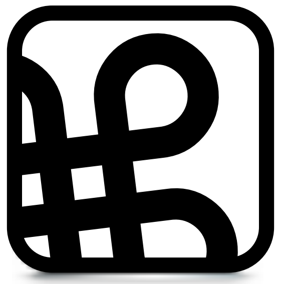


Macoun

QUALITY APPLICATIONS

Einführung in Scene Kit

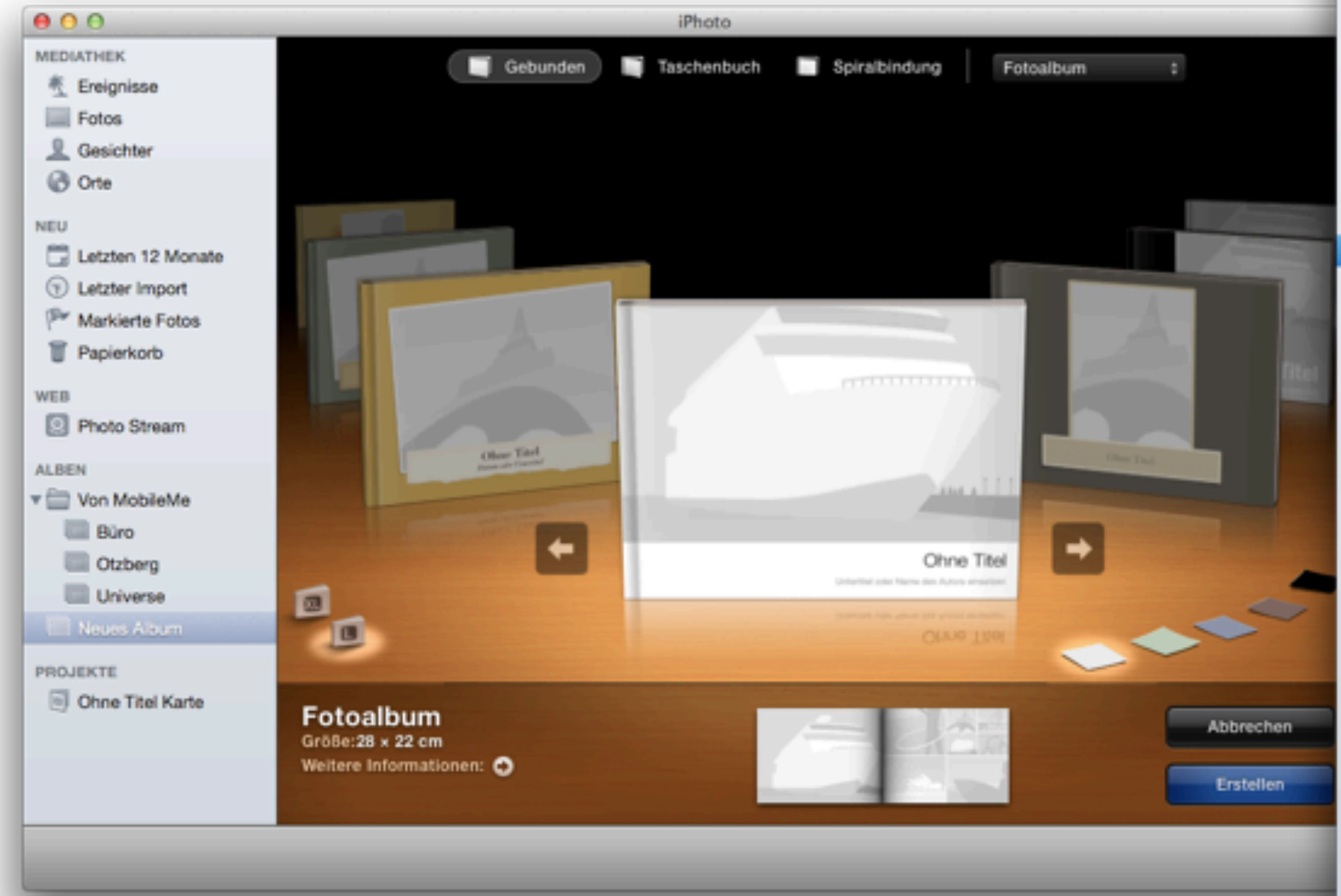
Daniel Dönigus

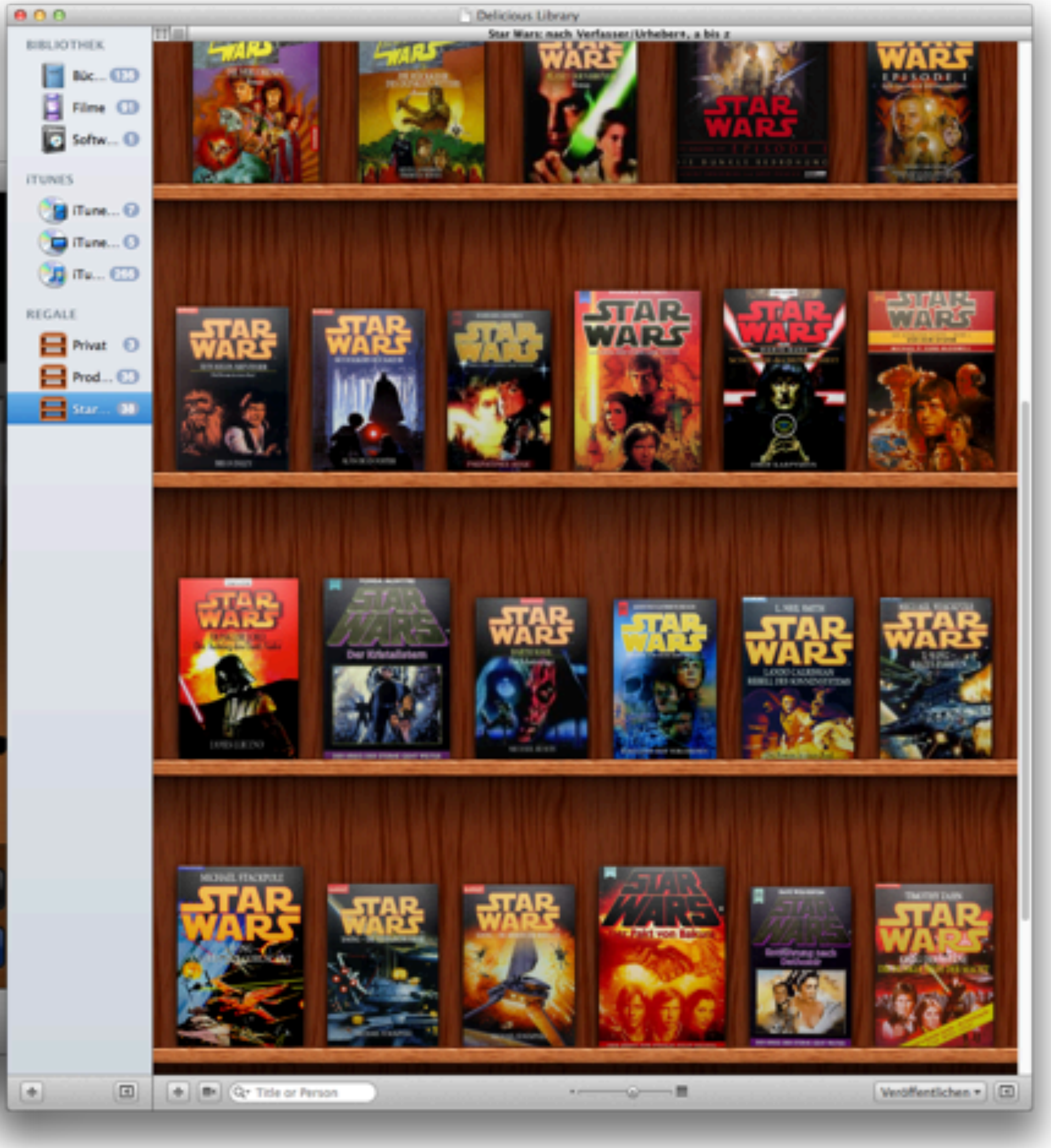
Ablauf

- Einführung / Grundlagen
- Werkzeuge
- Features und Programmierung
- Erweiterte Funktionen
- Demo Apps

Grundlagen

3D in Applikationen

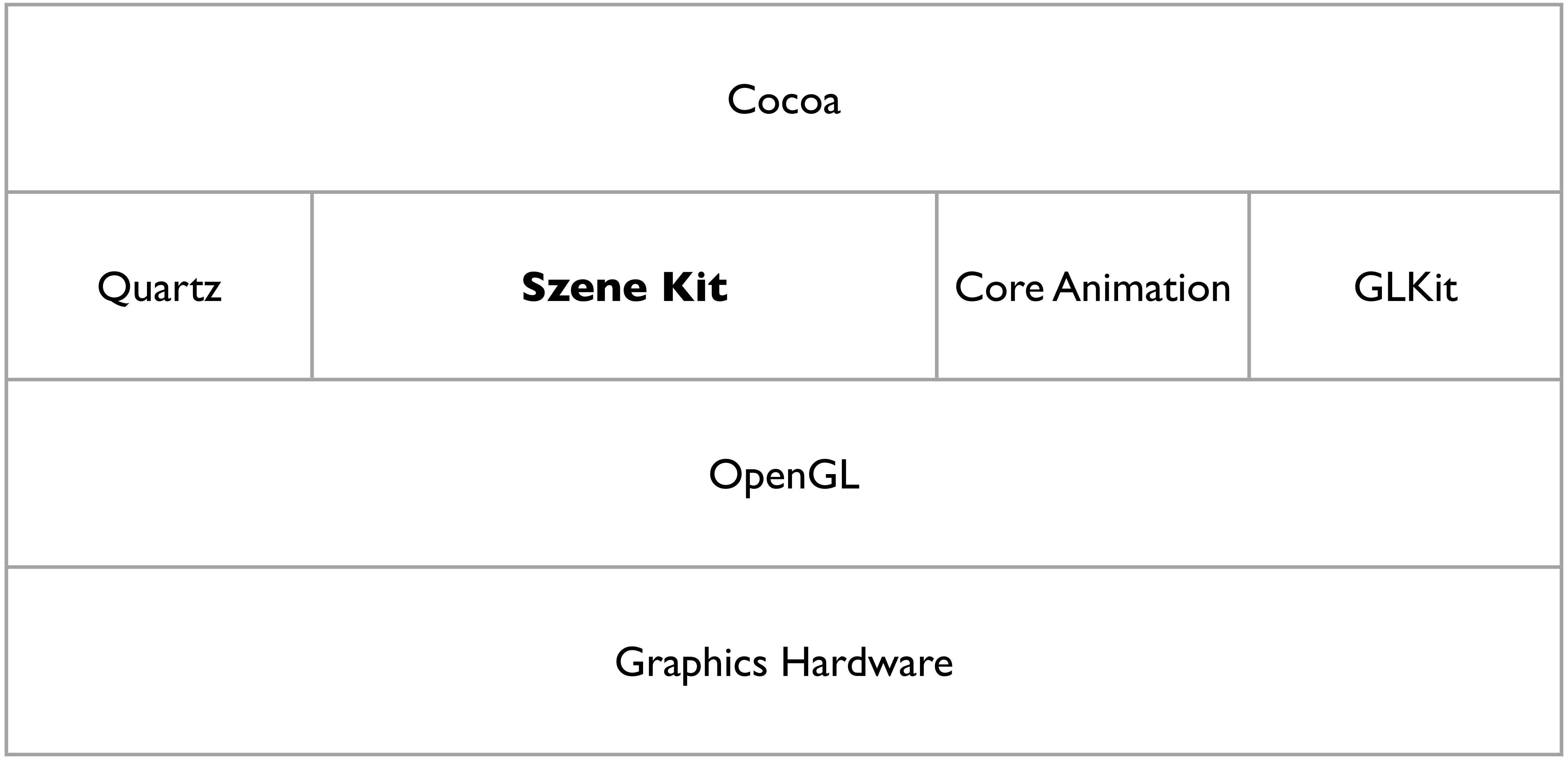




Scene Kit

- High Level 3D Framework
- Darstellung und Modifikation von 3D-Szenen
- Grundlage Szenegraphen: Collada-Format (DAE-Dateien)
- Zum Großteil animiert - Core Animation API

Grafikframeworks (ML)



Cocoa

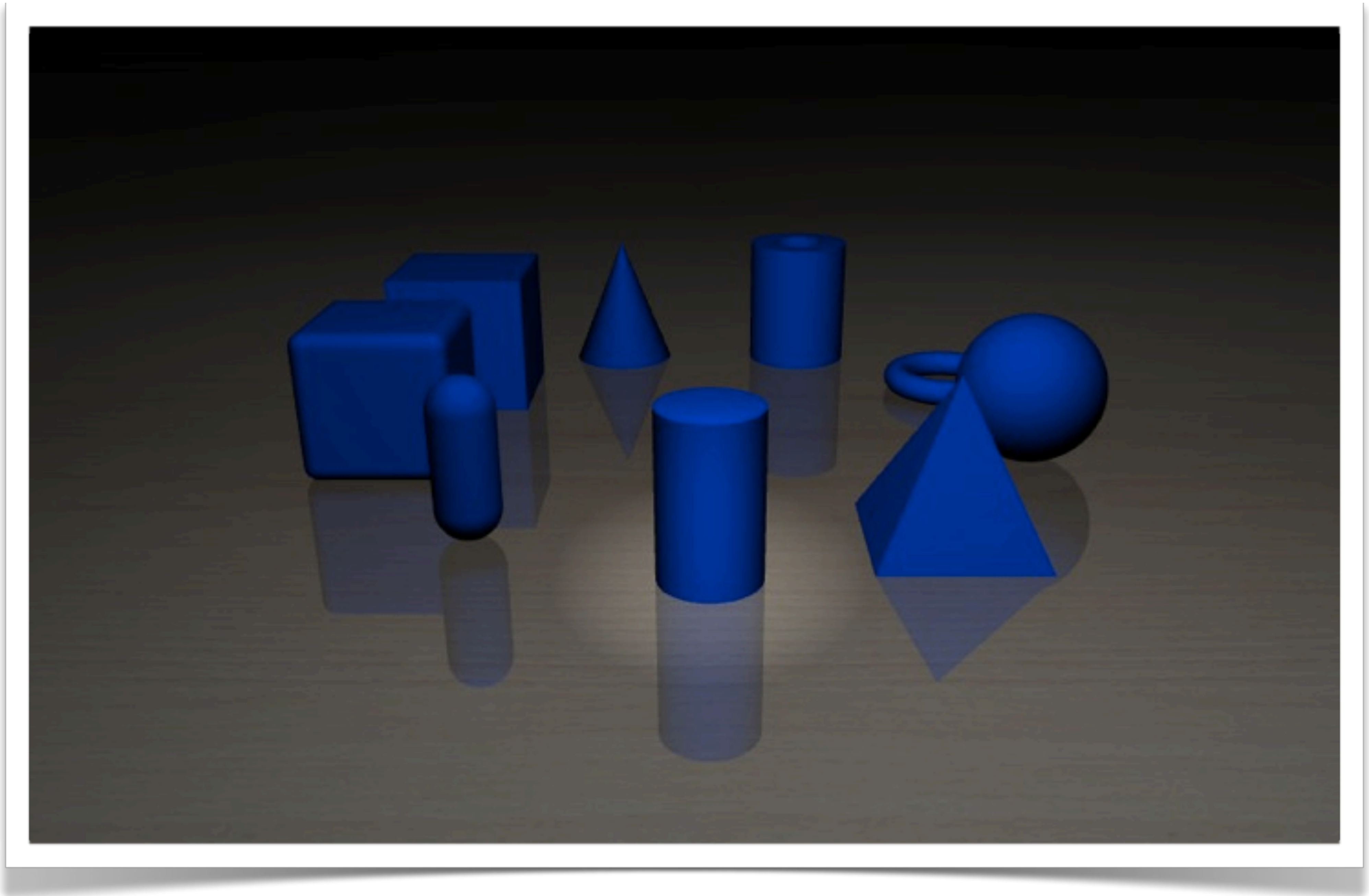
Quartz | **Szene Kit** | Core Animation | GLKit

OpenGL

Graphics Hardware

COLLADA (DAE-Dateien)

3D Geometrie (Primitive)



Materialien / Texturen

- Farbe
- Glanz/Mattheit
- Texturen
- Transparenz
- Chrom, spiegelndes Material

Lichtquellen

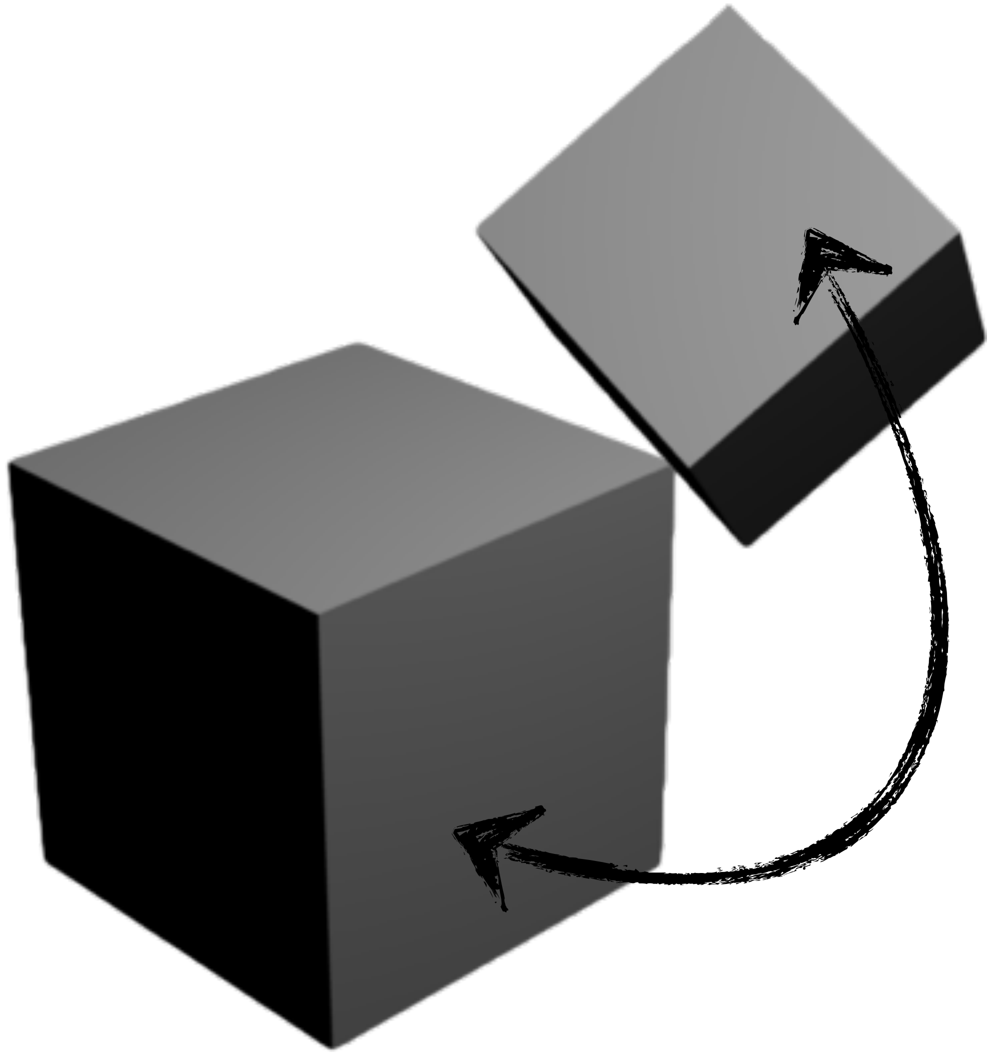
- Arten
  - Ambient
  - Omnidirektional
  - Gerichtet
  - Spot
- Farbig

Kamera

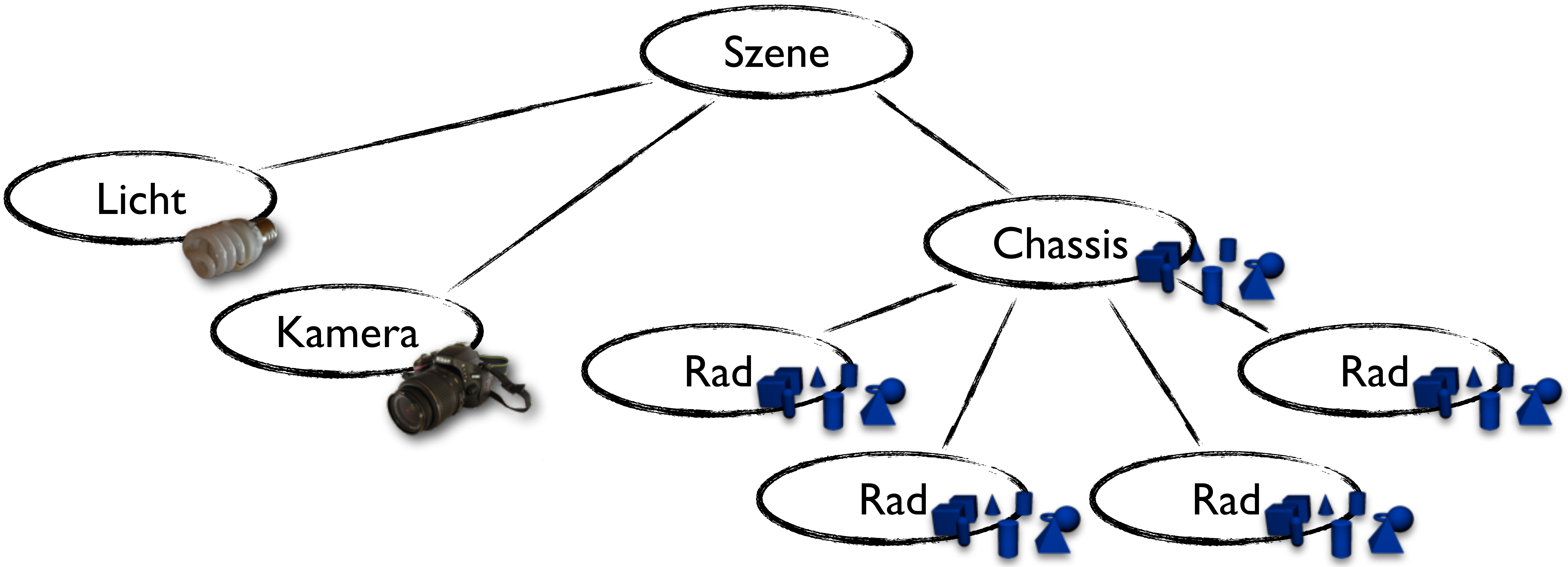
- Position und Orientierung (Node)
- Öffnungswinkel
- Verhältnis Höhe/Breite
- Near und Far-Plane

Transformationen

- Translation
- Skalierung
- Rotation



Szenegraphen

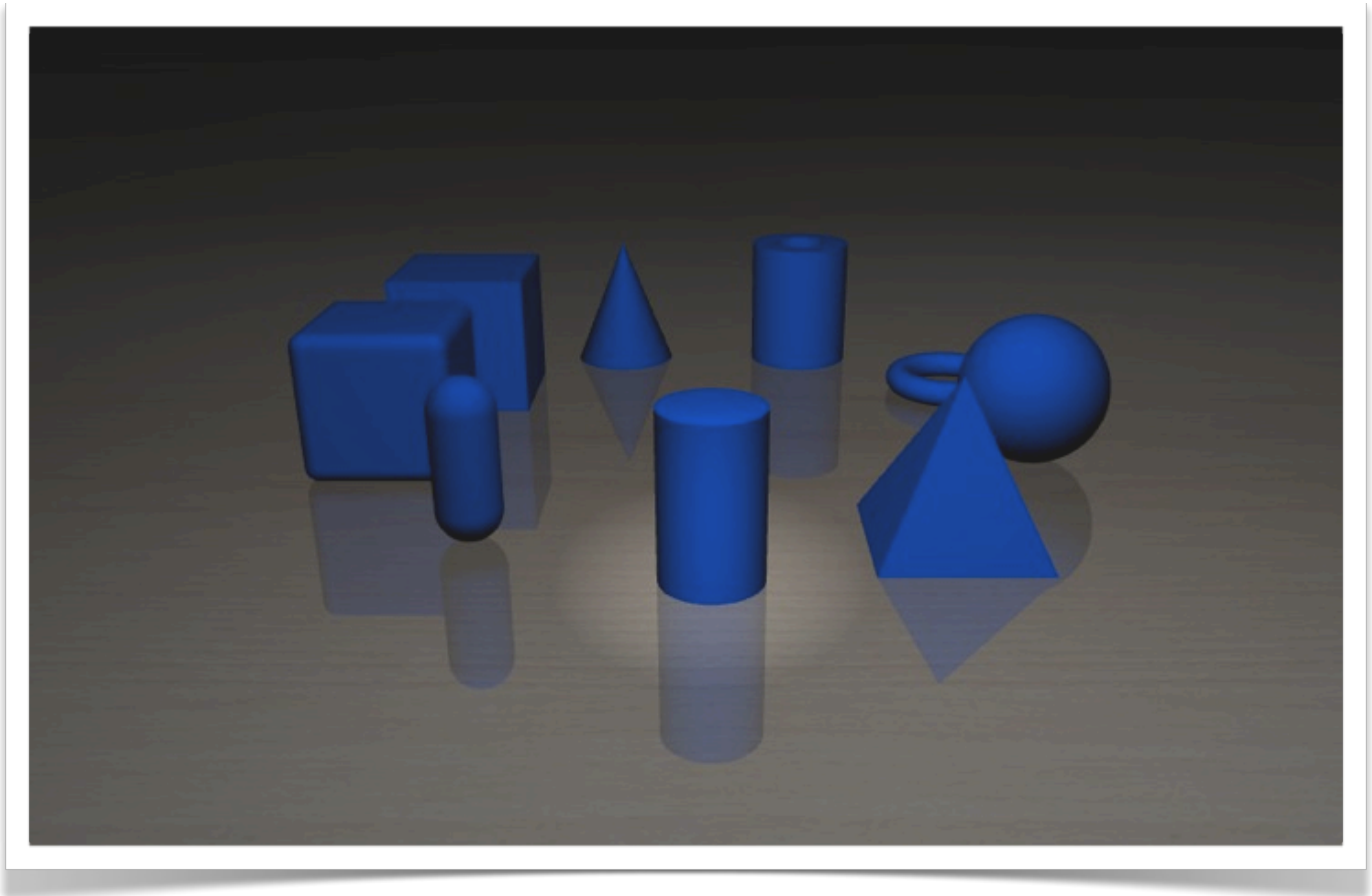


Werkzeuge

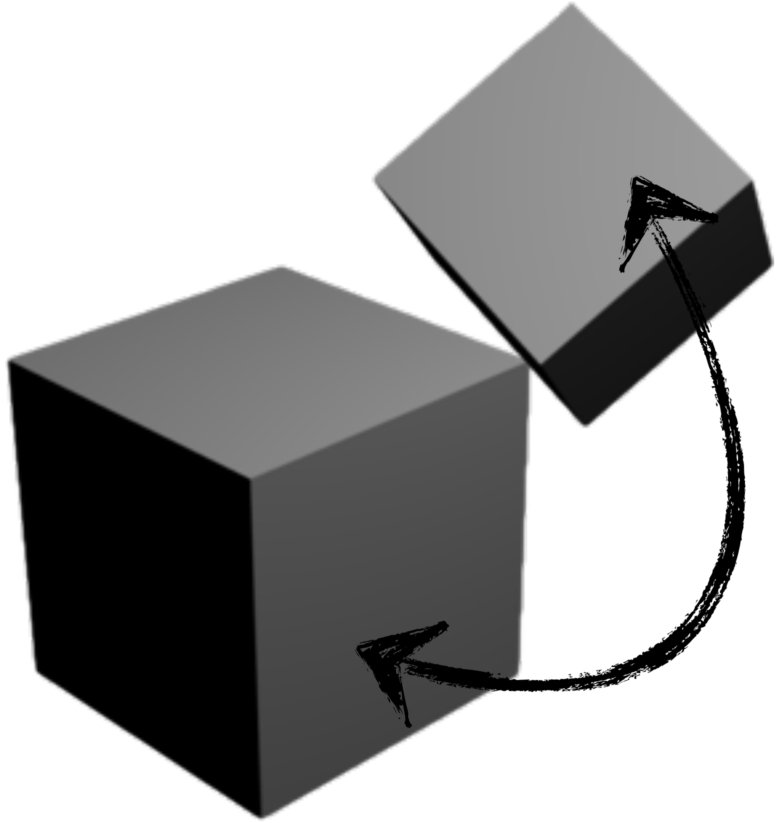
Features und Programmierung

Demo

SCNView / SCNScene



SCNNode (Rekursiv)



Optional:

SCNGeometry

```
SCNScene* scene = ... //my scene
SCNMaterial* material = ... //get a material

SCNNode* sphereNode = [SCNNode node];
SCNGeometry* sphere = [SCNSphere sphereWithRadius:1.0];
sphere.firstMaterial = material;
sphereNode.geometry = sphere;
[scene.rootNode addChildNode:sphereNode];

SCNNode* textNode = [SCNNode node];
SCNGeometry* text = [SCNText textWithString:@“Macoun“ extrusionDepth=1.0];
text.firstMaterial = material;
textNode.geometry = text;
[scene.rootNode addChildNode:textNode];
```

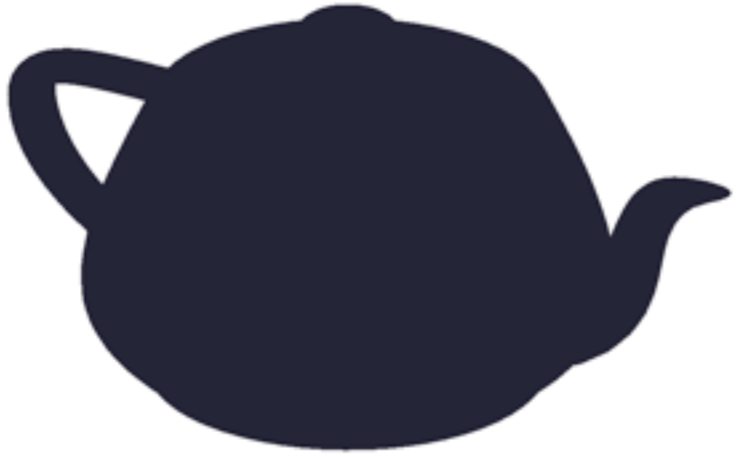
Spiegelungen - SCNFloor

```
SCNScene* scene = ... //my scene
SCNMaterial* material = ... //get a material

SCNNode* floorNode = [SCNNode node];
SCNGeometry* floor = [SCNFloor floor];
floor.material = material;
sphereNode.geometry = floor;
[scene.rootNode addChildNode:floorNode];
```

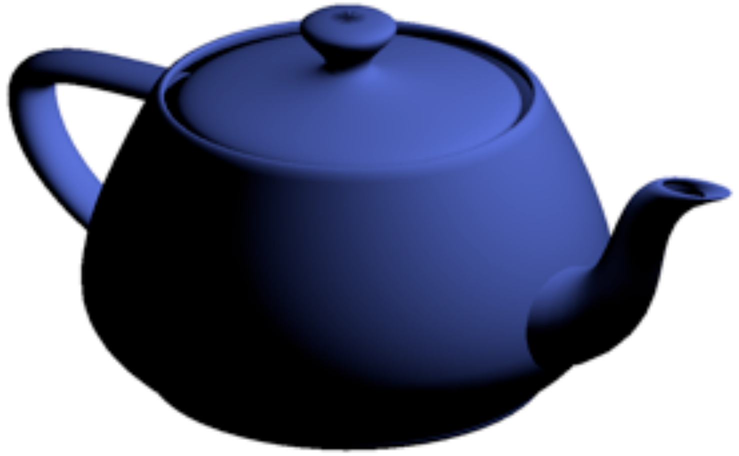
SCNMaterial

Blinn/Phong Beleuchtung



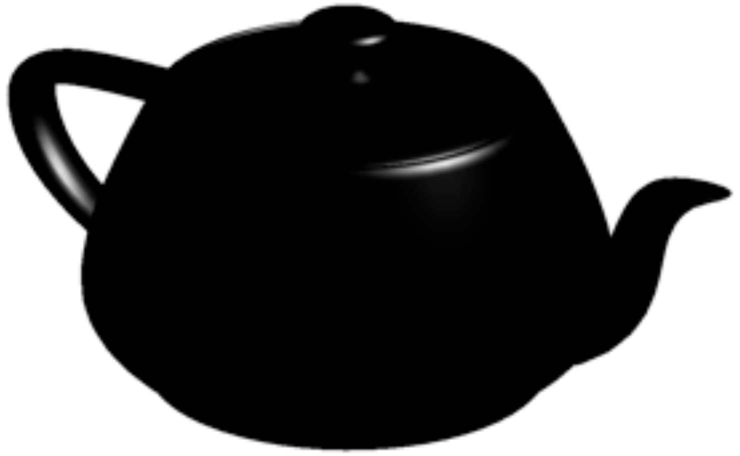
Ambient

+

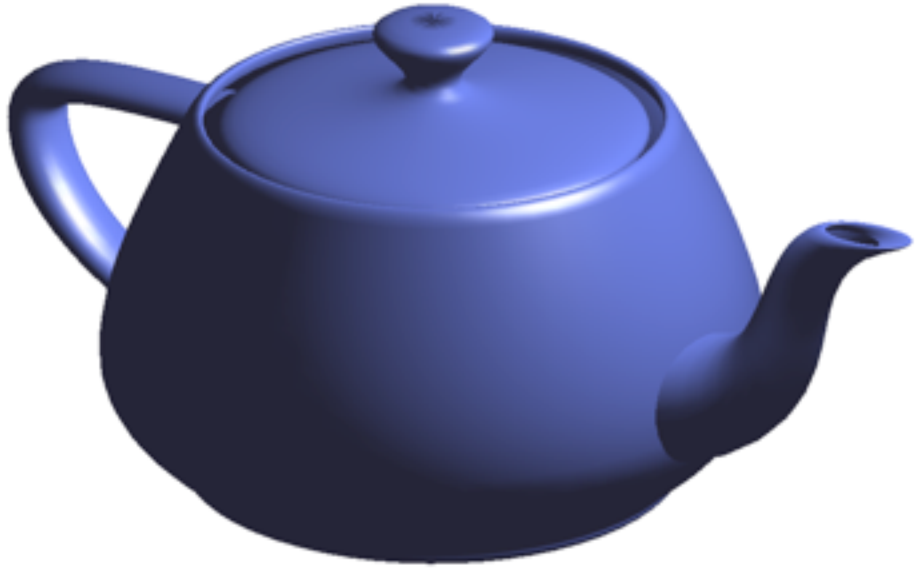


Diffuse

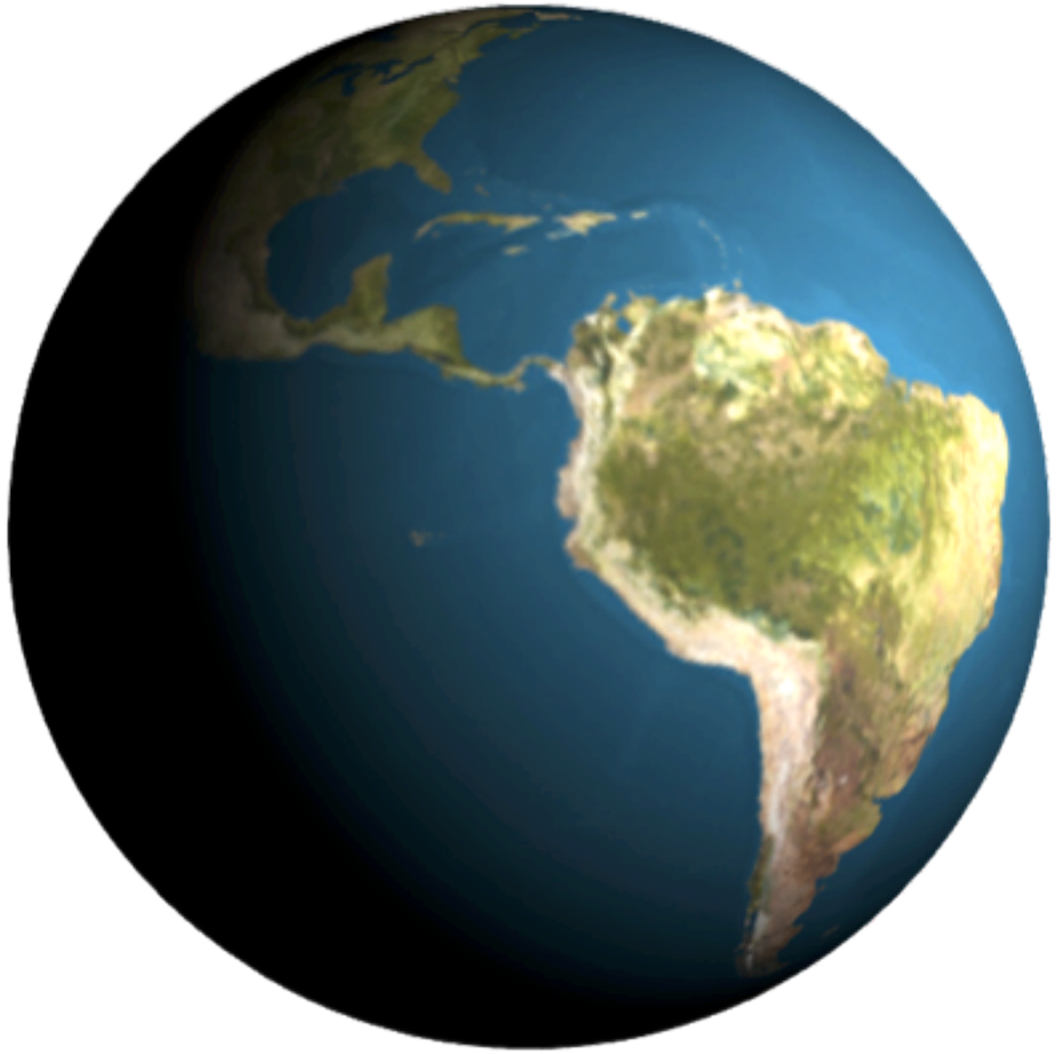
+



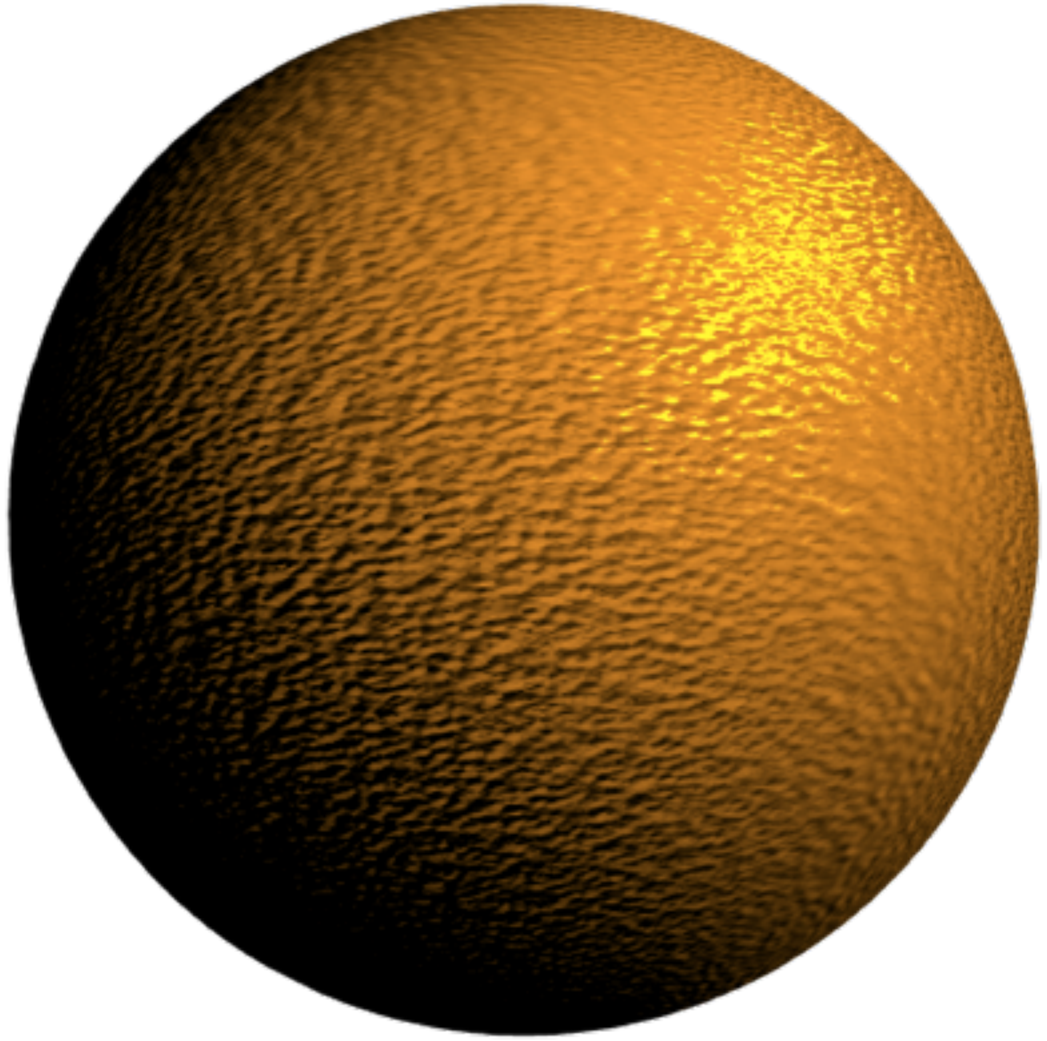
Specular+Shininess



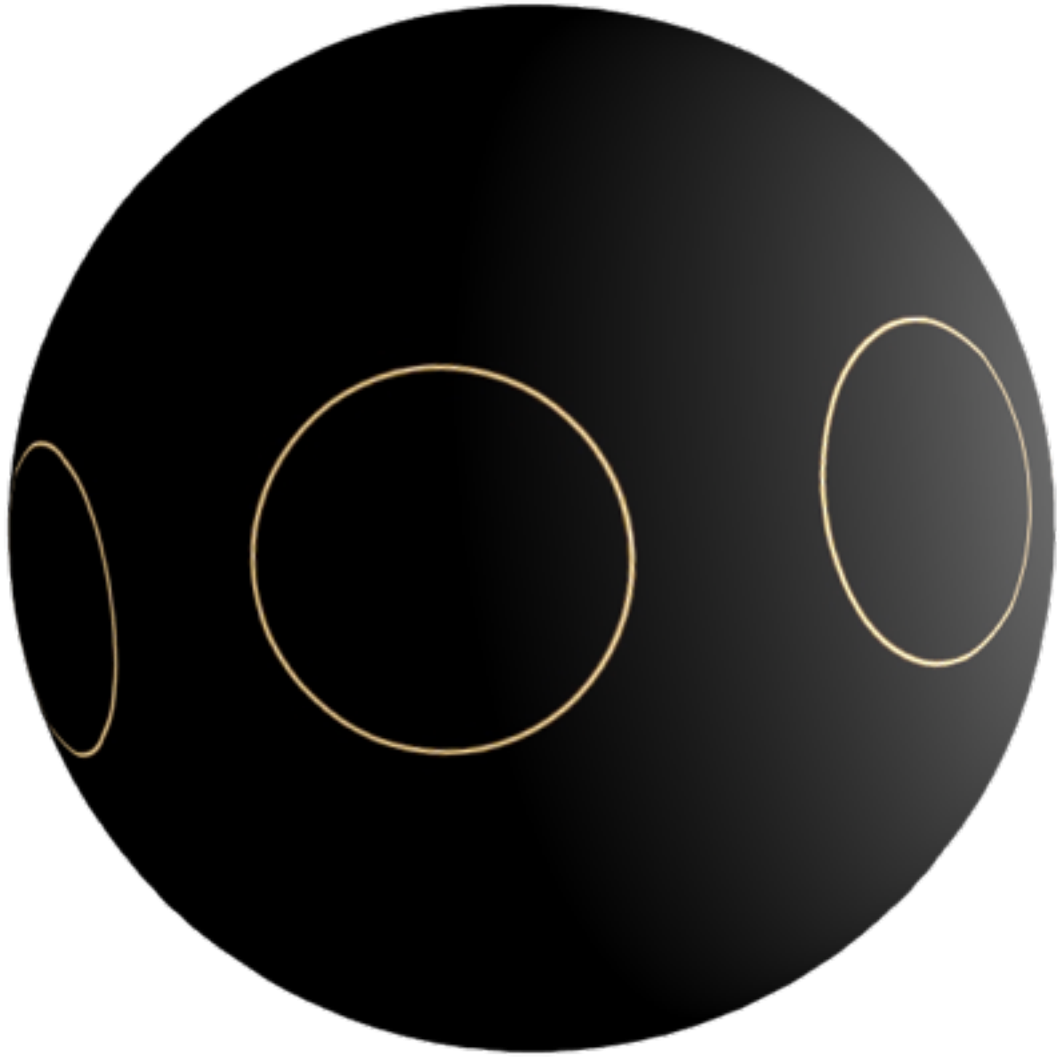
Diffuse (Texture)



Normal

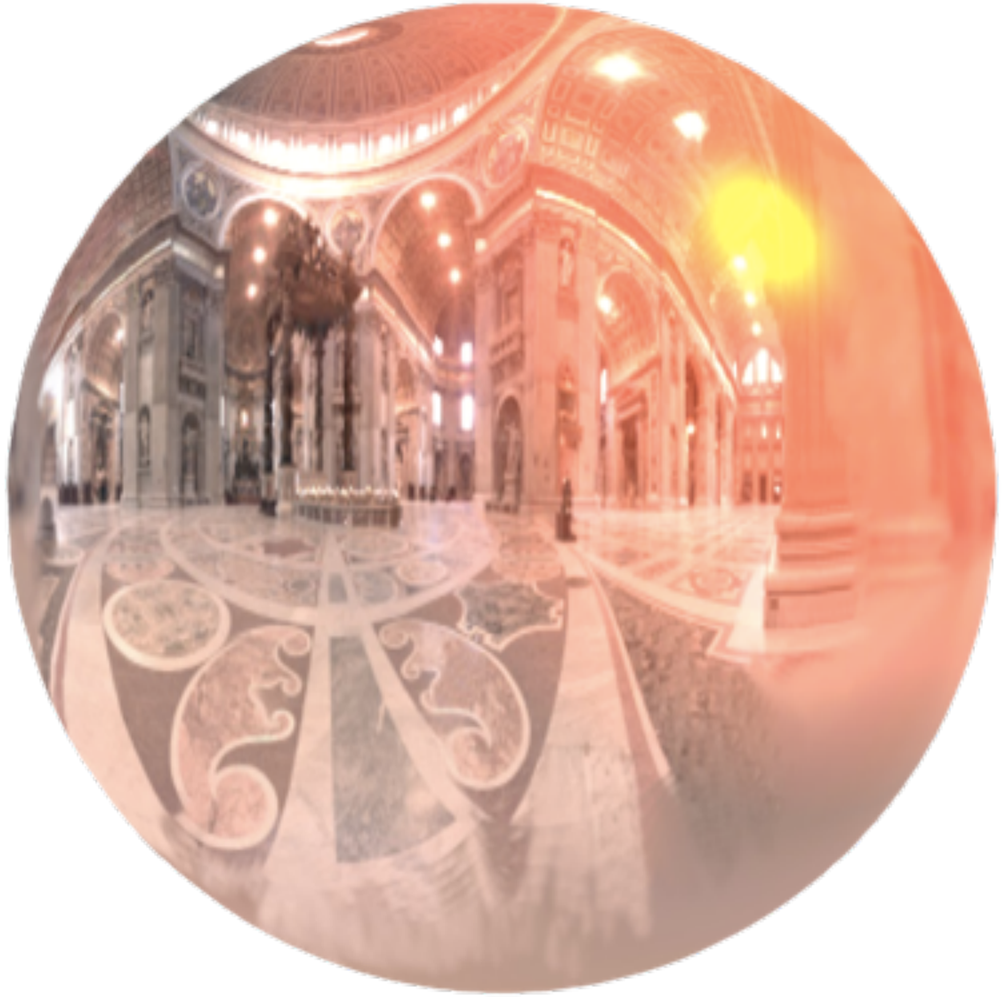


Emissive

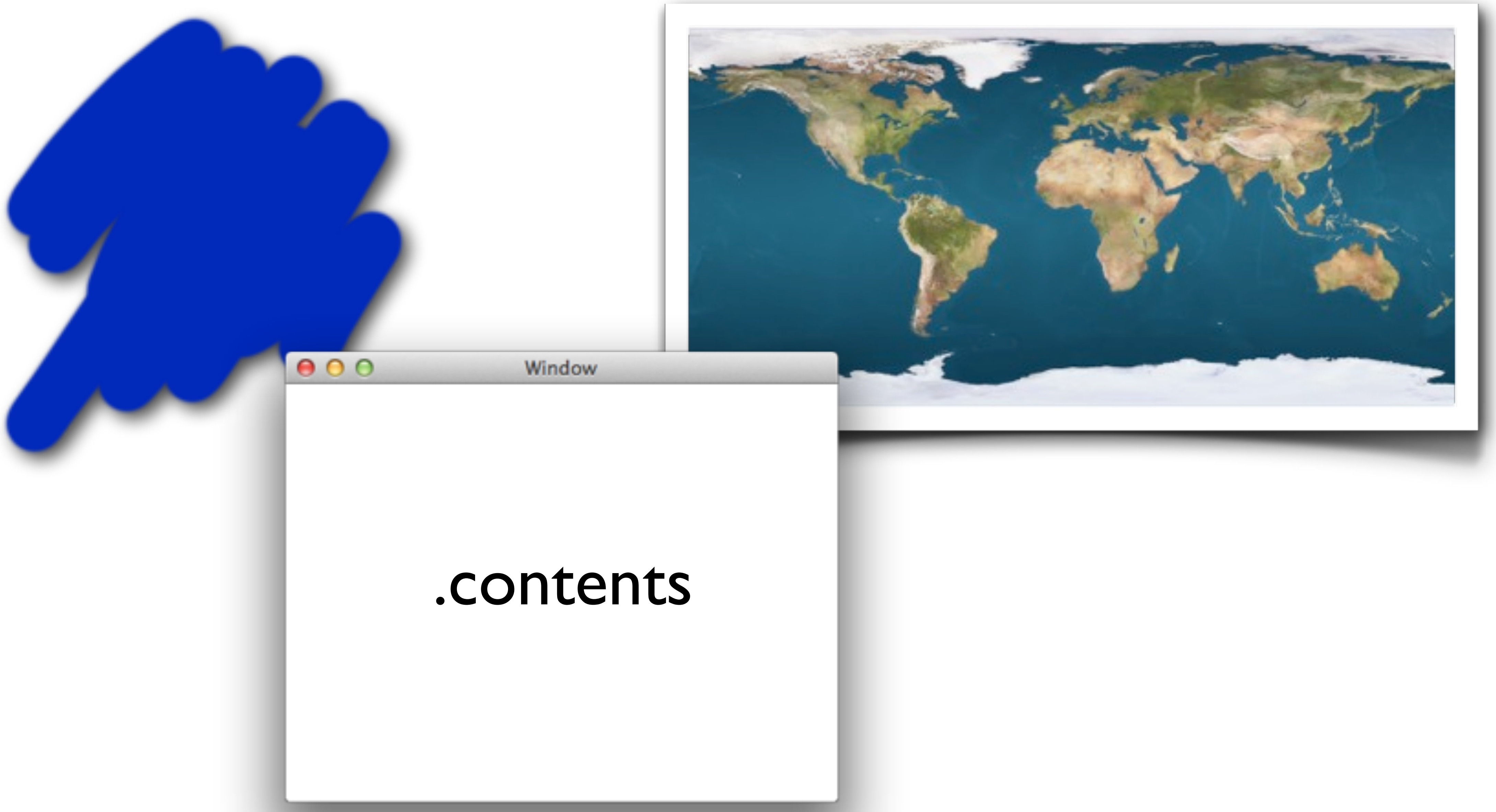


Transparency

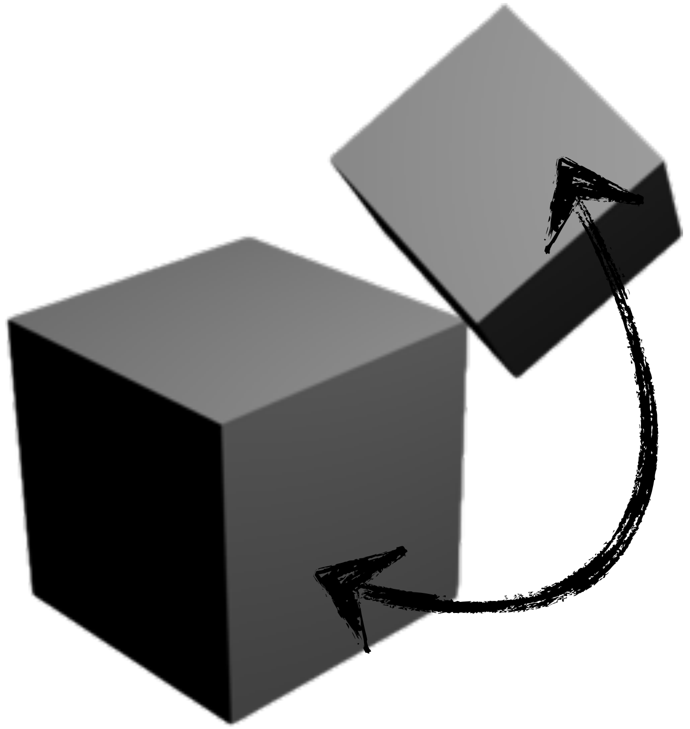
Reflective



SCNMaterialProperty



.contents



Demo

Animationen

- Implizite Animationen
- Explizite Animationen
  - Basic Animations
  - Keyframe Animations
  - Animation Groups

Implizite Animation

```
[SCNTransaction begin];
[SCNTransaction setAnimationDuration:2.0];

node.position = SCNVector3Make(0.0, 2.0, 3.0);

[SCNTransaction commit];
```

Explizite Animationen

```
CAAnimation* animation = ...
SCNNode* node = ...

[node addAnimation: animation forKey: @“MyAnimation“];
```

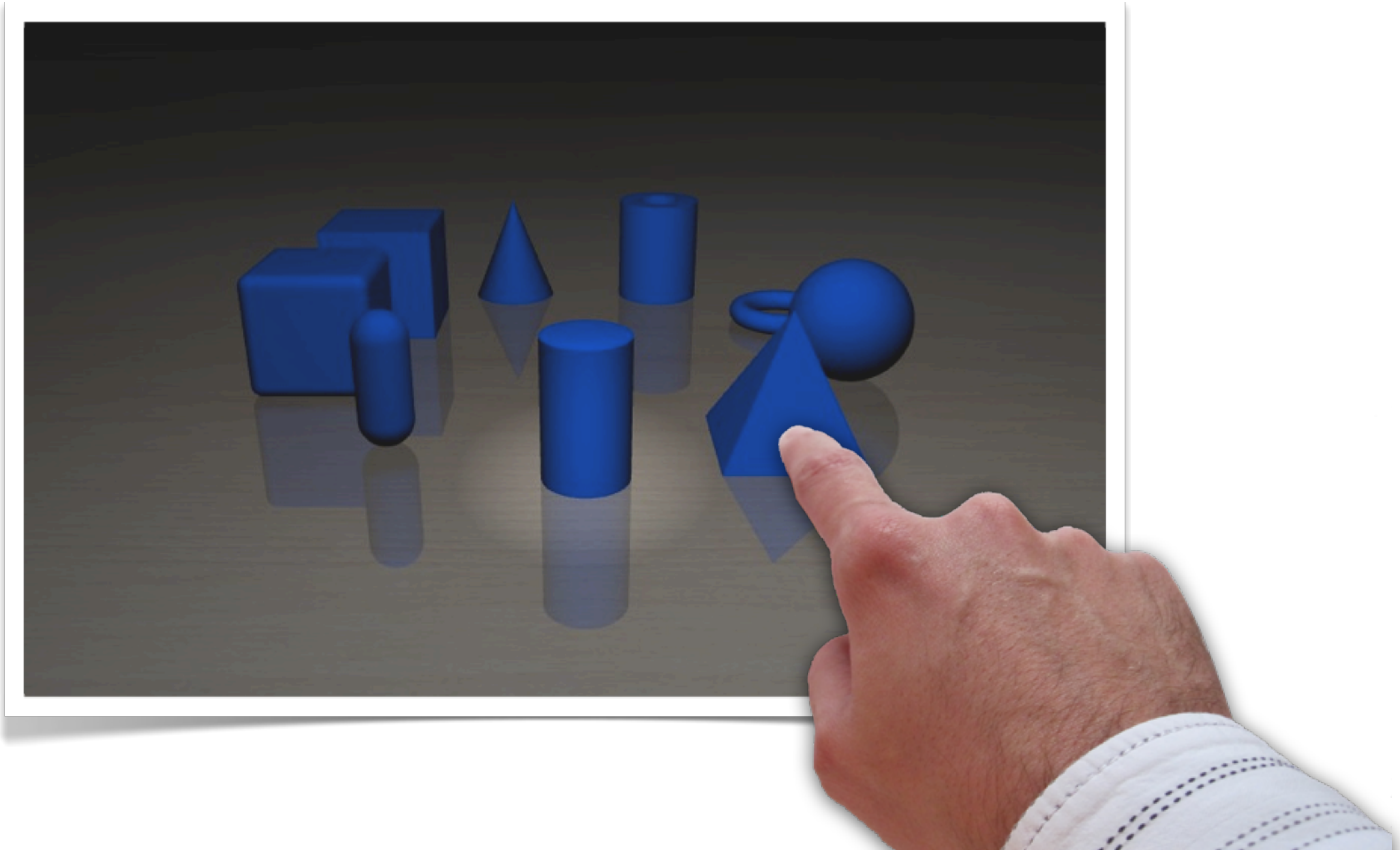
SCNSceneSource

```
SCNSceneSource* sceneSource = [SCNSceneScource sceneSourceWithURL:sceneURL
                                                          options:nil];
CAAnimation* animation = [sceneSource entryWithIdentifier:@“animationName“
                                                withClass:[CAAnimation class]];

[node addAnimation:animation forKey:nil];
```

Demo

Picking



SCNHitTestResult

```
-(void) mousePressedAtPoint:(CGPoint) point {
  NSArray* hitTestResults = [self.sceneView hitTest:point
                                            options:NULL];

  SCNNode* node = [[hitTestResults objectAtIndex:0] node];
  ... //do something special
}
```

Demo

Erweiterte Funktionen

Erweiterte Funktionen

- Eigene Renderer
- Malen auf 3D-Objekten

SCNProgram

- GL Shading Language
- Vertex Shader
- Fragment Shader
- Zugewiesen zu Material

Program Semantics

```
SCNProgram* program = [SCNProgram program];

program.vertexShader = ...;
program.fragmentShader = ...;

[program setSemantic: SCNGeometrySourceSemanticVertex
           forSymbol: @“a_Position“
             options: nil];
...
[program setSemantic: SCNModelViewProjectionTransform
           forSymbol: @“u_ModelViewProjectionMatrix“
             options: nil];
```

Program Delegate

```
- (BOOL)program:(SCNProgram*)program bindValueForSymbol:(NSString*) symbol
                                             atLocation:(unsigned int) location
                                              programID:(unsigned int) programID
                                               renderer:(SCNRenderer*) renderer {
  if ([symbol isEqualToString:@“u_Texture]) {
    glUniform1i(location, 0);
    return YES;
  }
  ...

  return NO;
}
```

Demo

content = CALayer

```
NSImage* image = ... //get image from somewhere

SCNMaterial* material = [SCNMaterial material];

CALayer* layer = [CALayer layer];
layer.contents = image;

material.diffuse.contents = layer;
```

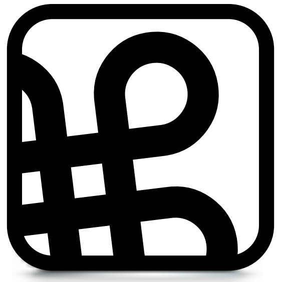
SCNHitTestResult - Textur

```
-(void) mousePressedAtPoint:(CGPoint) point {
  NSArray* hitTestResults = [self.sceneView hitTest:point
                                            options:NULL];

  SCNHitTestResult* hitTestResult = [hitTestResults objectAtIndex:0];
  CGPoint* point = [hitTestResult textureCoordinatesWithMappingChannel: 0];
  ... //Coordinate ranges: 0.0 - 1.0
}
```

Fragen?

Vielen Dank



Macoun