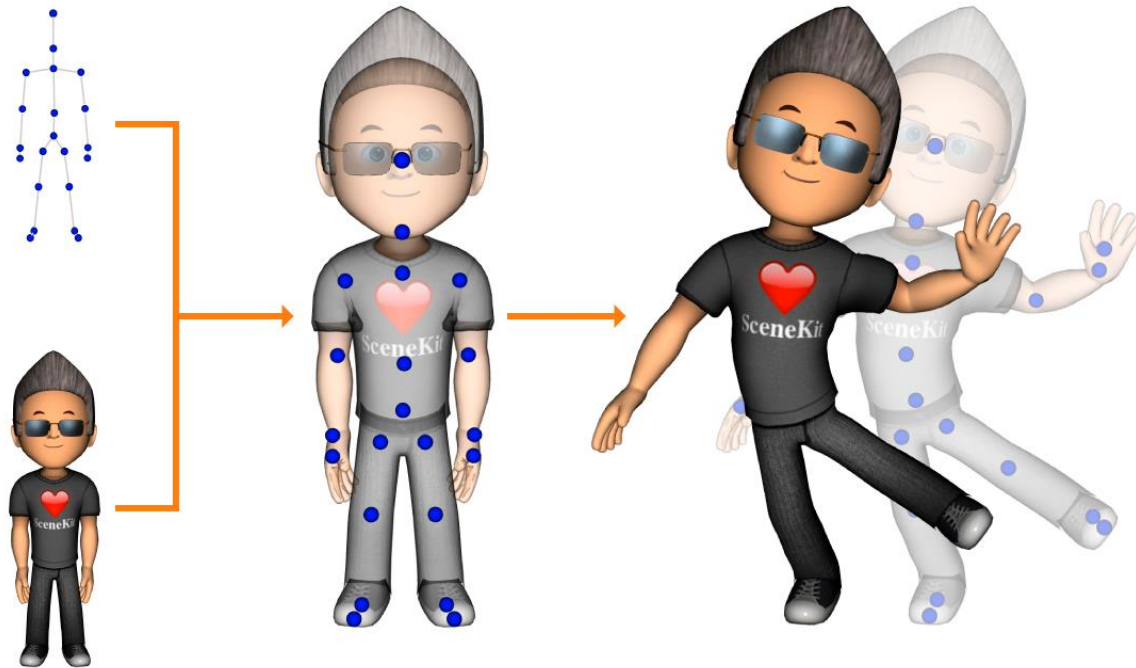


<https://sharathpatali.wordpress.com/tag/gsl/>



<http://www.versluis.com/2015/04/how-to-render-with-depth-of-field-in-blender/>

SceneKit – SCNNode Attachments - Skinner

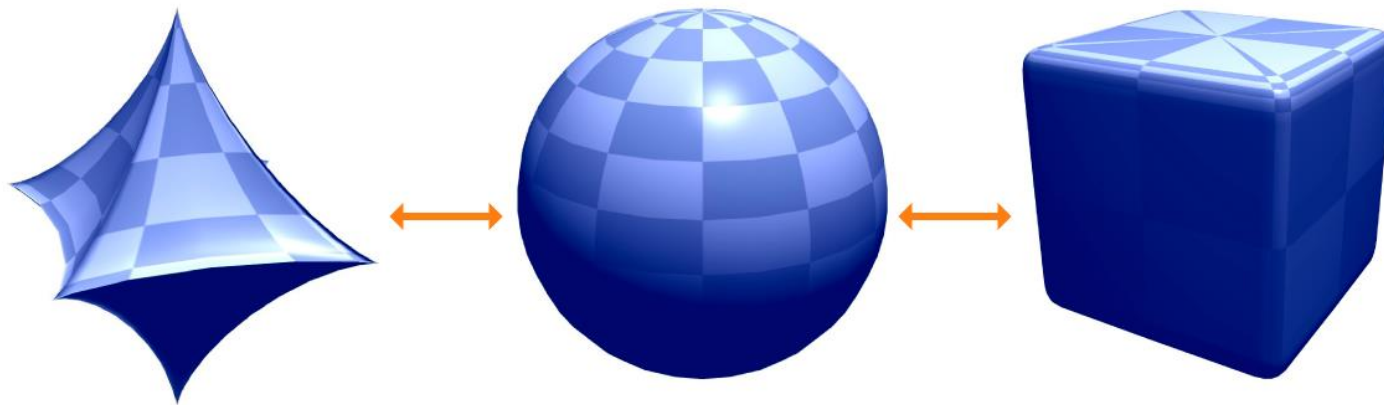


<https://developer.apple.com/documentation/scenekit/scnskinner>

- `SCNNode.skinner` Property
- SCNSkinner Objekte benötigen:
 - Basis-Geometrie, die verformt wird
 - Ein „Skelett“, das Kontrollknoten liefert
 - Gewichte, die den Einfluss einzelner Knochen auf die Basis-Geometrie festlegen

SceneKit – SCNNode Attachments - Morpher

- Transitionen zwischen zwei oder mehreren Geometrien.
- `SCNNode.morpher` Property
- Zielgeometrien werden als `SCNGeometry` Objekte beschrieben und zusätzlich gewichtet.



<https://developer.apple.com/documentation/scenekit/scnmorpher>

Wichtig: Animationen werden über `CAAnimation` Objekte konfiguriert.

SceneKit – SCNNode Attachments - Physik

- `SCNNode.physicsBody` Property
- Globale Parameter (z.B. Schwerkraft) über `SCNPhysicsWorld` Objekt (Property des `SCNScene` Objekts)
- `SCNPhysicsBody`:
 - Typ (`SCNPhysicsBodyType`): Statisch, Dynamisch, Kinematisch (Wird nicht von Kräften oder Kollisionen beeinflusst, löst aber Kollisionen aus)
 - Physikalische Eigenschaft (Geschwindigkeit, Kräfte aus bestimmter Richtung, Masse, Reibung, Ladung, ...)
 - Form (`SCNPhysicsShape`):
 - Kann automatisch erzeugt werden (einfachste wird gewählt).
 - Für Basisgeometrie (z.B. `SCNBox`) wird eine implizite Beschreibung gewählt.
 - => Beste Performance: Komposition aus transformierten Formen.
 - Optionen (`SCNPhysicsShape.Option`): Benutzter LOD für Geometrie=>Form Umsetzung, Typ (bounding Box, konvexe Hülle,...)

Games

ARKit

ARKit

Neu: ARKit 2 mit
Multiplayer und
Persistenz



- Virtuelle Objekte auf Echtwelt-Oberflächen platzieren:



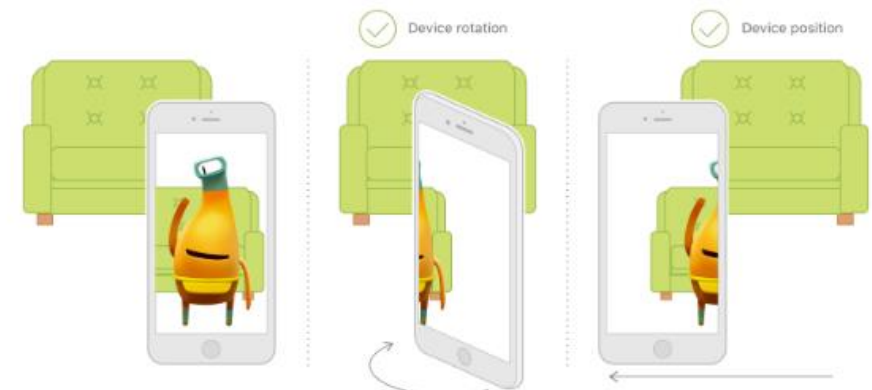
<https://www.archdaily.com/879403/the-real-star-of-the-apple-keynote-arkit-augmented-reality-technology>

- Exakte Gesichtserkennung (Ausdruck, Topologie, Position, Orientierung)
- Präzise Positionserkennung
- Checkout: <https://developer.apple.com/arkit/>

ARKit – ARSession

- Koordinatorobjekt aller ARKit relevanter Prozesse:
 - Lesen von Bewegungssensordaten
 - Kontrolle der Kamerasysteme
 - Schätzen der Pose + Tiefenwerte
 - Hinzufügen von Ankern, die dann getrackt werden
 - Zugriff auf Weltkarte (ARWorldMap) mit komplettem State (WKS, Ankern, Dimensionen, Feature Punktwolke)
 - Erzeugung von 3D Referenzobjekten aus einem 3D Scan
- Wird über ein ARConfiguration Objekt konfiguriert.
 - Beispiel: ARWorldTrackingConfiguration
 - Konfiguriert 6DOF Bewegungstracking
 - Ebenenerkennung Ja/Nein
 - Bilderkennung Ja/Nein
 - 3D Objekterkennung
 - Umgebungstexturaufnahme (Cubemap für image-based lighting)
 - Autofokus aktiviert Ja/Nein
 - Beispiel: ARObjectScanningConfiguration
 - Für detailreiche 3D Scans

Wichtig für ARKit 2
Multiplayer und
Persistenz



<https://developer.apple.com/documentation/arkit/arworldtrackingconfiguration>

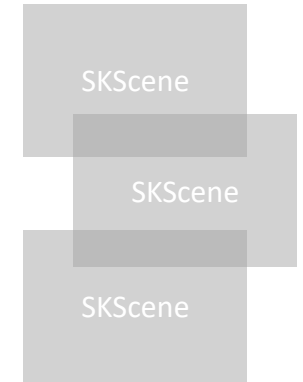
ARKit – ARSession und ARFrame

- Zugriff auf Informationen zum aktuellen Frame
 - Per `ARSession.currentFrame`
 - Oder per `ARSessionDelegate` Implementierung
- Enthält Informationen zu:
 - Pixelpuffer des aktuellen Kamerabilds
 - Zeitstempel
 - Geschätzte Tiefenkarte
 - Weltkartenstatus (Wie gut ist die aktuelle Weltkarte)
 - Aktuelle Punktwolke
 - Kamera (Tracking Qualität, externe und interne Parameter)
 - Geschätzte Lichtumgebung (Ambiente Lichtintensität, Lichtfarbe)

ArKit – ARSCNView und ARSKView

- SpriteKit: ARSKView

- Rendert Kamerabild im Hintergrund
- Rendern von automatisch rotierten und skalierten 2D Elementen an Ankerpunkten



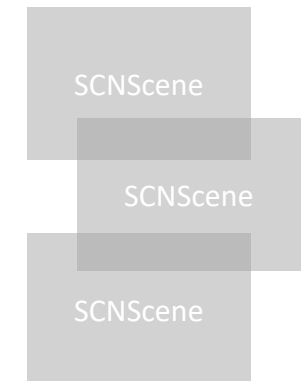
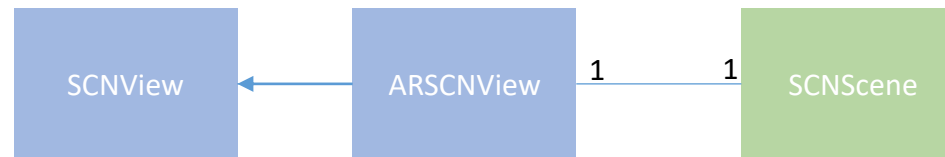
<https://www.raywenderlich.com/378-augmented-reality-and-arkit-tutorial>



Bild: Apple

- SceneKit: ARSCNView

- Rendert Kamerabild im Hintergrund
- Setzt SceneKit WKS auf ARKit WKS
- Setzt SceneKit Kamera auf ARKit Kamera



ARKit – ARSCNView Beispiel

Wichtig für Interaktion:
ARSCNView.hitTest

- Beispiel für die Platzierung eines Objekts zu einem Anker:
- Implementierung des ARSCNViewDelegates:

https://developer.apple.com/documentation/arkit/arscnview/providing_3d_virtual_content_with_scenekit

```
func renderer(_ renderer: SCNSceneRenderer, didAdd node: SCNNode, for anchor: ARAnchor) {  
  
    guard let planeAnchor = anchor as? ARPlaneAnchor else { return }  
  
    let plane = SCNPlane(width: CGFloat(planeAnchor.extent.x), height:  
        CGFloat(planeAnchor.extent.z))  
    let planeNode = SCNNode(geometry: plane)  
  
    planeNode.position = SCNVector3Make(planeAnchor.center.x, 0, planeAnchor.center.z)  
    planeNode.transform = SCNMatrix4MakeRotation(-Float.pi / 2, 1, 0, 0)  
  
    node.addChildNode(planeNode)  
}
```

ARKit – Ankerpunkte

- Ankerpunkte (ARAnchor) sind Echtweltpositionen, an denen Objekte platziert werden können (mit `ARSession.add(anchor: ARAnchor)`).

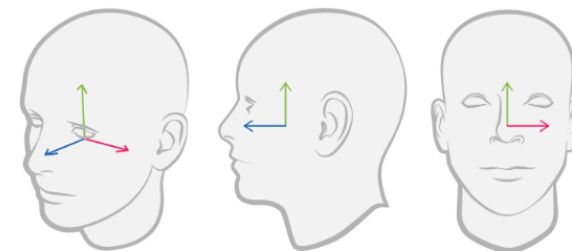
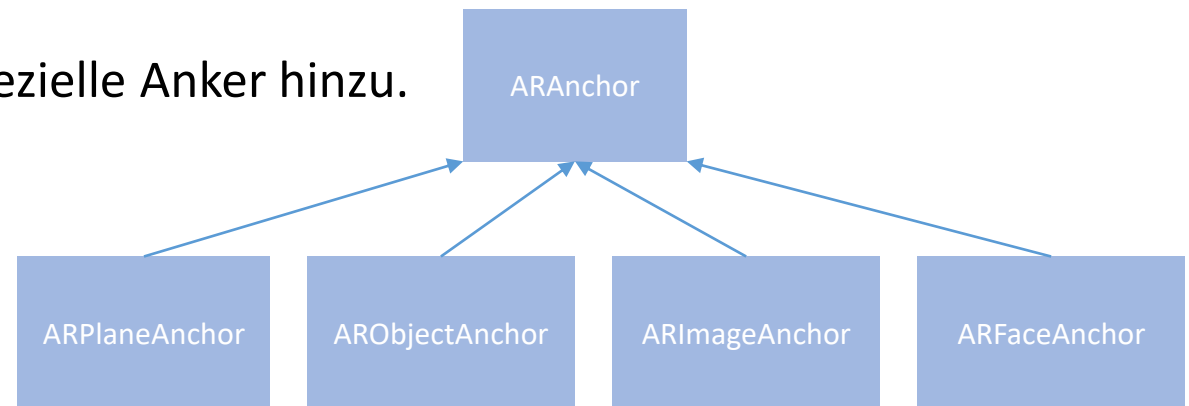
- Manche Subsysteme fügen automatisch spezielle Anker hinzu.

- Beispiel 1: ARPlaneAnchor

- Wenn planeDetection in Konfig. aktiviert
 - Fügt Anker mit Position und Orientierung von erkannten Ebenen hinzu.

- Beispiel 2: ARFaceAnchor

- Wenn ARSession mit ARFaceTrackingConfiguration konfiguriert
 - Fügt Anker mit Pose, Topologie und Ausdruck eines Gesichts hinzu.
 - **Sehr genau!** z.B. für Eye Tracking



<https://developer.apple.com/documentation/arkit/arfaceanchor>