Graphics and Games

#WWDC16

Advances in SceneKit Rendering

Session 609

Amaury Balliet SceneKit Engineer
Jean-Baptiste Bégué SceneKit Engineer
Sébastien Métrot SceneKit Engineer
Nick Porcino Model I/O Engineer

Agenda

SceneKit in a Nutshell

Rendering Advances

Demo

Behind the Demo

Camera Effects

Model I/O

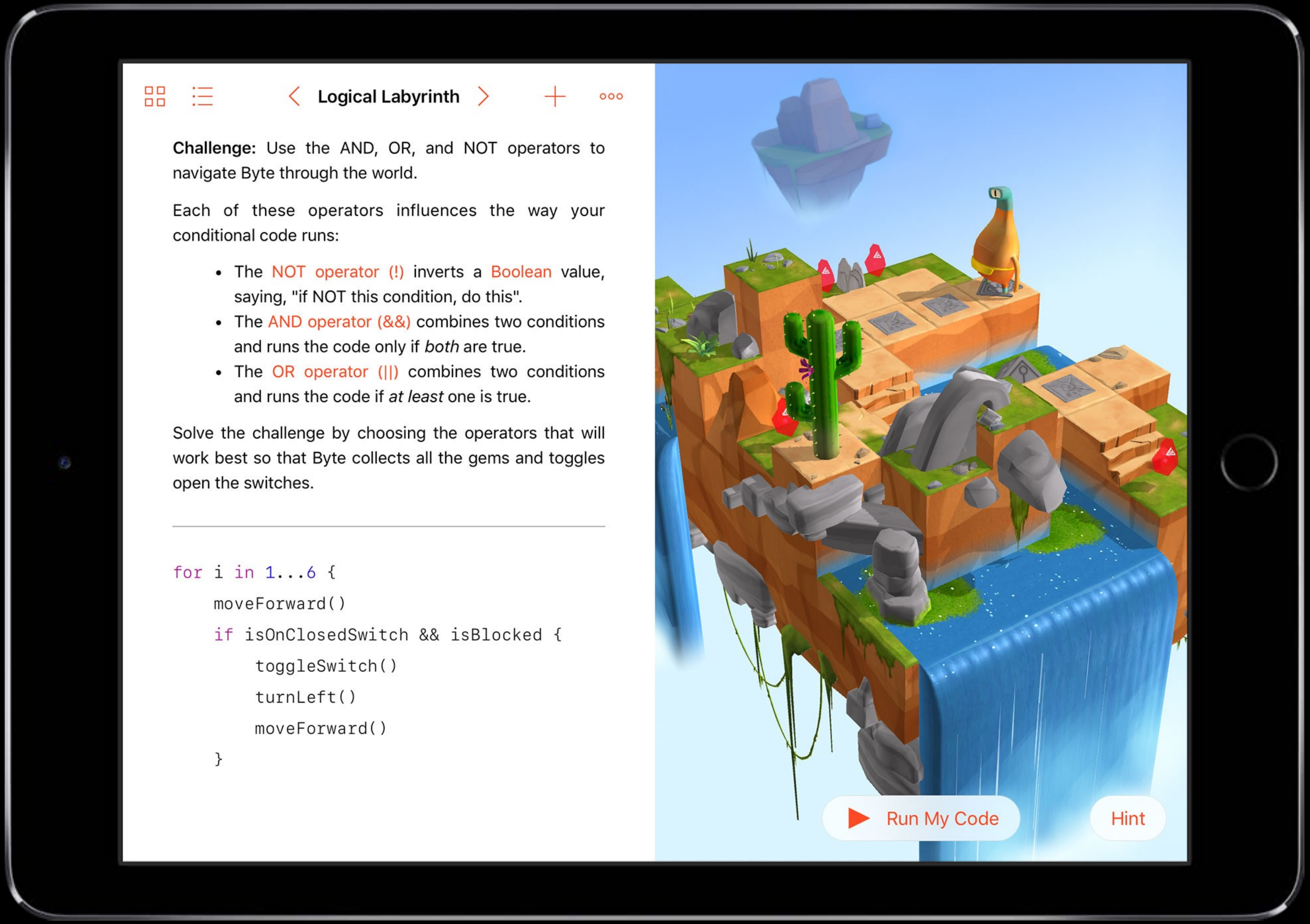
SceneKit

In a nutshell

Amaury Balliet SceneKit Engineer

SceneKit

SceneKit



Challenge: Use the AND, OR, and NOT operators to navigate Byte through the world.

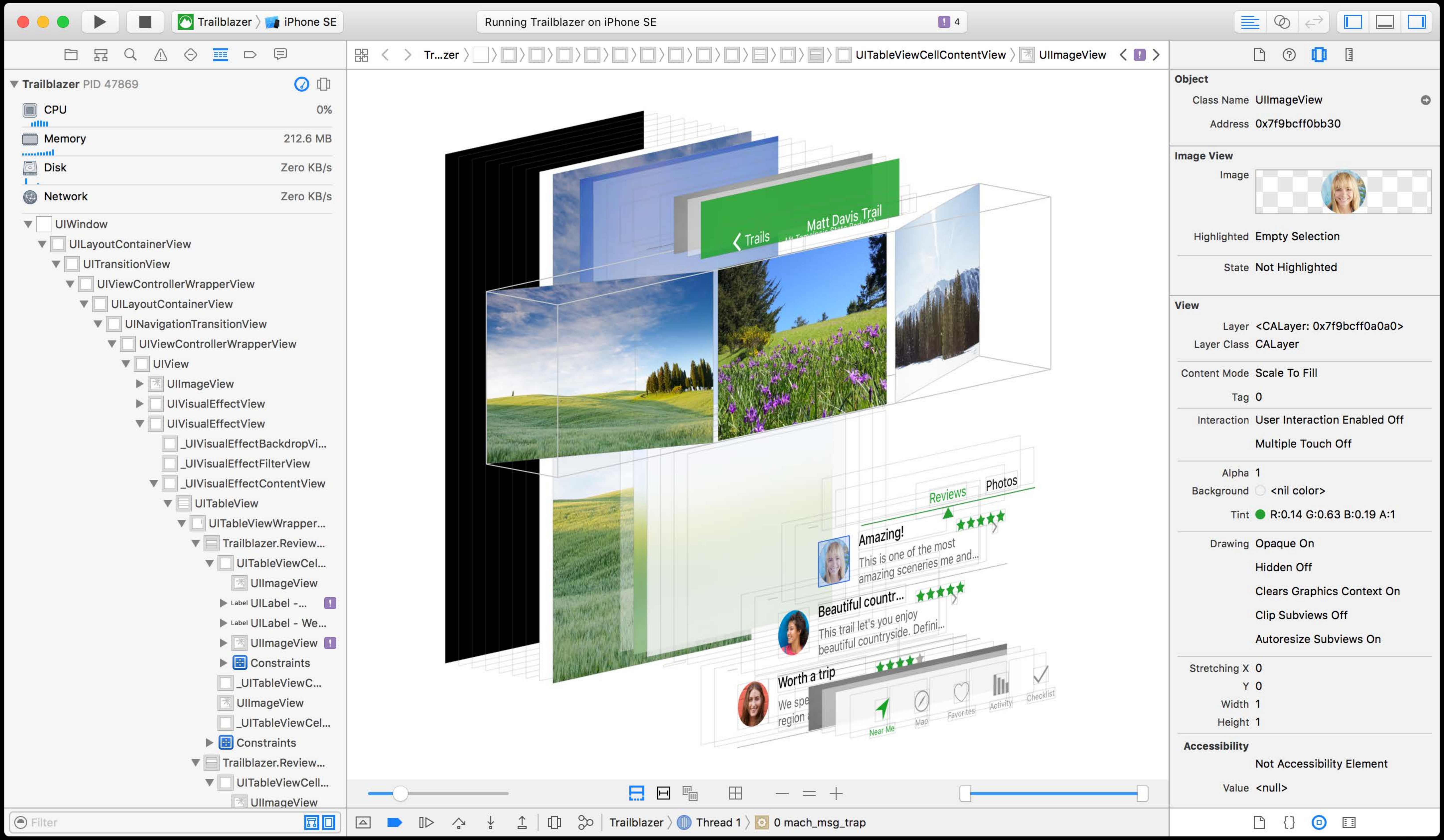
Each of these operators influences the way your conditional code runs:

- The NOT operator (!) inverts a Boolean value, saying, "if NOT this condition, do this".
- The AND operator (&&) combines two conditions and runs the code only if *both* are true.
- The OR operator (||) combines two conditions and runs the code if *at least* one is true.

Solve the challenge by choosing the operators that will work best so that Byte collects all the gems and toggles open the switches.

```
for i in 1...6 {
    moveForward()
    if isOnClosedSwitch && isBlocked {
        toggleSwitch()
        turnLeft()
        moveForward()
    }
```

SceneKit



Running Trailblazer on iPhone SE

Trailblazer PID 47869

CPU 0%

Memory 212.6 MB

Disk Zero KB/s

Network Zero KB/s

Object

Class Name UIImageView

Address 0x7f9bcff0bb30

Image View

Image

Highlighted Empty Selection

State Not Highlighted

View

Layer <CALayer: 0x7f9bcff0a0a0>

Layer Class CALayer

Content Mode Scale To Fill

Tag 0

Interaction User Interaction Enabled Off

Multiple Touch Off

Alpha 1

Background <nil color>

Tint R:0.14 G:0.63 B:0.19 A:1

Drawing Opaque On

Hidden Off

Clears Graphics Context On

Clip Subviews Off

Autoresize Subviews On

Stretching X 0

Y 0

Width 1

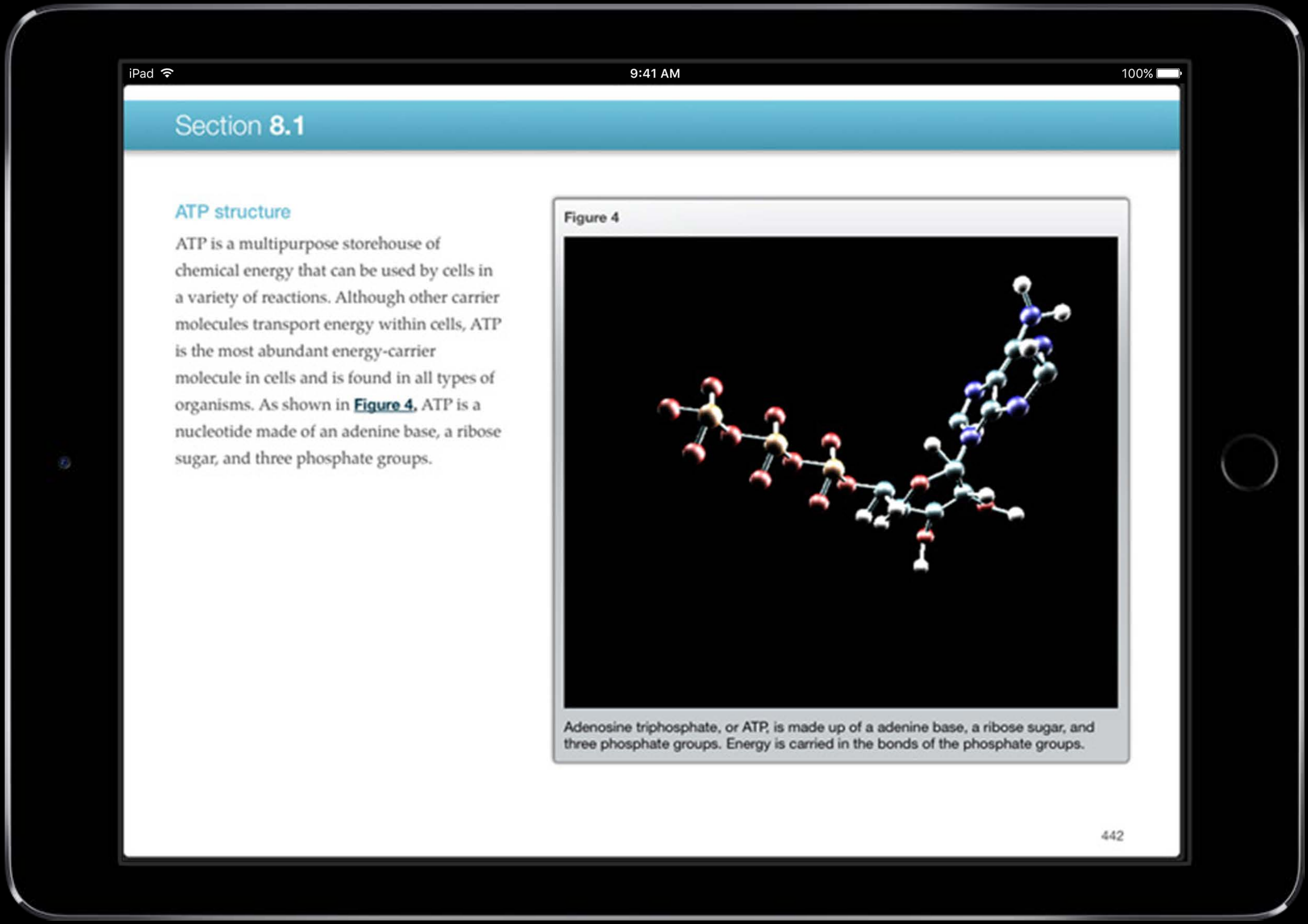
Height 1

Accessibility

Not Accessibility Element

Value <null>

SceneKit



Section 8.1

ATP structure

ATP is a multipurpose storehouse of chemical energy that can be used by cells in a variety of reactions. Although other carrier molecules transport energy within cells, ATP is the most abundant energy-carrier molecule in cells and is found in all types of organisms. As shown in **Figure 4**, ATP is a nucleotide made of an adenine base, a ribose sugar, and three phosphate groups.

Figure 4

Adenosine triphosphate, or ATP, is made up of a adenine base, a ribose sugar, and three phosphate groups. Energy is carried in the bonds of the phosphate groups.

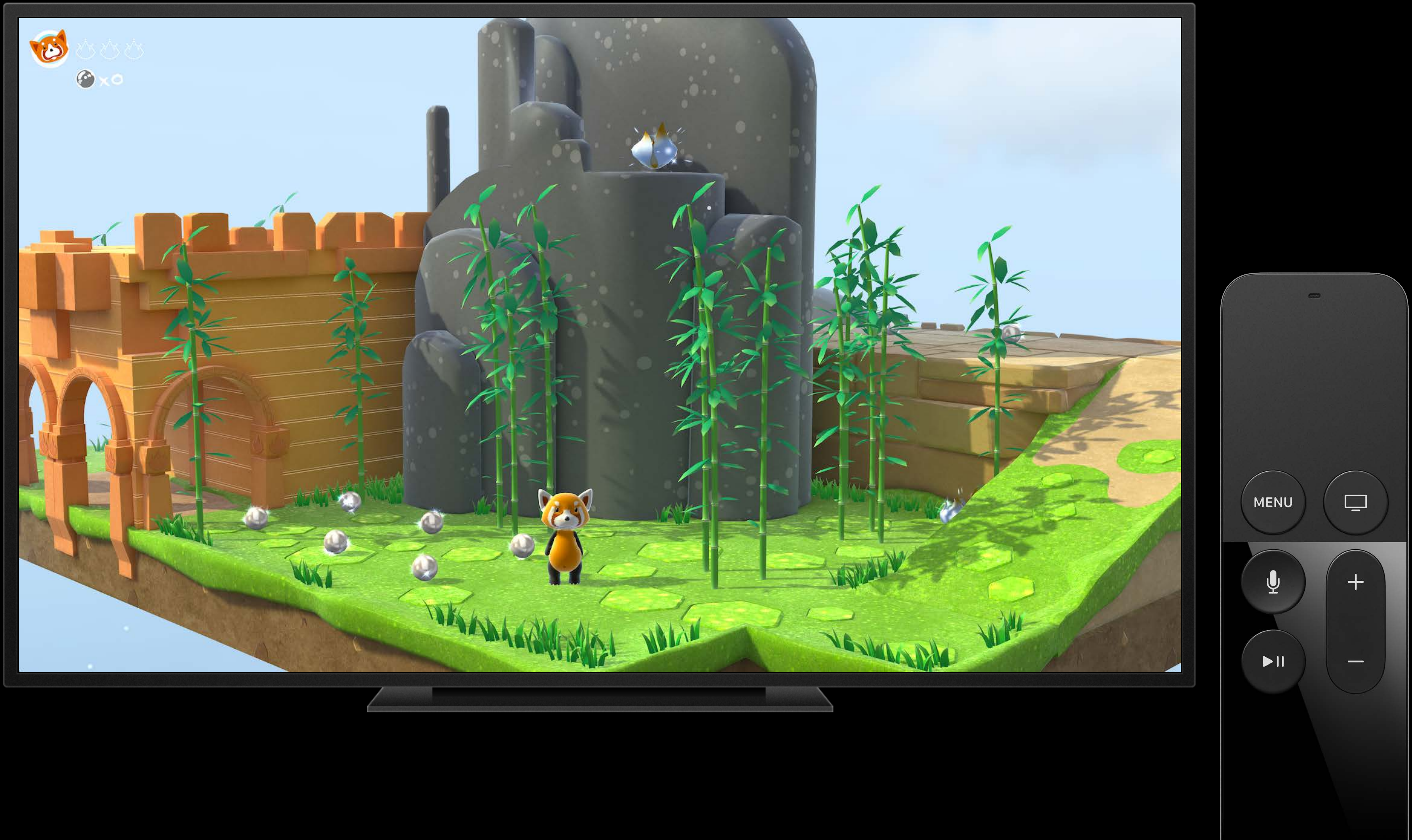
442

SceneKit

Thank you!

macOS iOS

macOS iOS tvOS


×0
MENU

macOS iOS tvOS

macOS iOS tvOS watchOS

SceneKit

watchOS 3

SceneKit is now available everywhere

Great opportunity to make attractive apps

New interactions with content on your wrist

SceneKit

watchOS 3

SceneKit is now available everywhere

Great opportunity to make attractive apps

New interactions with content on your wrist

Game Technologies for Apple Watch | Mission | Friday 3:00PM

SceneKit

watchOS 3

SceneKit is now available everywhere

Great opportunity to make attractive apps

New interactions with content on your wrist

Game Technologies for Apple Watch	Mission	Friday 3:00PM

SceneKit

What's New in SceneKit	WWDC 2013
What's New in SceneKit	WWDC 2014
Enhancements to SceneKit	WWDC 2015

Advances in SceneKit Rendering

Physically based rendering

Physically based rendering
in the hands of everyone.

FOX
FIRE DEPT.
ENGINE N

Advances in SceneKit Rendering

Biggest leap forward since SceneKit's introduction

Latest advances in 3D graphics

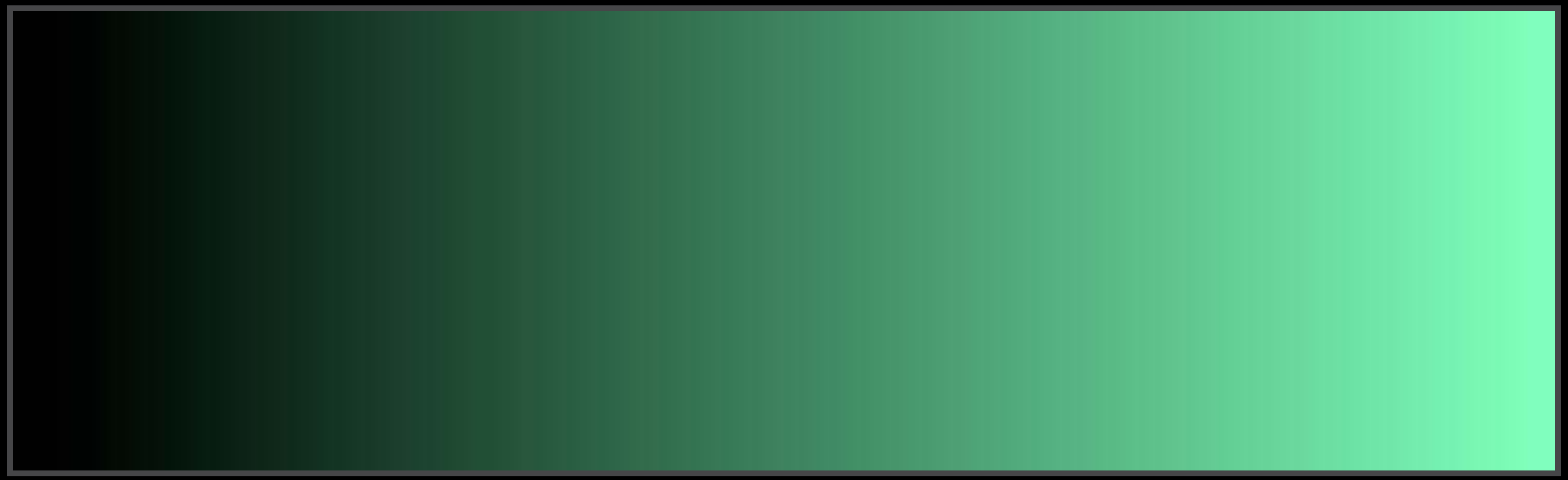
Modern technologies

- Accurate rendering
- Physically based materials and lighting

Accurate Rendering

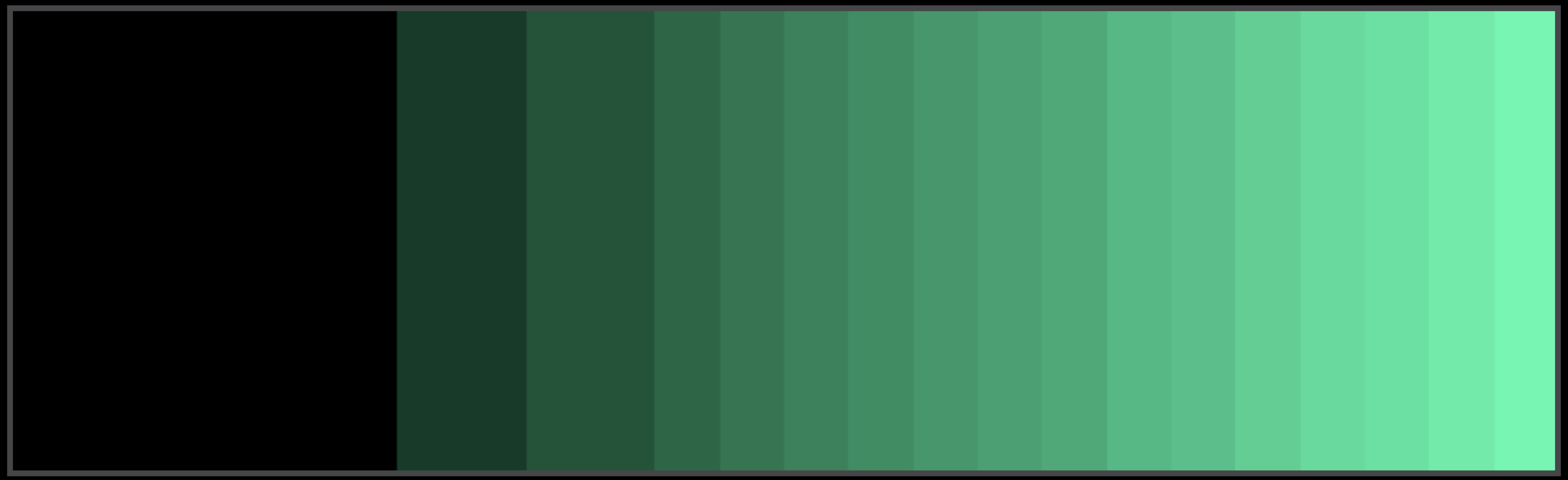
Linear Rendering and Color Management

Linear rendering



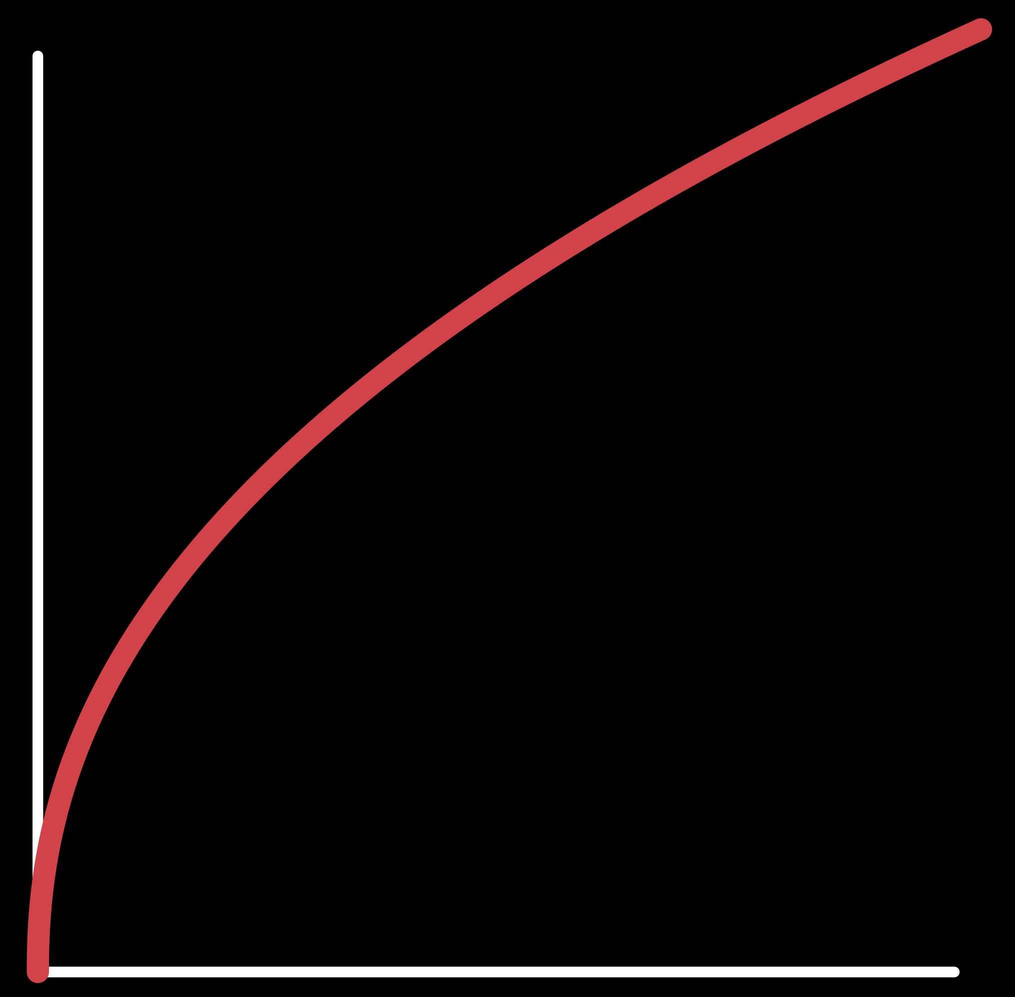
Linear Rendering and Color Management

Linear rendering



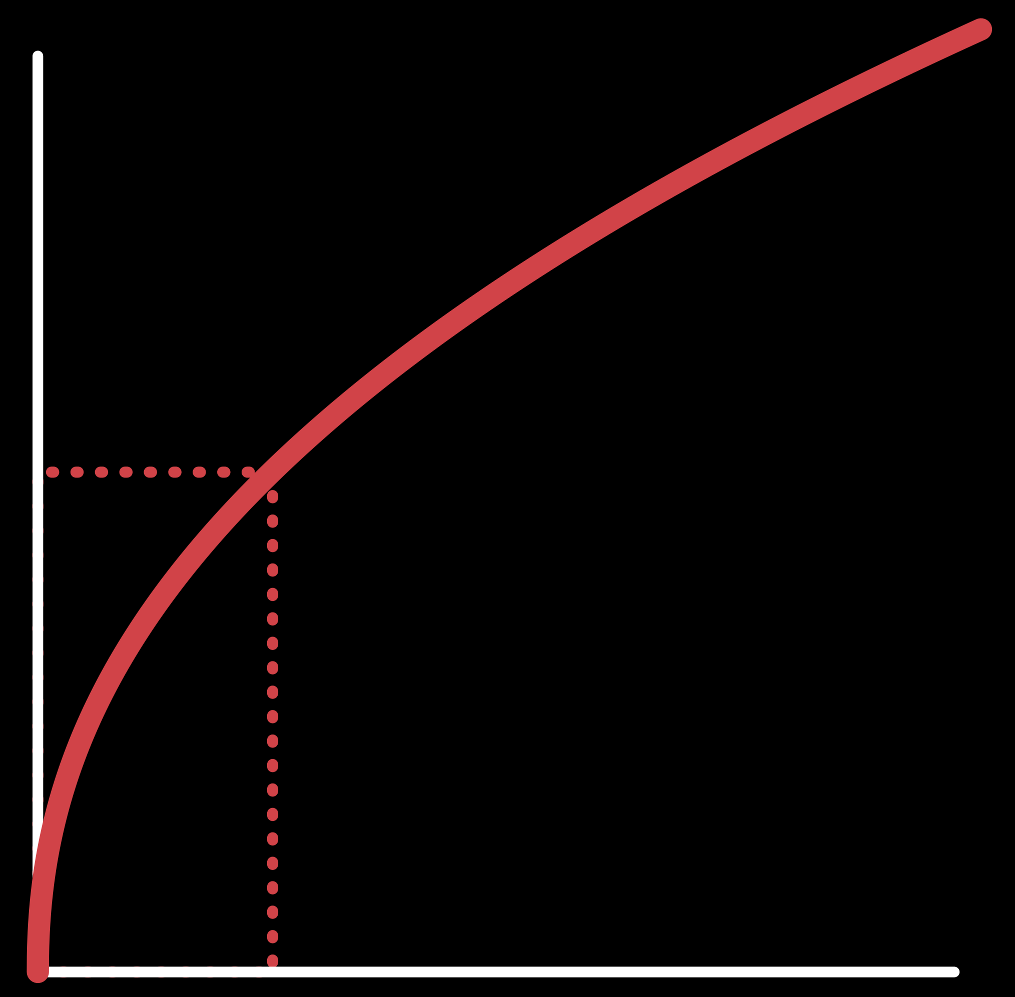
Linear Rendering and Color Management

Linear rendering



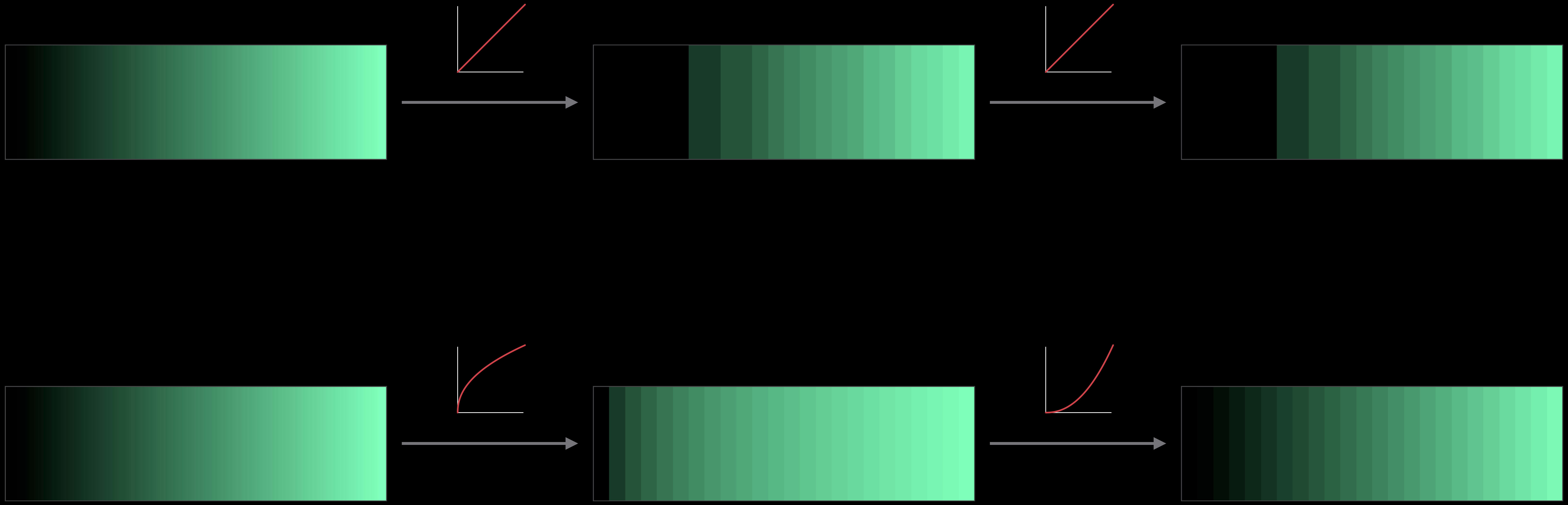
Linear Rendering and Color Management

Linear rendering



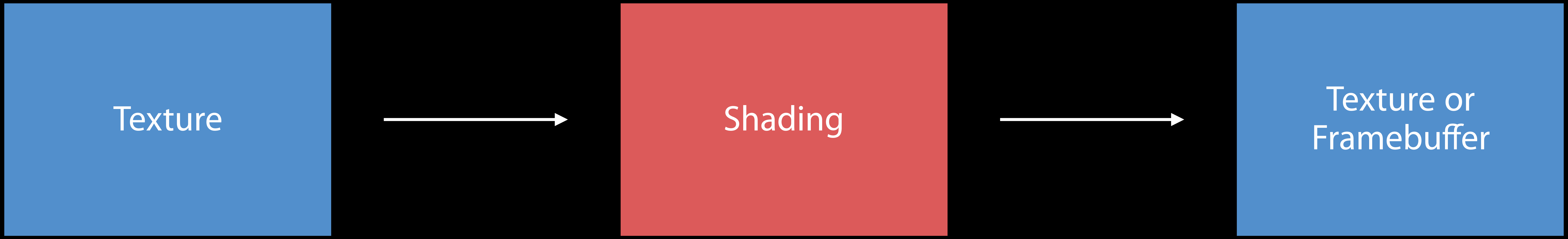
Linear Rendering and Color Management

Linear rendering



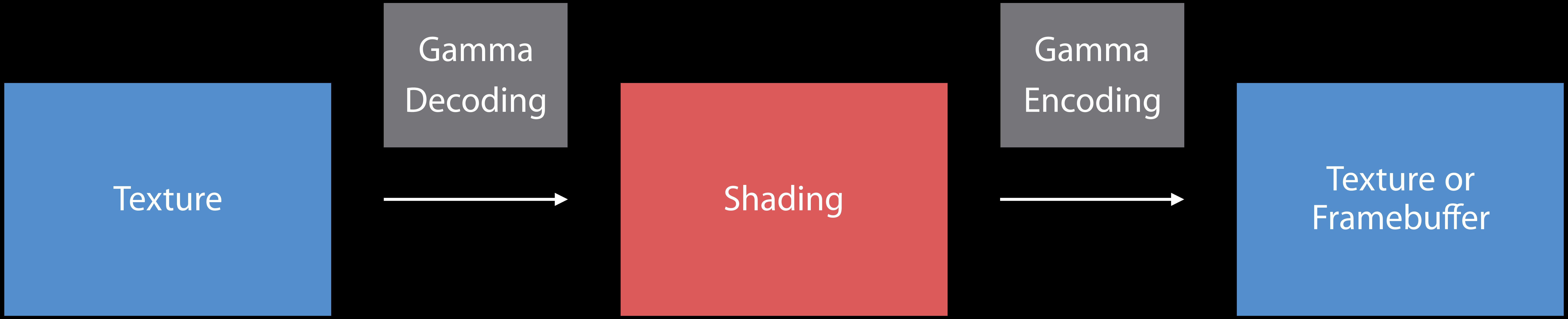
Linear Rendering and Color Management

Shading in gamma space



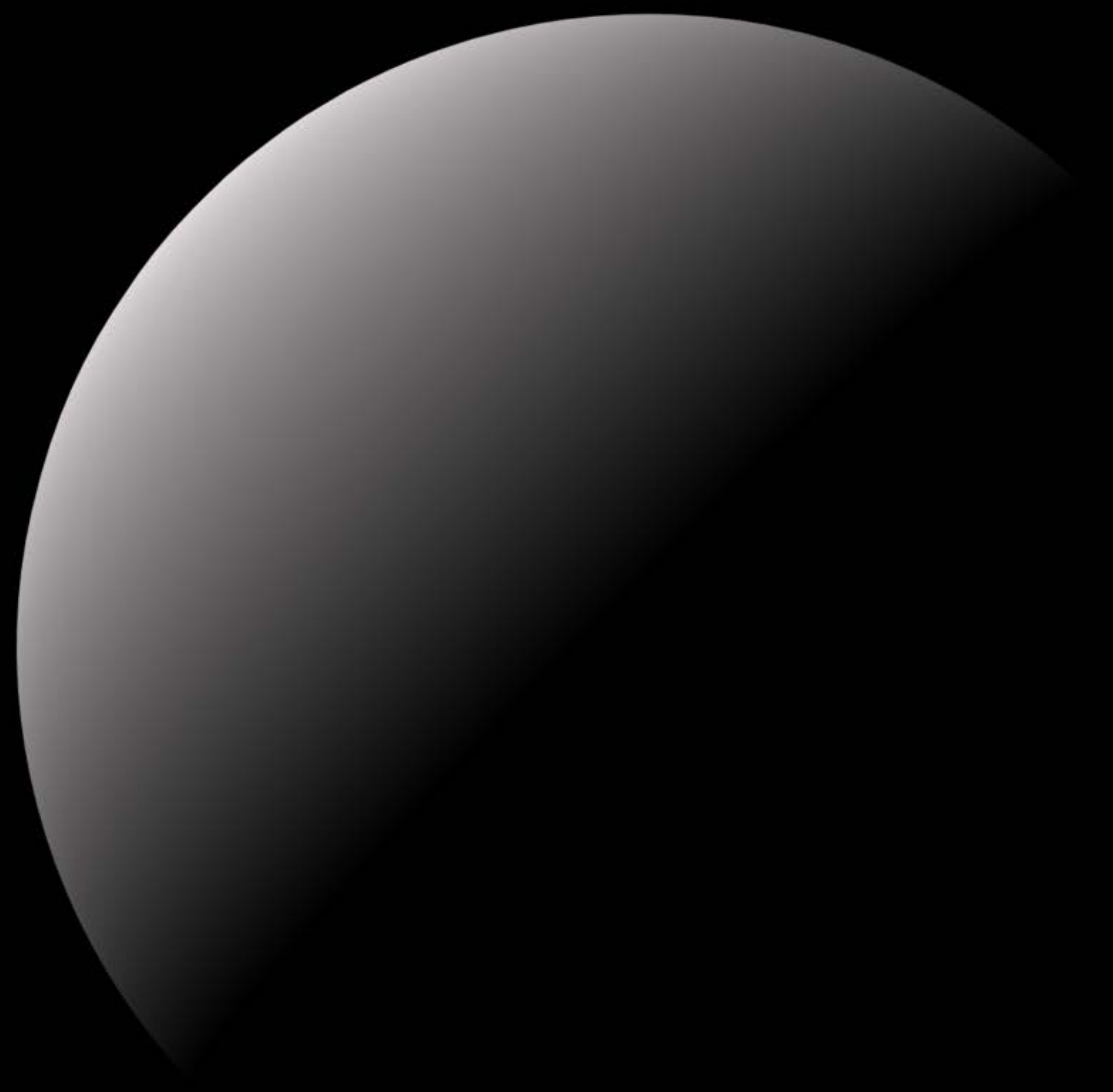
Linear Rendering and Color Management

Shading in linear space



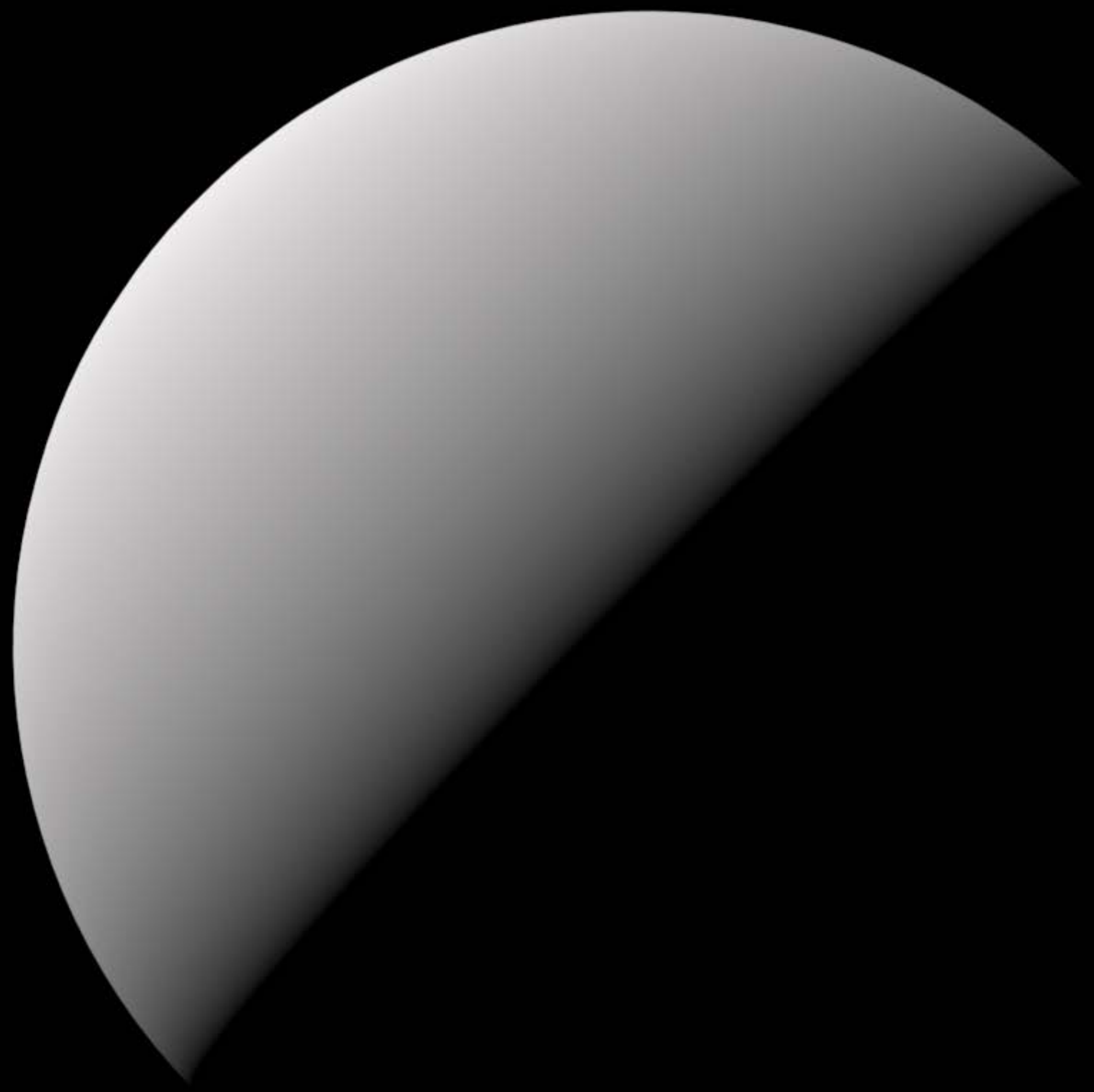
Linear Rendering and Color Management

Shading in gamma space



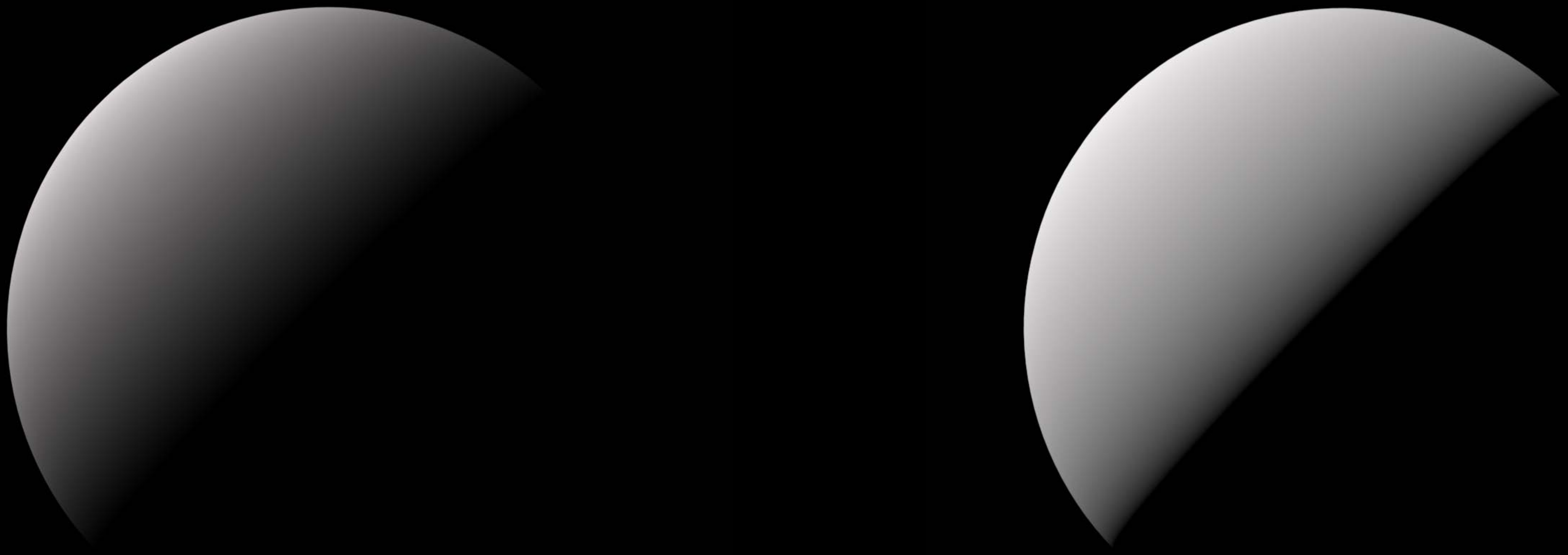
Linear Rendering and Color Management

Shading in linear space



Linear Rendering and Color Management

Linear rendering



Linear Rendering and Color Management

Linear rendering

Essential for physically based shading

Being linear is necessary to get the math right

Benefits to all other lighting models

Linear Rendering and Color Management

Color management

Cross-framework effort for color accuracy

Fully embraced by SceneKit

Linear Rendering and Color Management

Color management for textures

Automatic color management for images

Textures that represent raw data are
supposed to be sRGB

Have a look at texture sets and asset catalogs

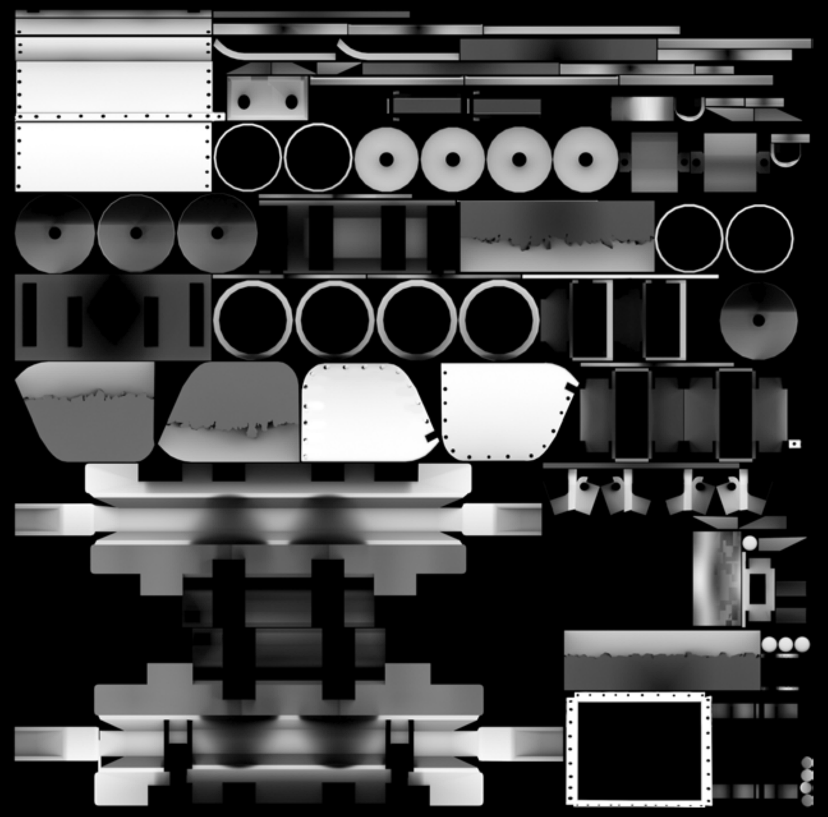
Linear Rendering and Color Management

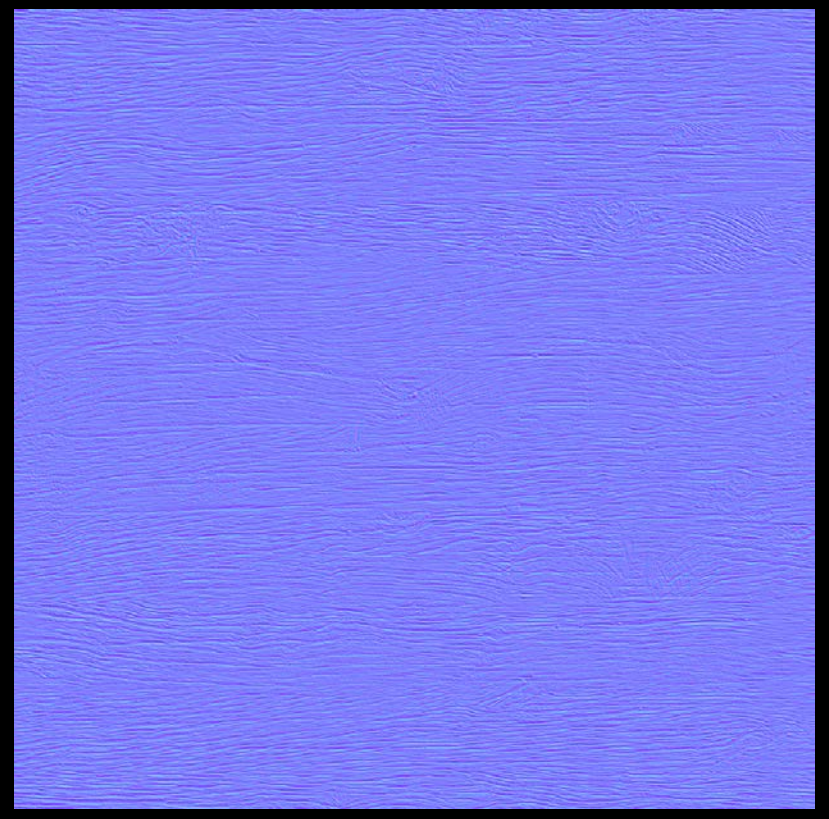
Color management for textures

Automatic color management for images

Textures that represent raw data are supposed to be sRGB

Have a look at texture sets and asset catalogs

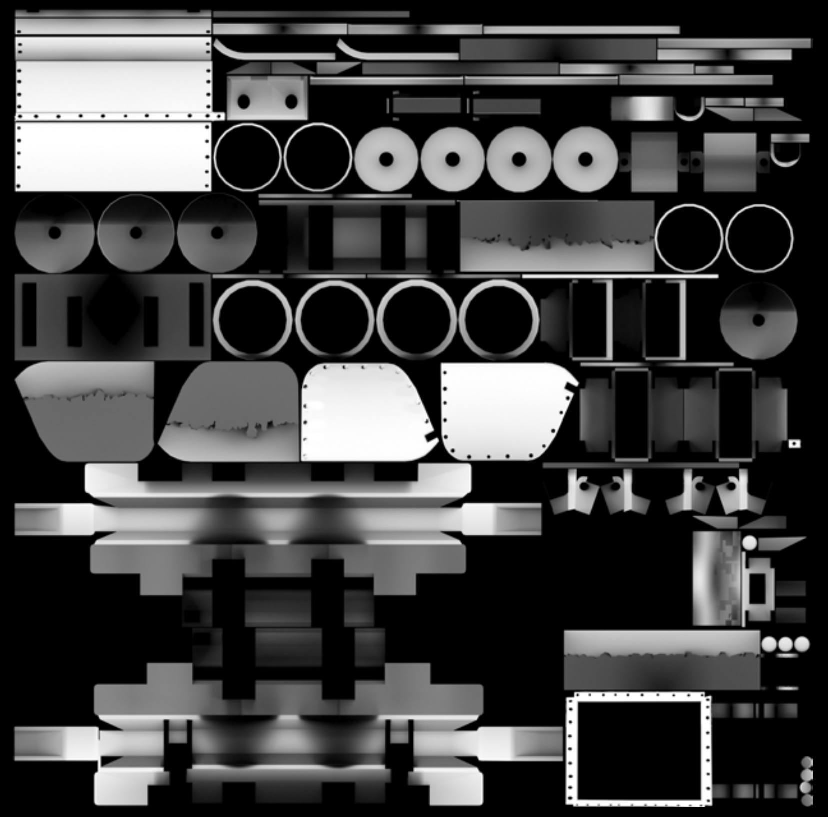


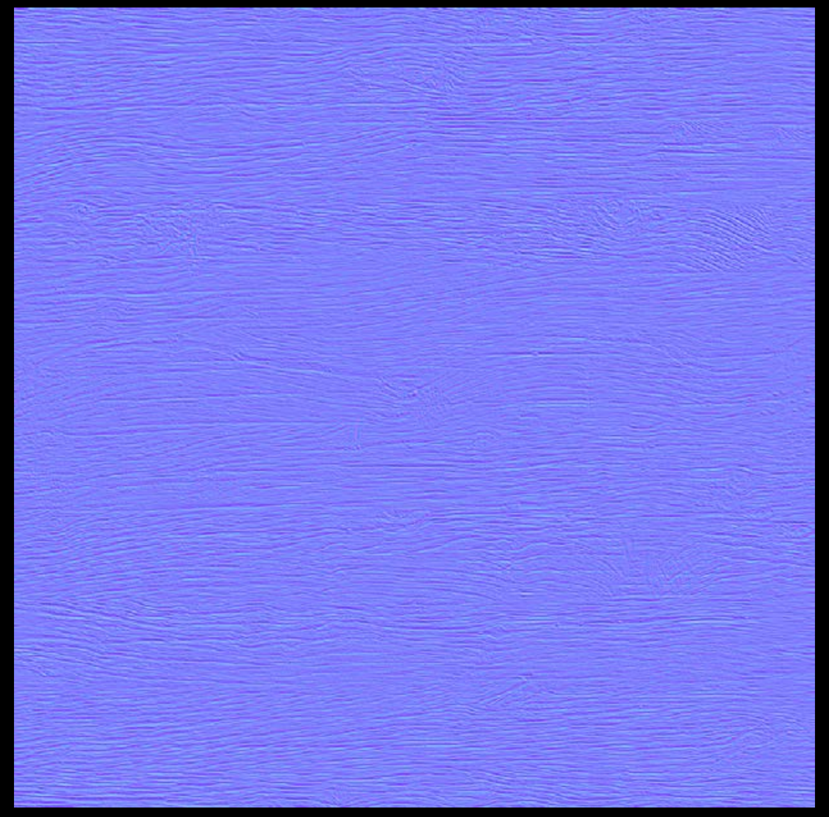


Linear Rendering and Color Management

Color management for textures

- Automatic color management for images
- Textures that represent raw data are supposed to be sRGB
- Have a look at texture sets and asset catalogs





Linear Rendering and Color Management

Color management for color objects

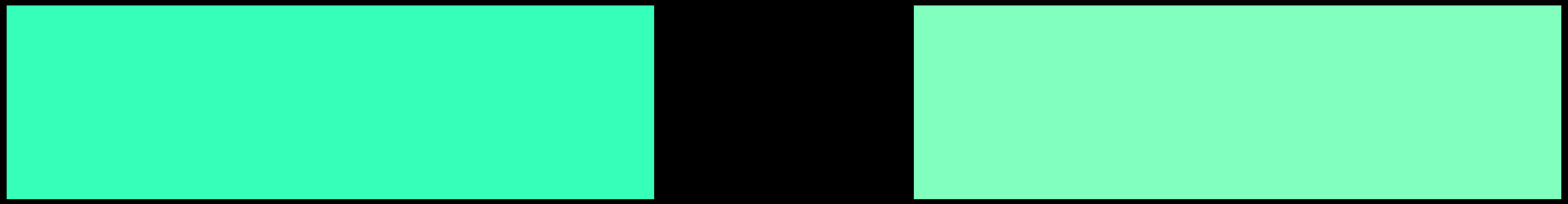
Automatic color management for color objects

Color components previously assumed to be sRGB

Be careful with programmatically-generated color objects

Linear Rendering and Color Management

Color management for color objects



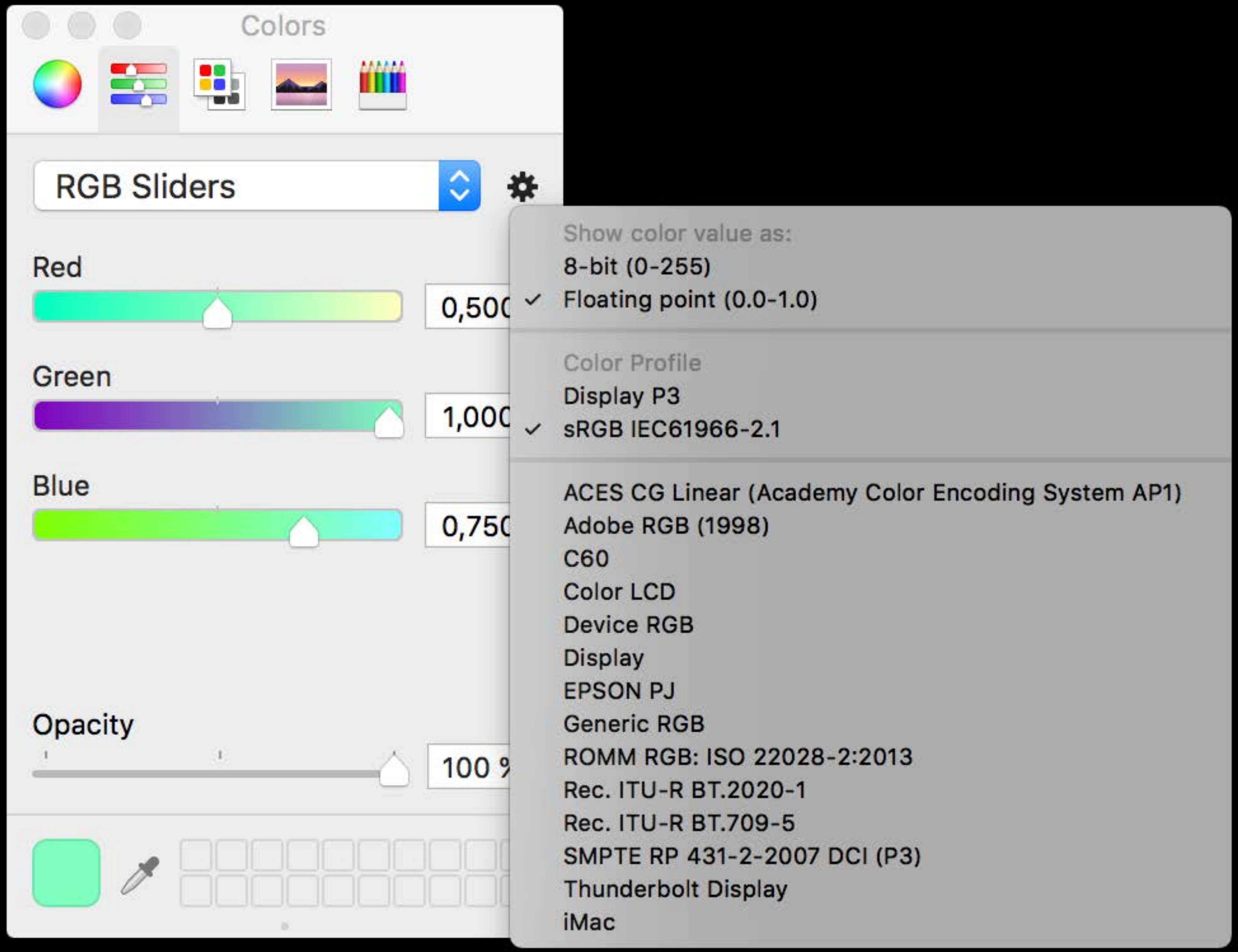
Display P3
(0.5, 1.0, 0.75)

sRGB
(0.5, 1.0, 0.75)

```
let colorA = NSColor(displayP3Red: 0.5, green: 1.0, blue: 0.75, alpha: 1) // Display P3
let colorB = NSColor(srgbRed: 0.5, green: 1.0, blue: 0.75, alpha: 1)      // sRGB
```

Linear Rendering and Color Management

Color management for color objects



Red 0,500
Green 1,000
Blue 0,750
Opacity 100 %

✓ Floating point (0.0-1.0)
✓ sRGB IEC61966-2.1

Linear Rendering and Color Management

Color management for color objects

Automatic color management for color objects

Color components previously assumed to be sRGB

Be careful with programmatically-generated color objects

Be careful with shader modifiers

```
// Metal Shading Language shader modifier
// linear extended sRGB components for sRGB(0.5, 1.0, 0.75)
_surface.diffuse.rgb += float3(0.235514164, 1.03112769, 0.523271978)
```

Linear Rendering and Color Management

Backward compatibility

No performance cost

Enabled when building against the new SDKs

Dramatic visual impact for older scenes

Linear Rendering and Color Management

Backward compatibility

Linear Rendering and Color Management

Backward compatibility

Linear Rendering and Color Management

Backward compatibility

Linear Rendering and Color Management

Backward compatibility

No performance cost

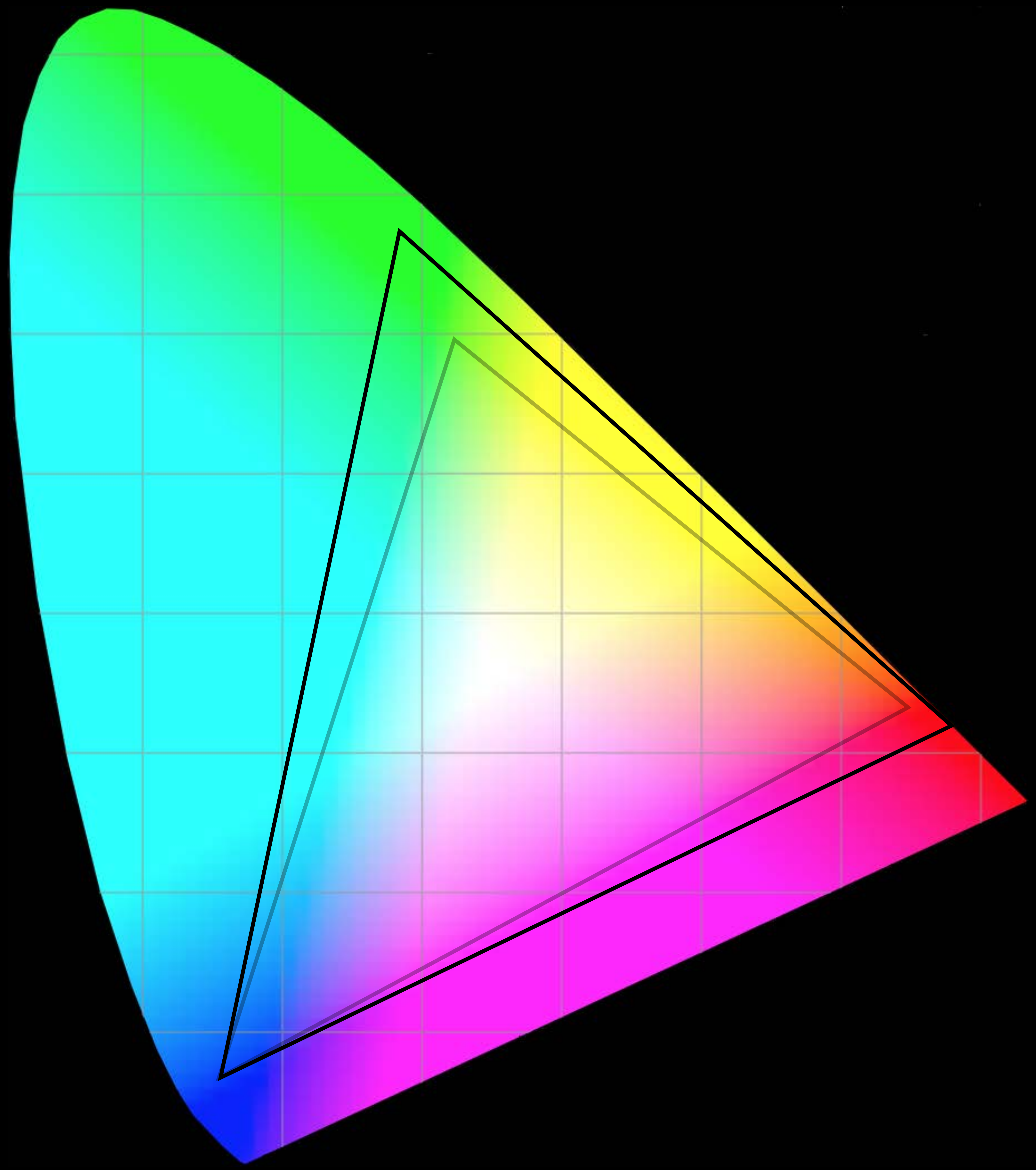
Enabled when building against the new SDKs

Dramatic visual impact for older scenes

Global option to opt-out

```
// Info.plist
<key>SCNDisableLinearSpaceRendering</key>
<true/>
```

Wide Gamut Content



Wide Gamut Content

Transparent support for wide gamut images and color

Full support of wide gamut displays

- 9.7-inch iPad Pro
- iMac with Retina display

Wide Gamut Content

Caveats

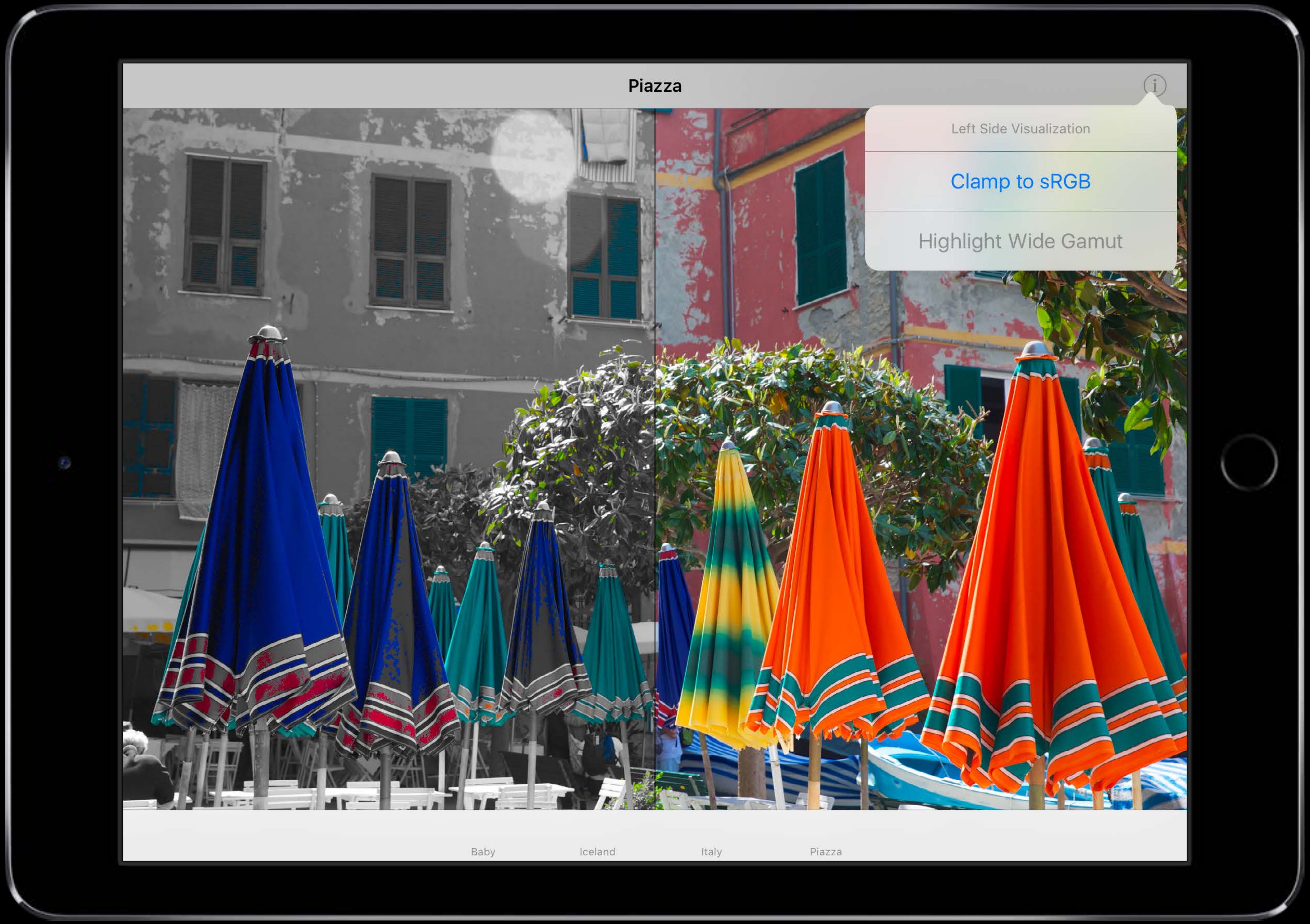
Increased memory usage

Global option to opt-out

```
// Info.plist
<key>SCNDisableWideGamut</key>
<true/>
```

Wide Gamut Content

“Color Gamut Showcase” sample code



Wide Gamut Content

Working with Wide Color	Mission	Thursday 1:40PM

Advances in SceneKit Rendering

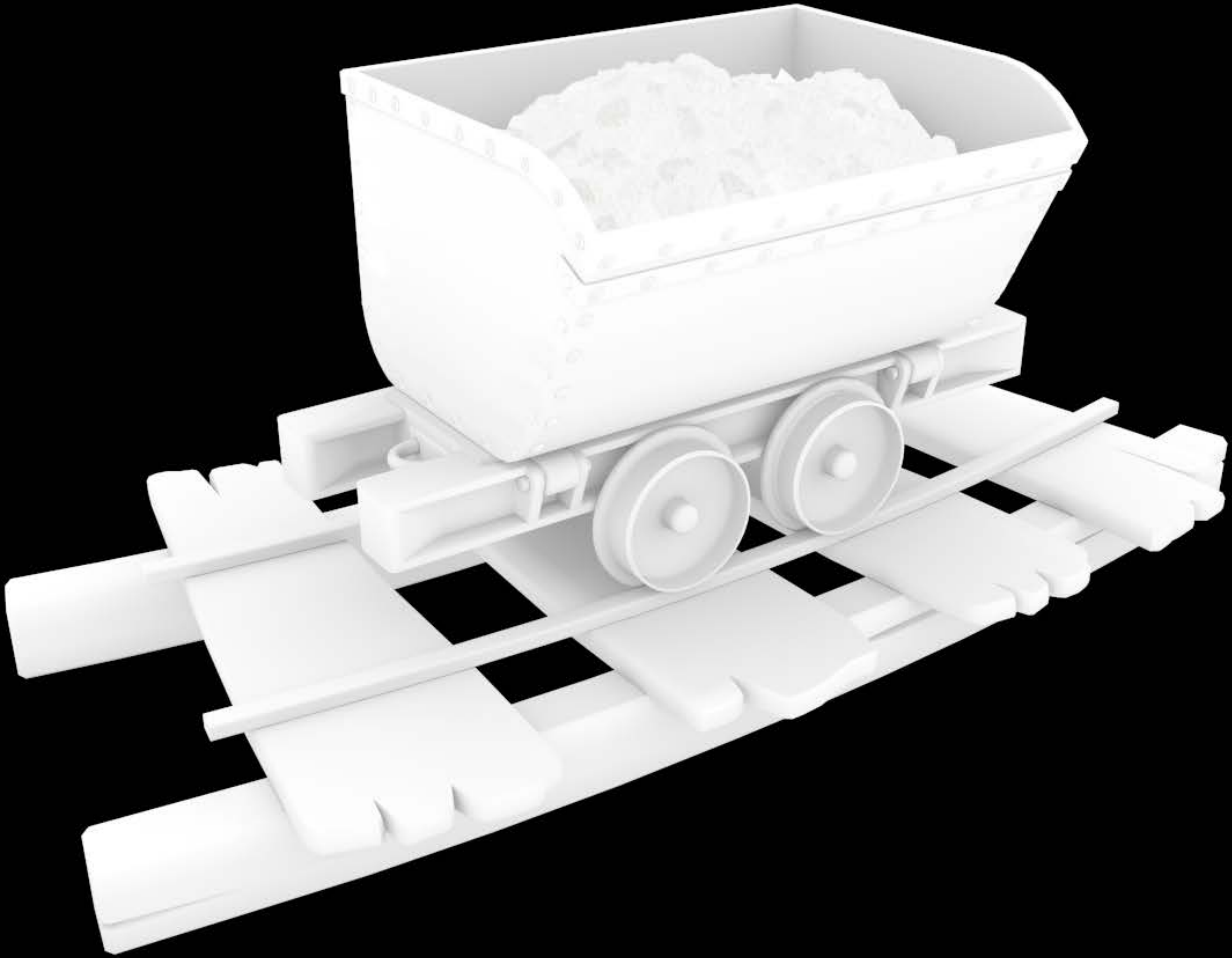
Biggest leap forward since SceneKit's introduction

Latest advances in 3D graphics

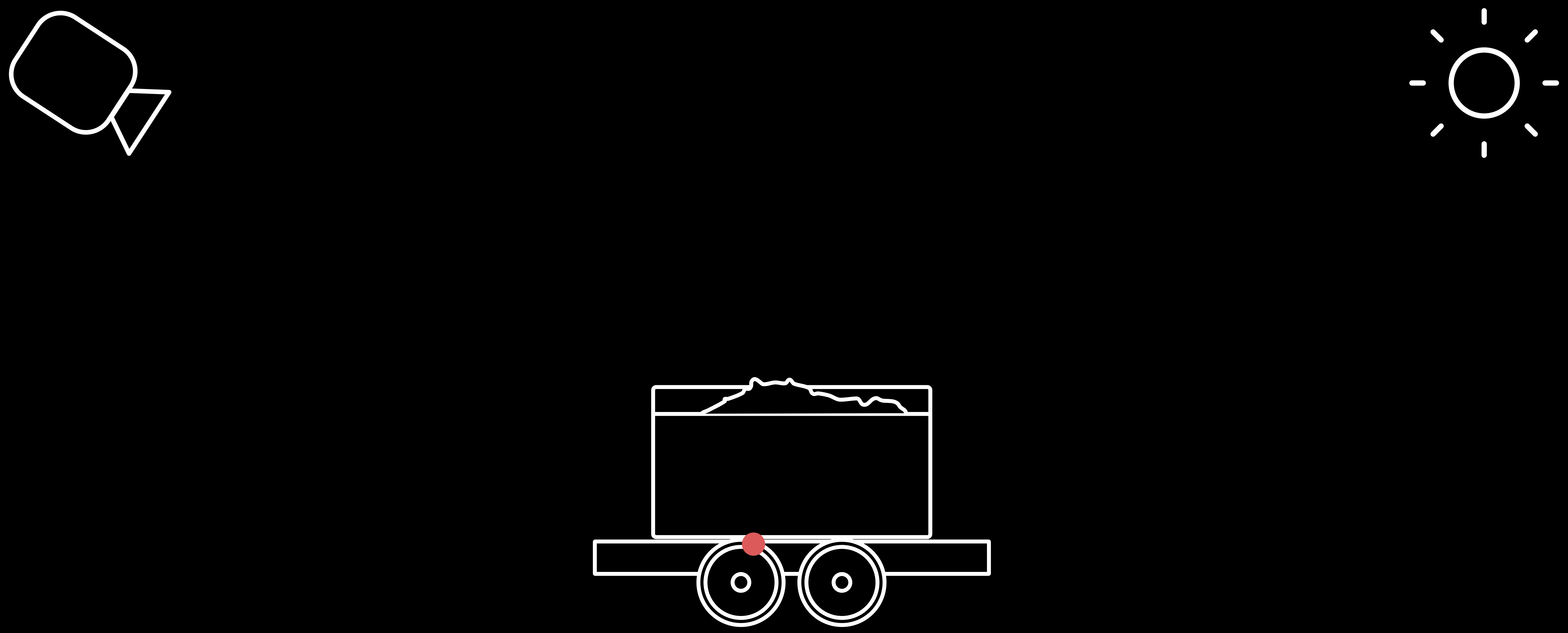
Modern technologies

- Accurate rendering
- Physically based materials and lighting

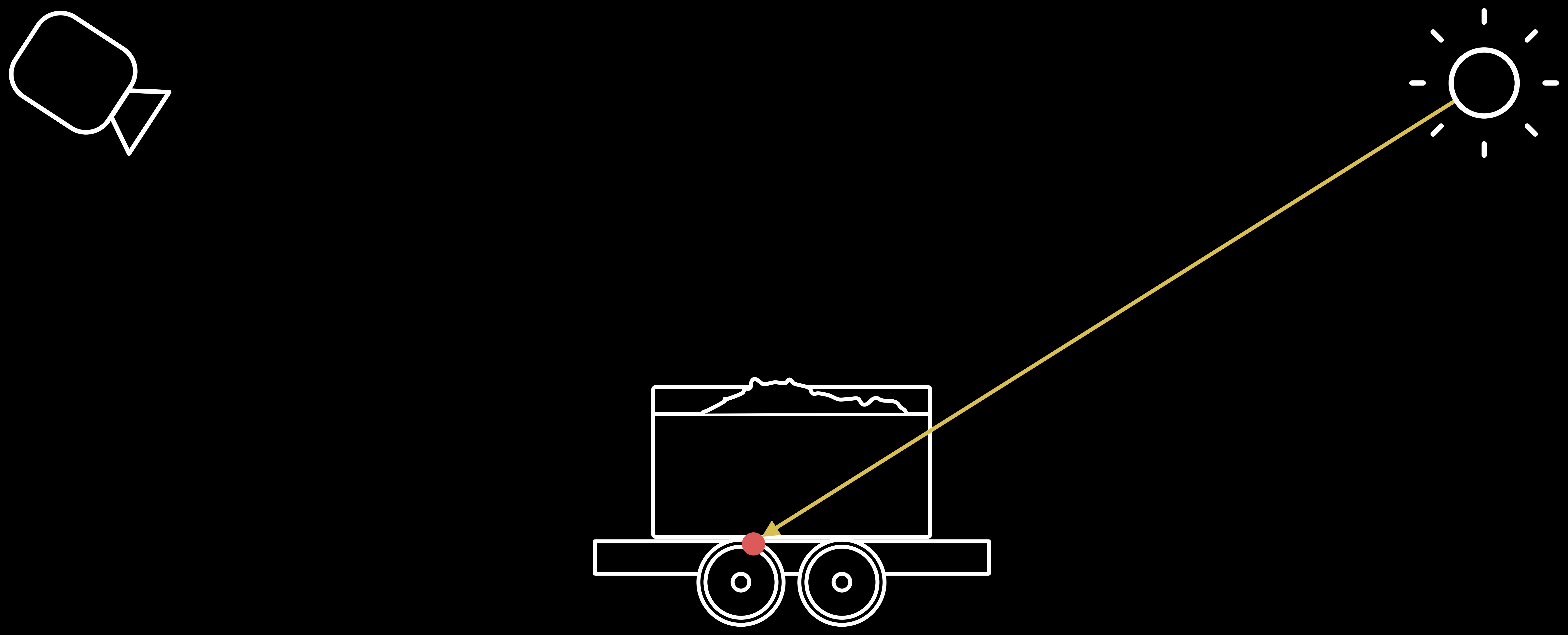
Physically Based Rendering



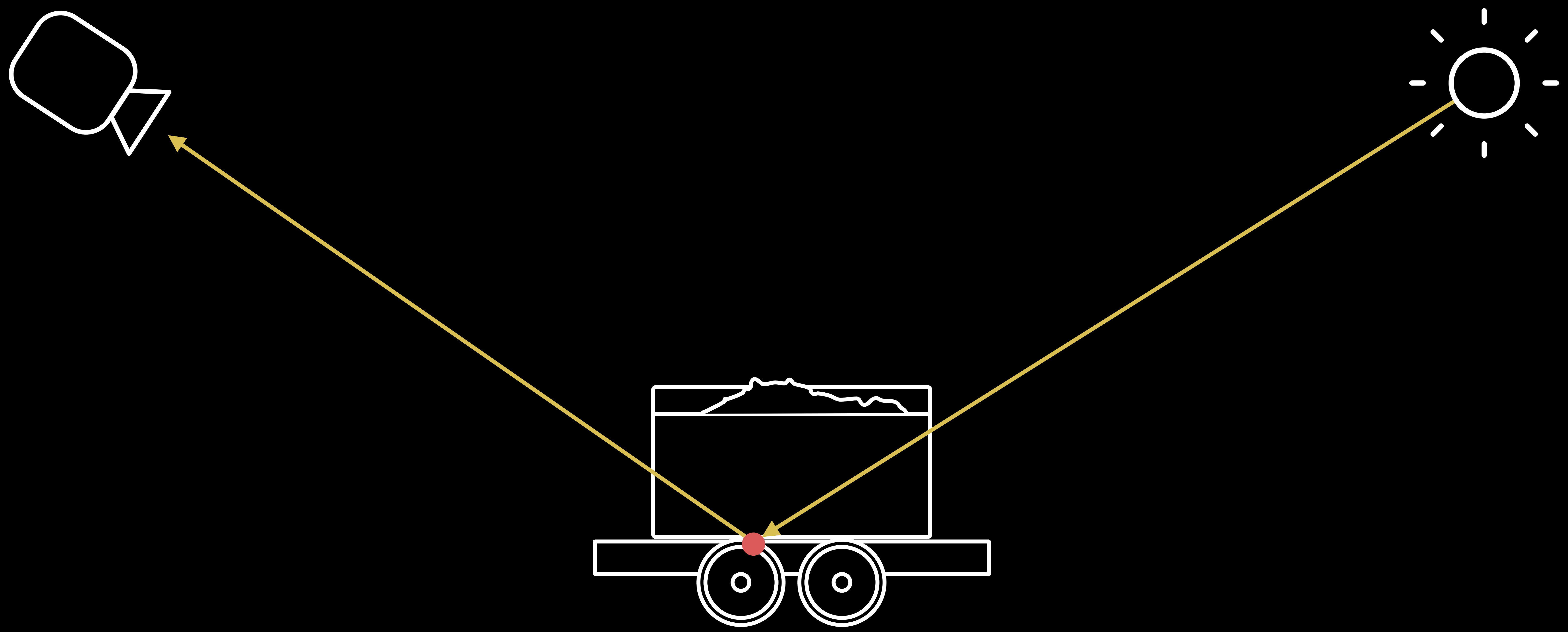
Physically Based Rendering



Physically Based Rendering



Physically Based Rendering



Physically Based Rendering

Bidirectional reflectance distribution function

$$L_o(\boldsymbol{v}) = \int_\Omega f(\boldsymbol{l},\boldsymbol{v})\, L_i(\boldsymbol{l})\, \langle \boldsymbol{n} \cdot \boldsymbol{l} \rangle\, d\boldsymbol{l}$$

$$f(\boldsymbol{l},\boldsymbol{v}) = f_d(\boldsymbol{l},\boldsymbol{v}) + f_r(\boldsymbol{l},\boldsymbol{v})$$

$$f_d(\boldsymbol{l},\boldsymbol{v}) = \frac{c_{diff}}{\pi} \qquad f_r(\boldsymbol{l},\boldsymbol{v}) = \frac{D(\boldsymbol{h})\, G(\boldsymbol{l},\boldsymbol{v})\, F(\boldsymbol{l},\boldsymbol{v})}{4\, \langle \boldsymbol{n} \cdot \boldsymbol{l} \rangle \langle \boldsymbol{n} \cdot \boldsymbol{v} \rangle}$$

Physically Based Rendering

Relies on intuitive physical material properties

Adopted and loved by artists

High-level and easy-to-use API

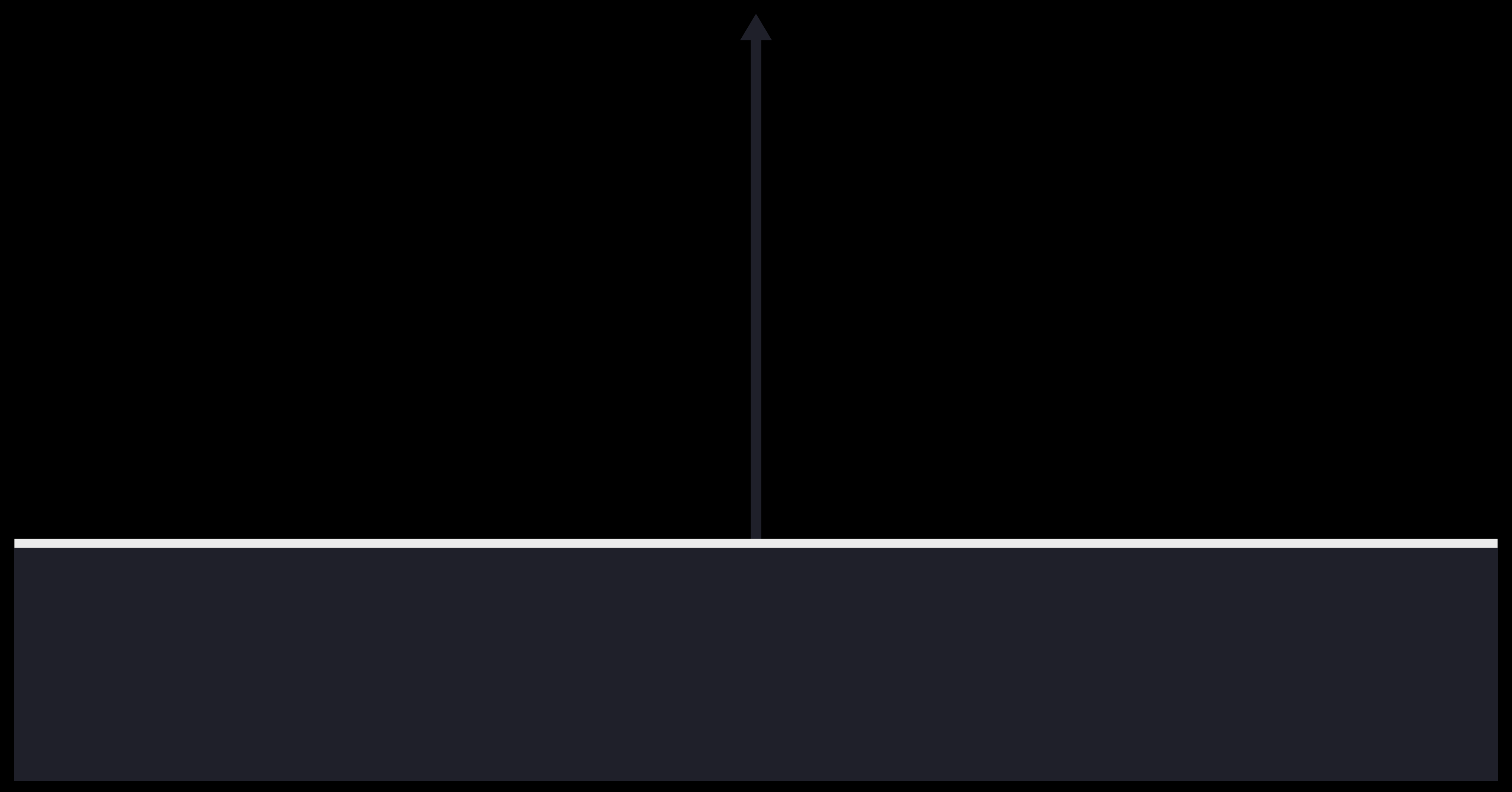
Physically Based Rendering

Physically based materials

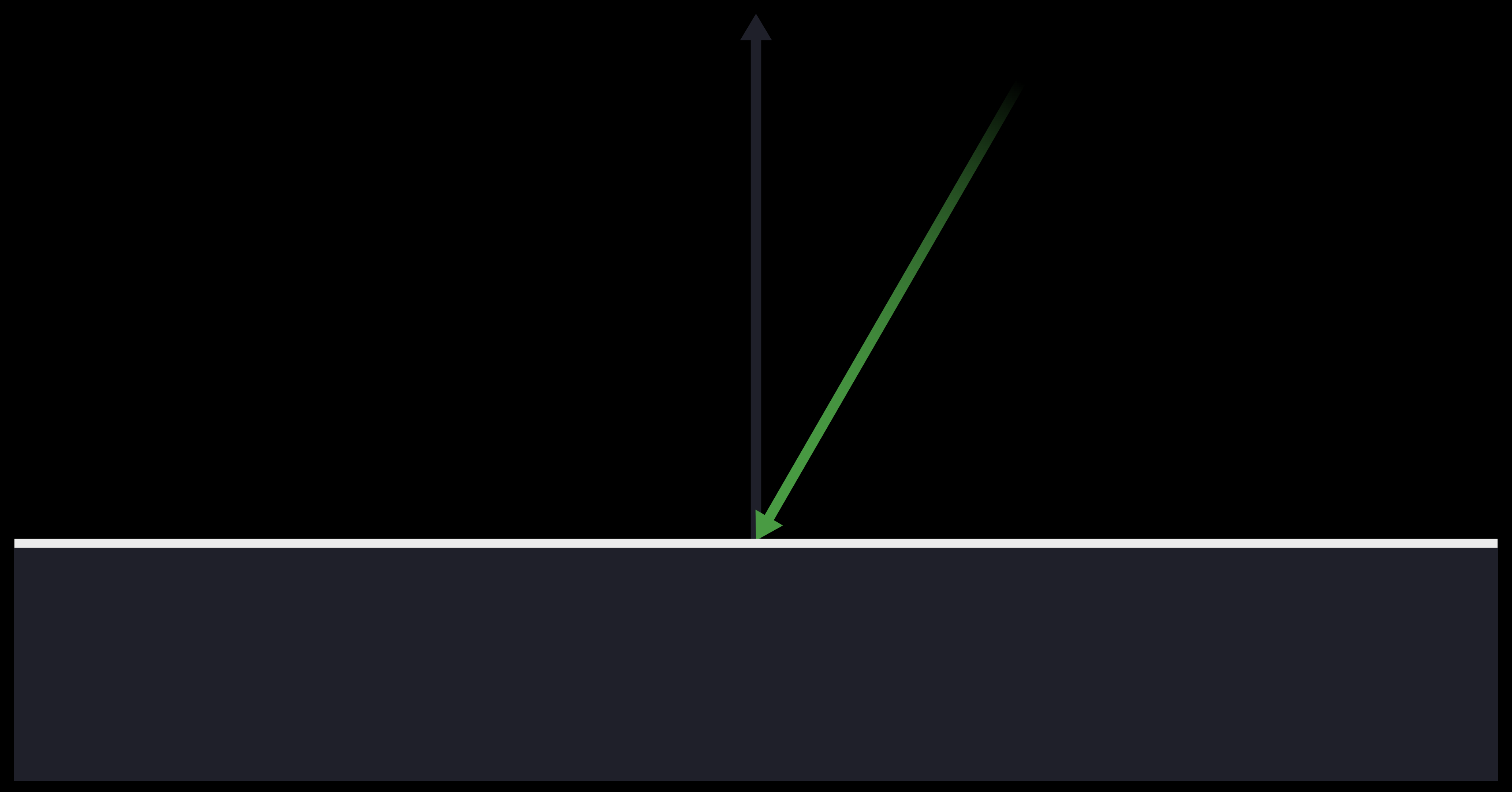
Physically based lights

Physically Based Materials

Physically Based Materials



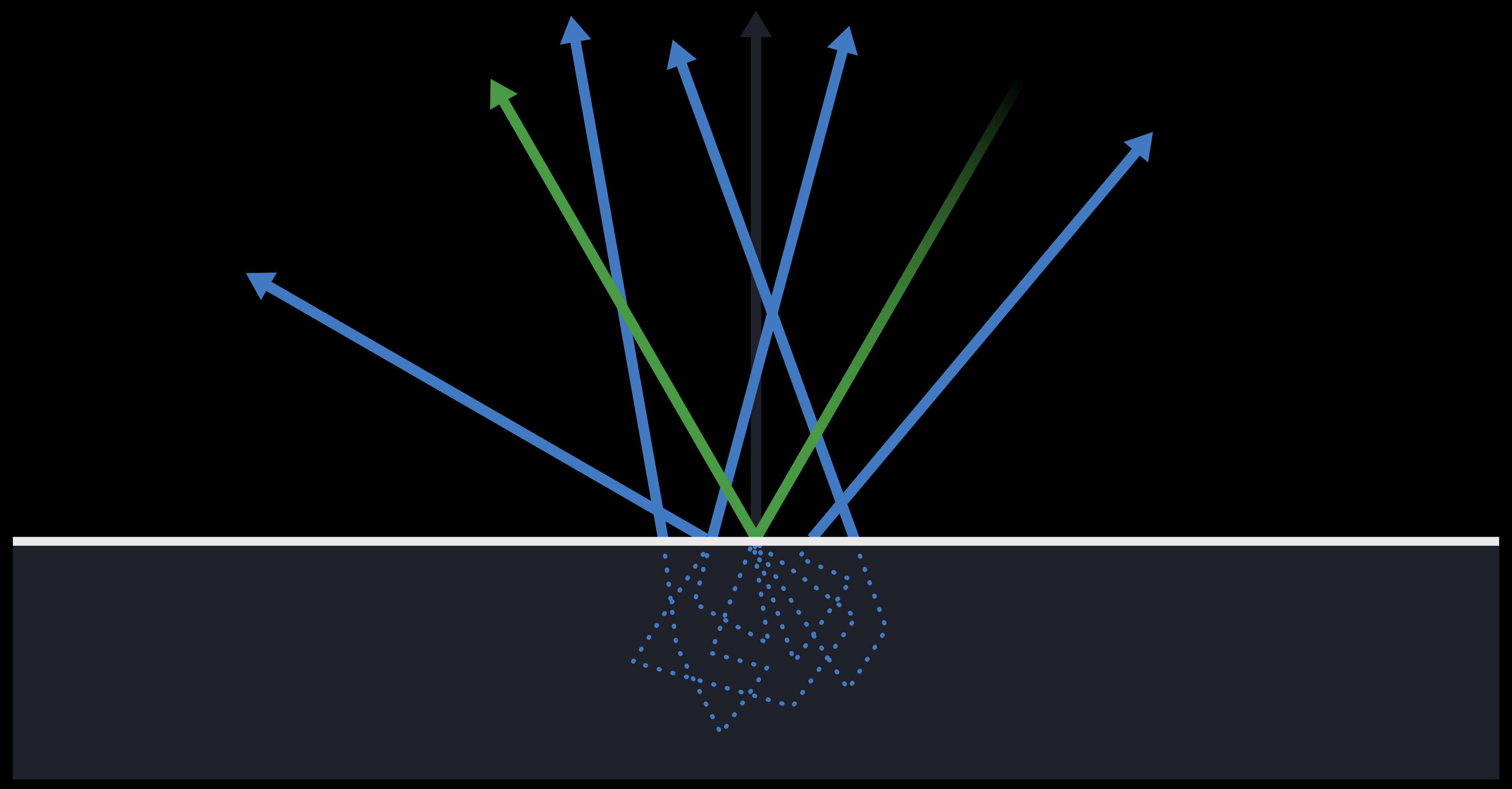
Physically Based Materials



Physically Based Materials

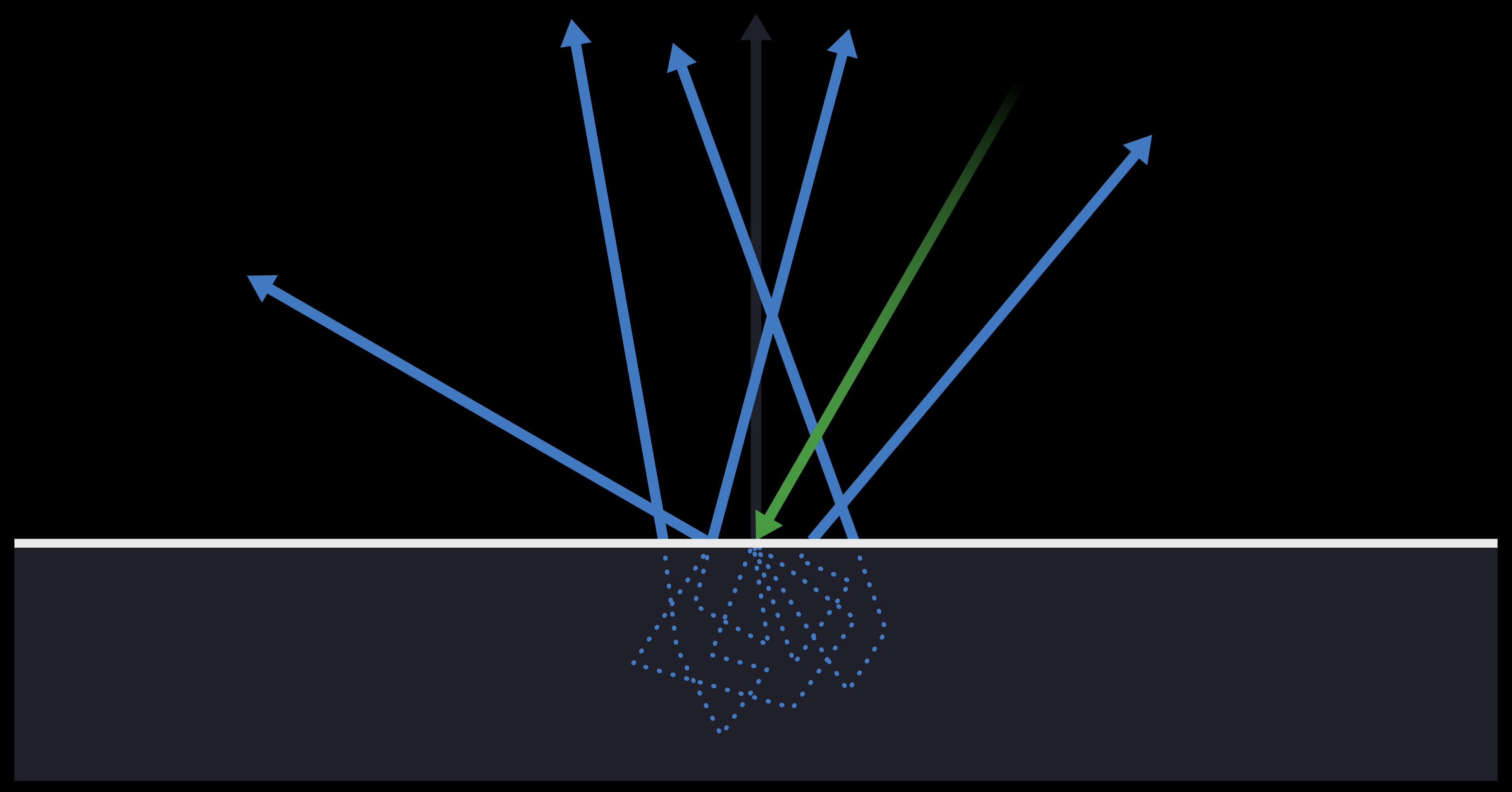
Diffuse Reflection

Specular Reflection



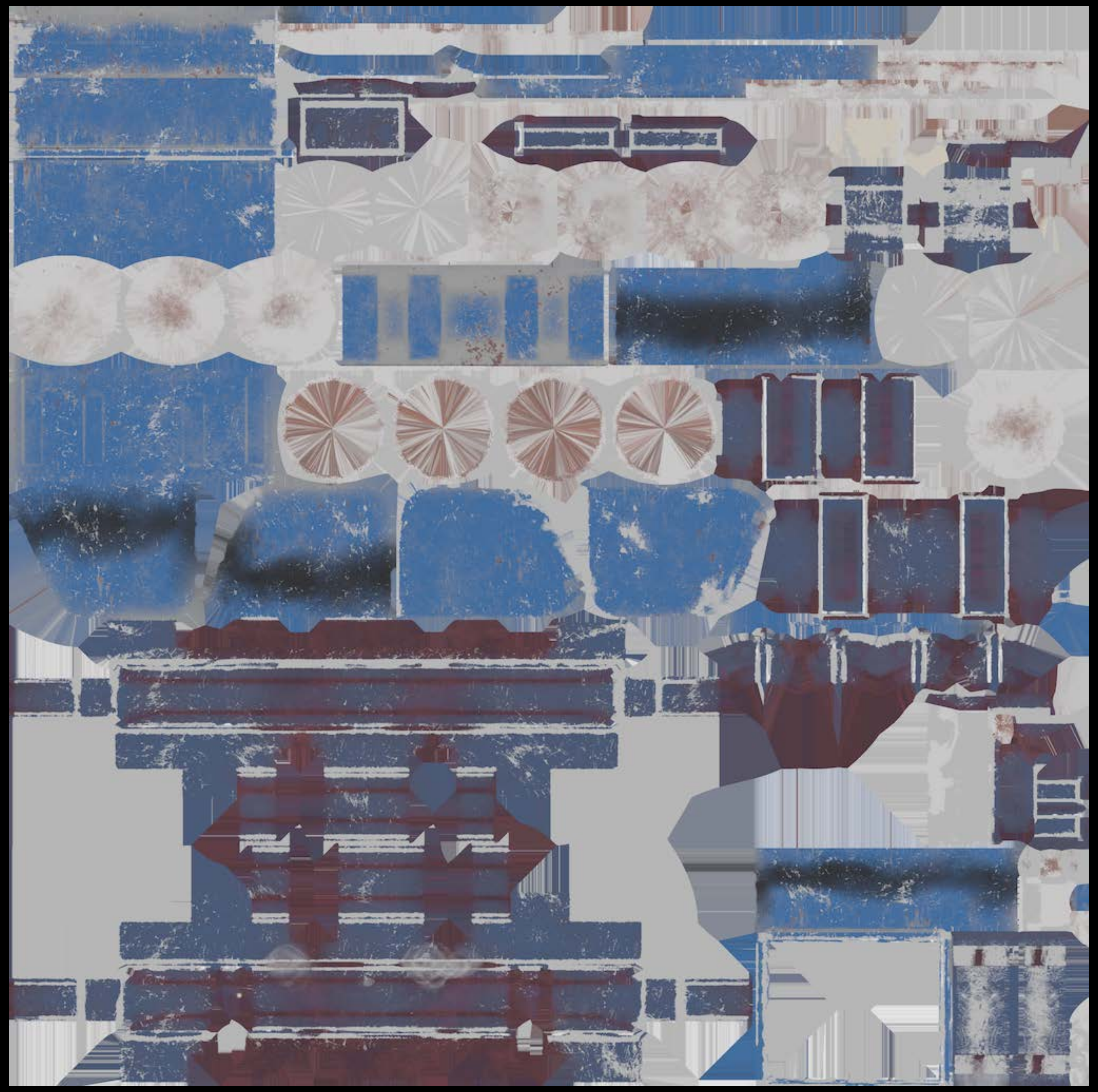
Physically Based Materials

Diffuse reflection



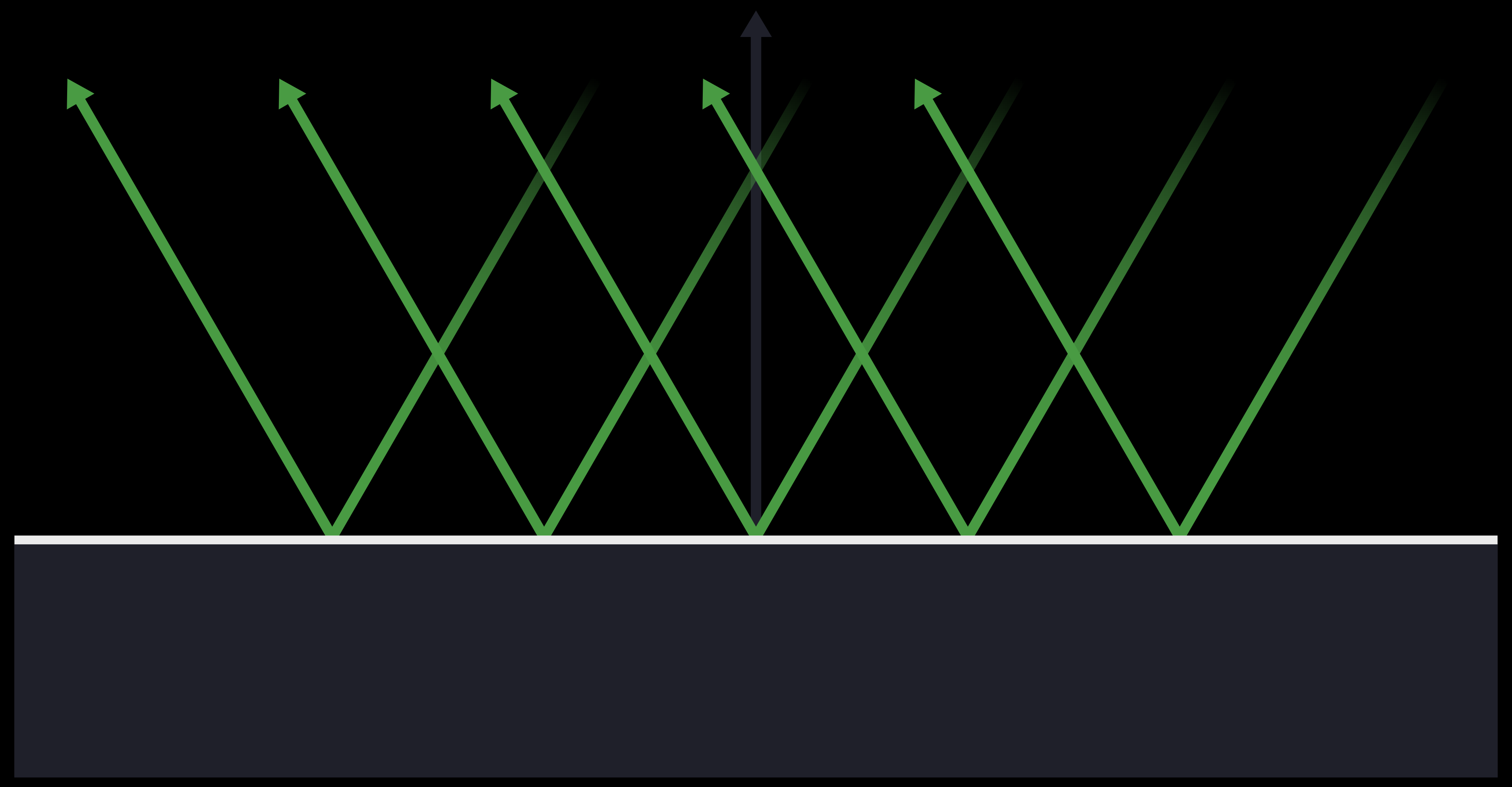
Physically Based Materials

Diffuse reflection



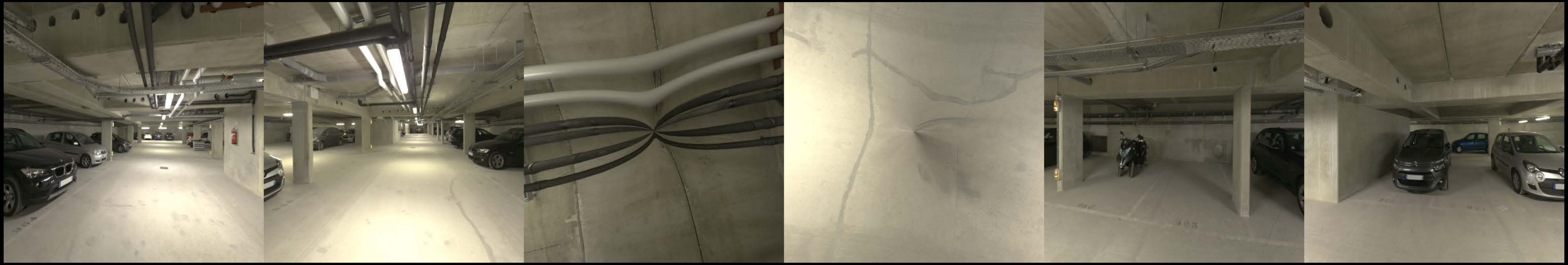
Physically Based Materials

Specular reflection



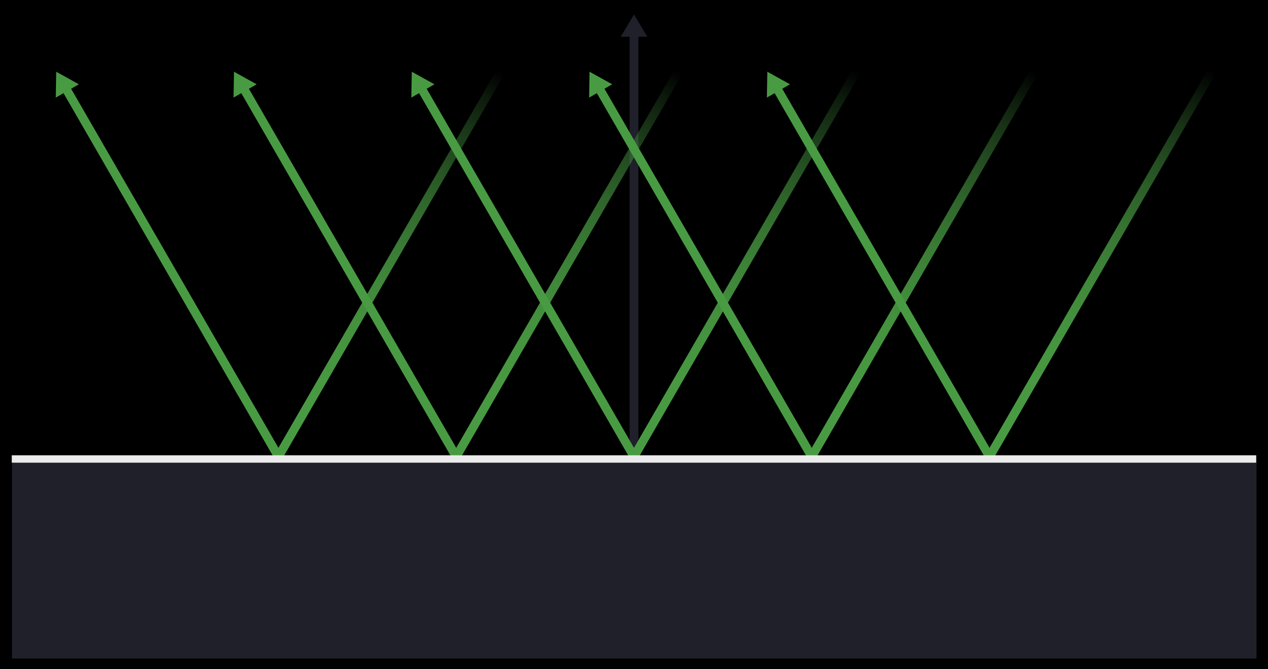
Physically Based Materials

Specular reflection



Physically Based Materials

Specular reflection

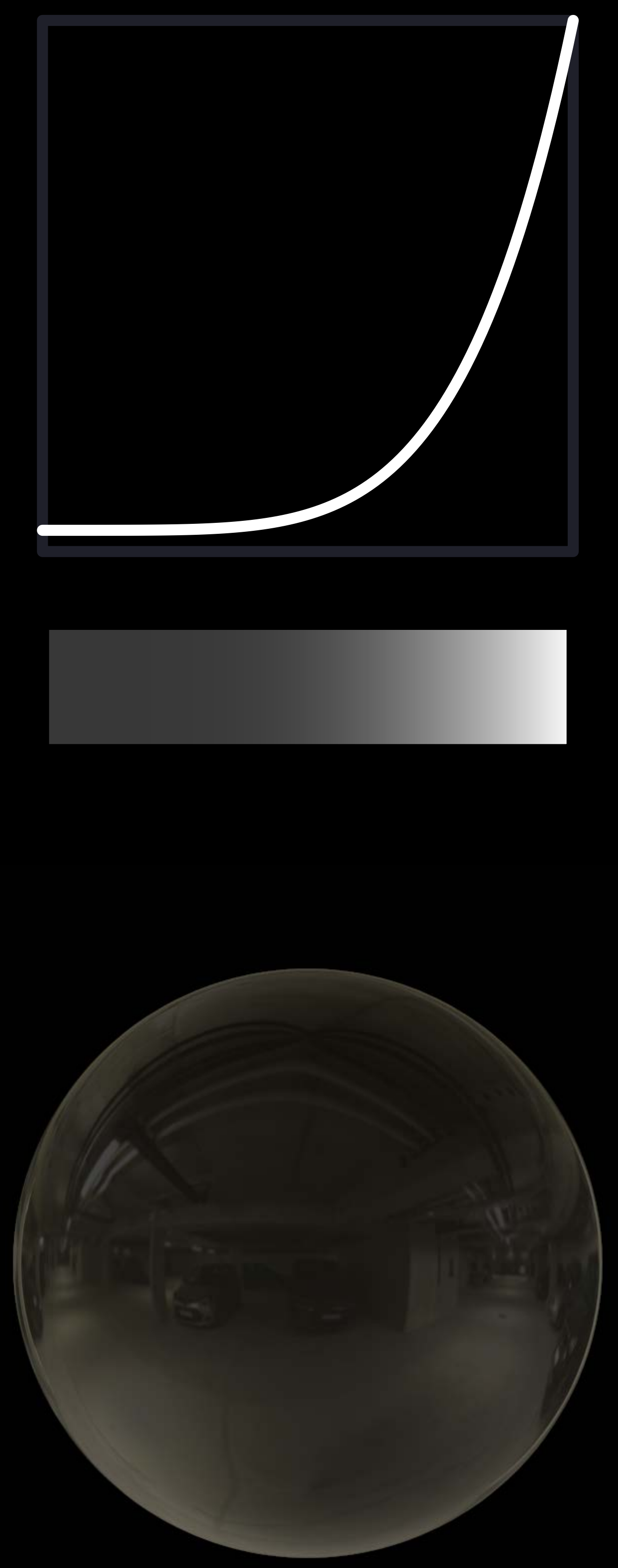


Physically Based Materials

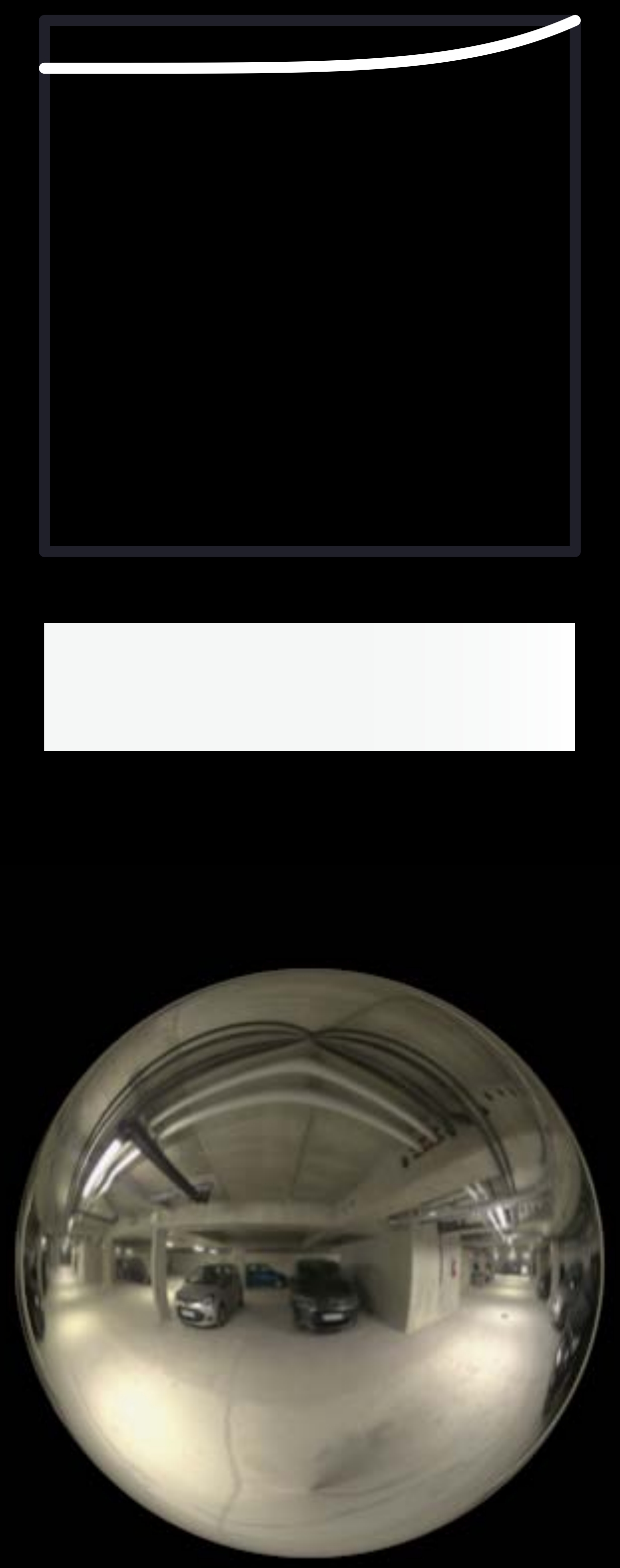
Reflectance

Physically Based Materials

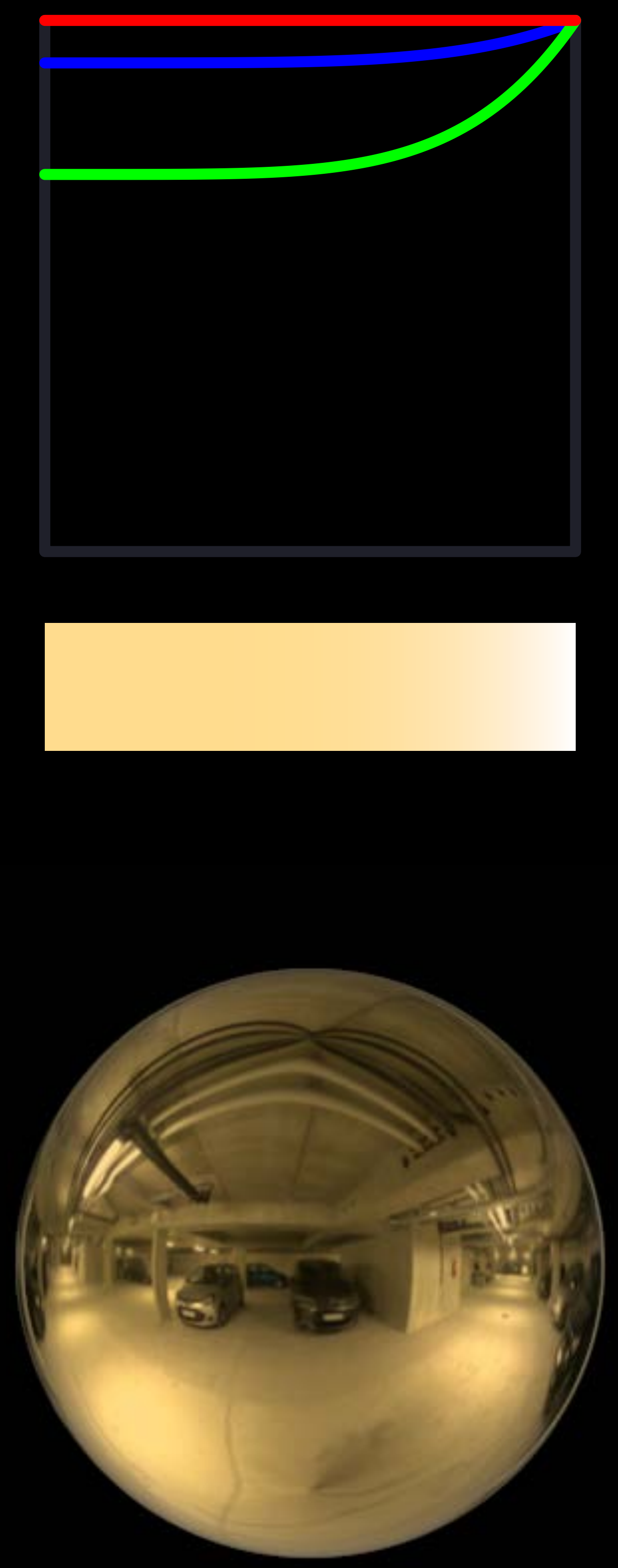
Reflectance



Plastic



Aluminum

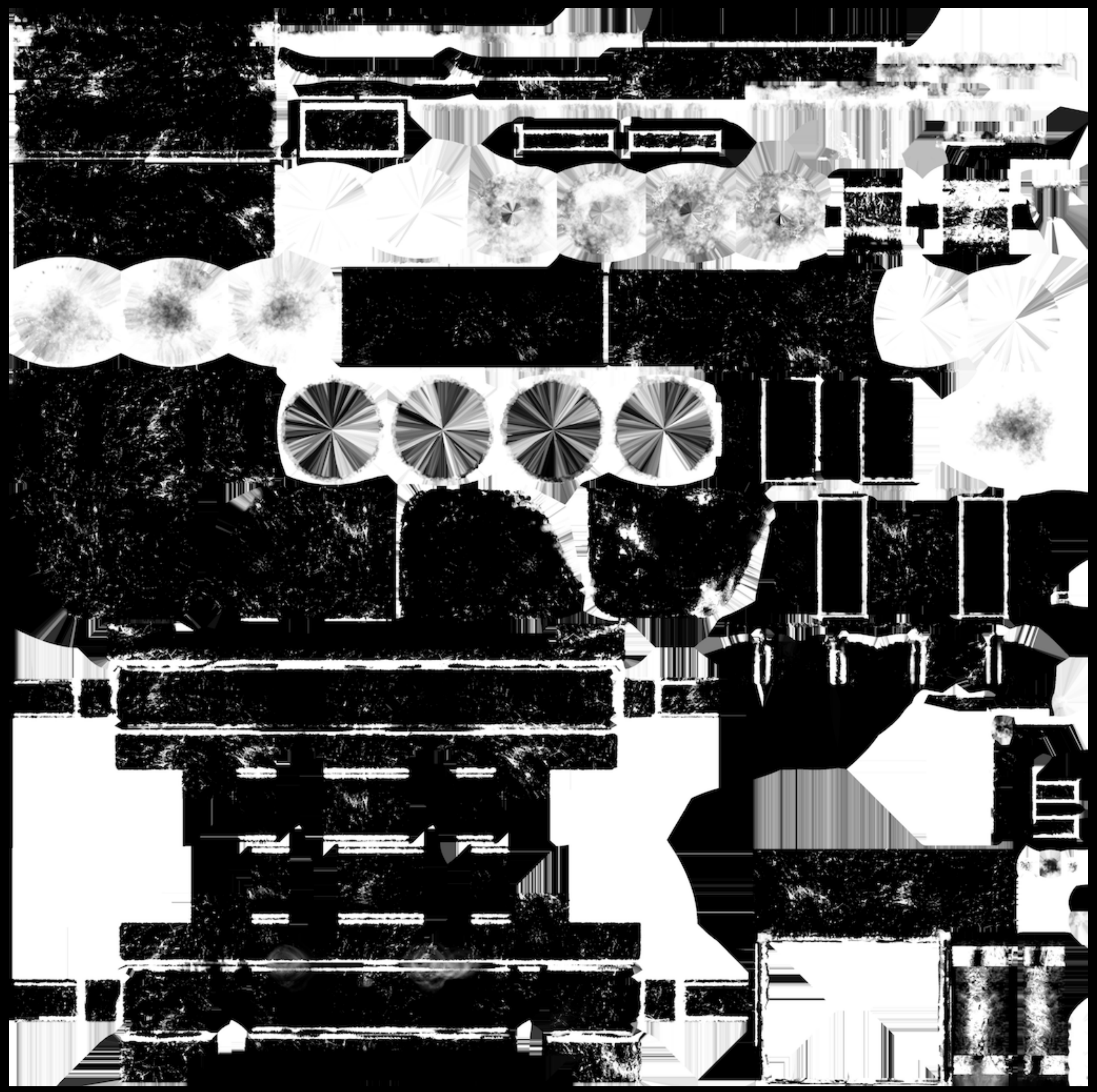


Gold

Physically Based Materials

Metalness

NEW



```
public class SCNMaterial {
   public var metalness: SCNMaterialProperty { get }
}
```

Physically Based Materials

Metal versus dielectric

Metal	Dielectric
High reflectance	Low reflectance
Absorb light	Absorb and scatter light

Physically Based Materials

Metal versus dielectric

Metal	Dielectric
Bright specular reflection	Specular reflection at grazing angles
No diffuse reflection	Mainly diffuse reflection

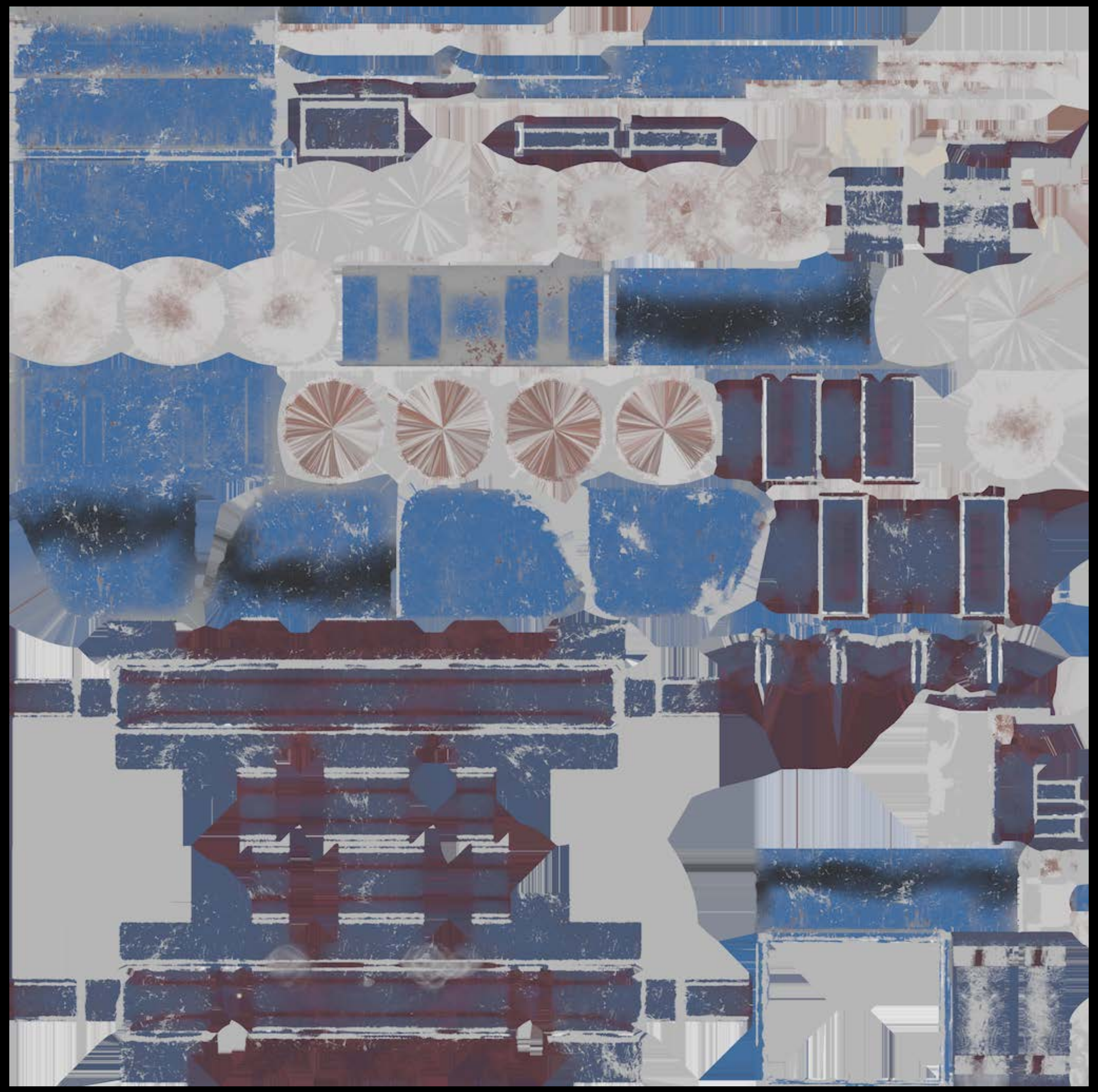
Physically Based Materials

Metal versus dielectric

Metal	Dielectric
Reflectance at 0° `diffuse` map	Reflectance at 0° constant
—	Object albedo `diffuse` map

Physically Based Materials

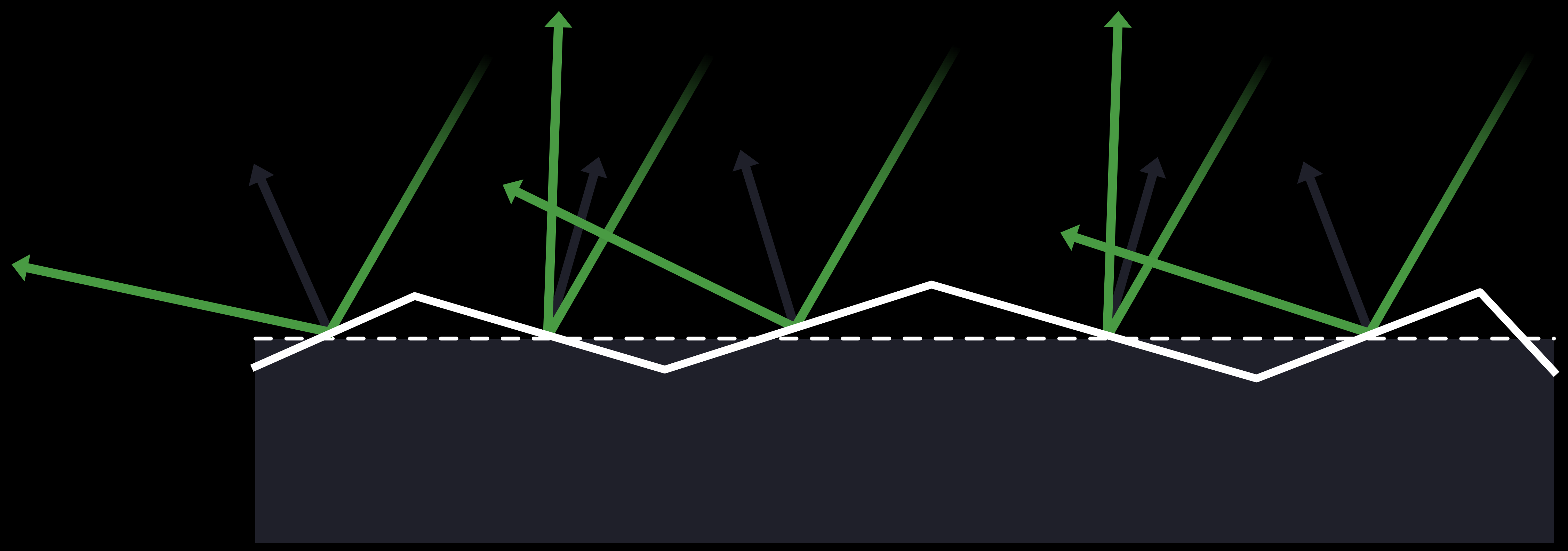
Metal versus dielectric



```
public class SCNMaterial {
   public var diffuse: SCNMaterialProperty { get }
}
```

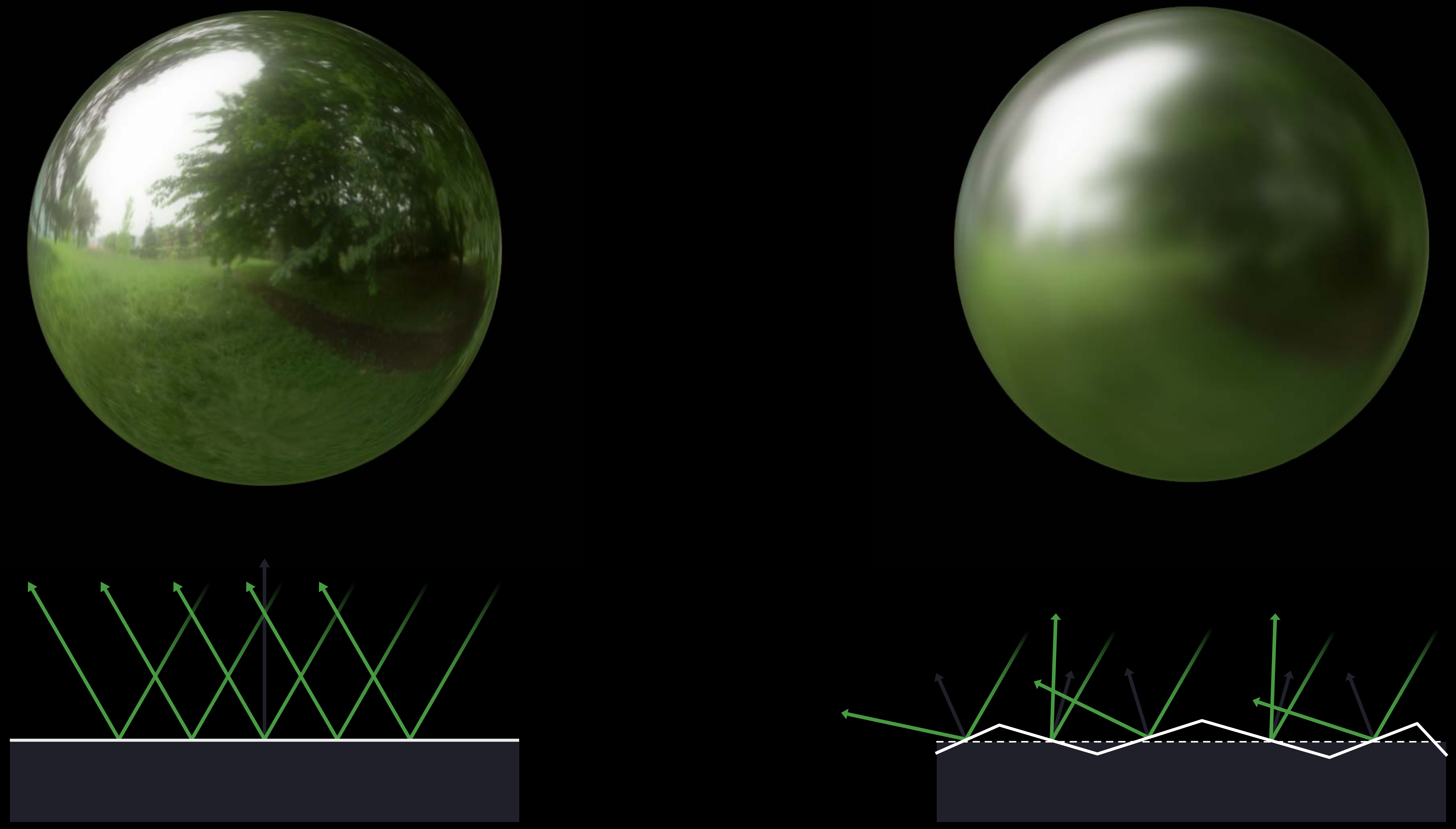
Physically Based Materials

Roughness



Physically Based Materials

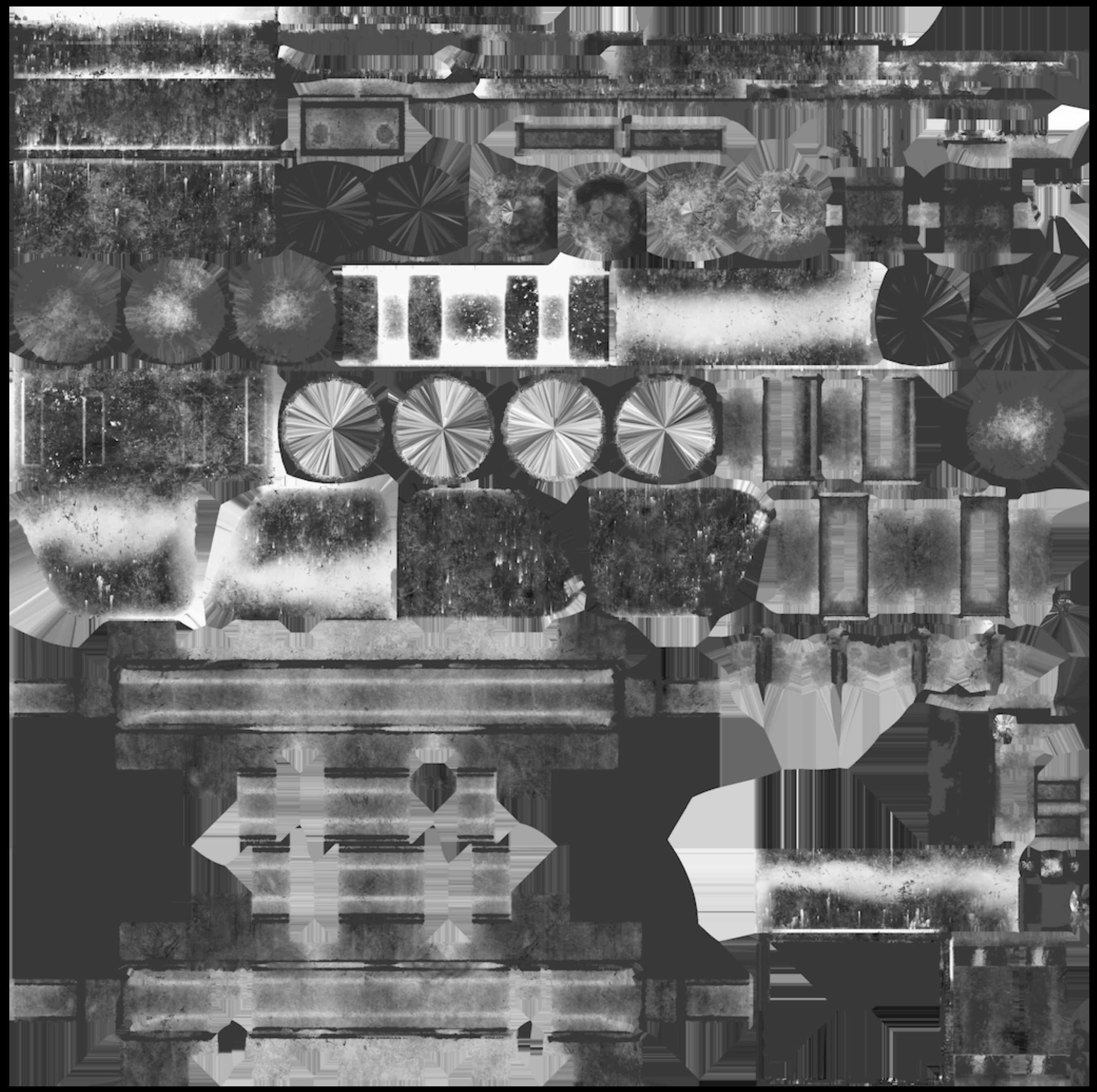
Roughness



Physically Based Materials

Roughness

NEW



```
public class SCNMaterial {
   public var roughness: SCNMaterialProperty { get }
}
```

Physically Based Materials

Material API

Three fundamental properties

- Albedo or reflectance at 0°
- Metalness
- Roughness

```
public class SCNMaterial {
   public var diffuse: SCNMaterialProperty { get }
   public var metalness: SCNMaterialProperty { get }
   public var roughness: SCNMaterialProperty { get }
}
```

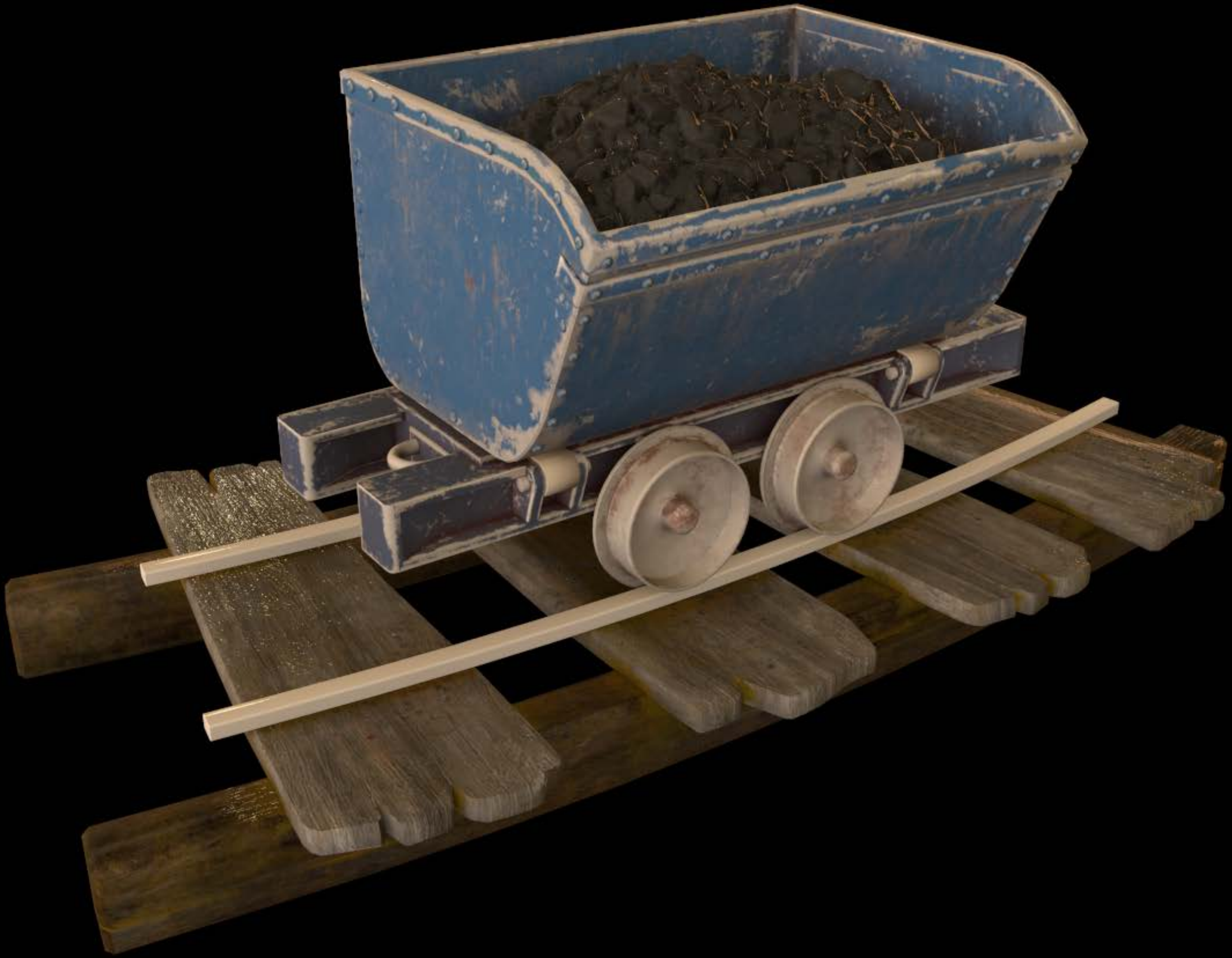
Physically Based Materials

Material API

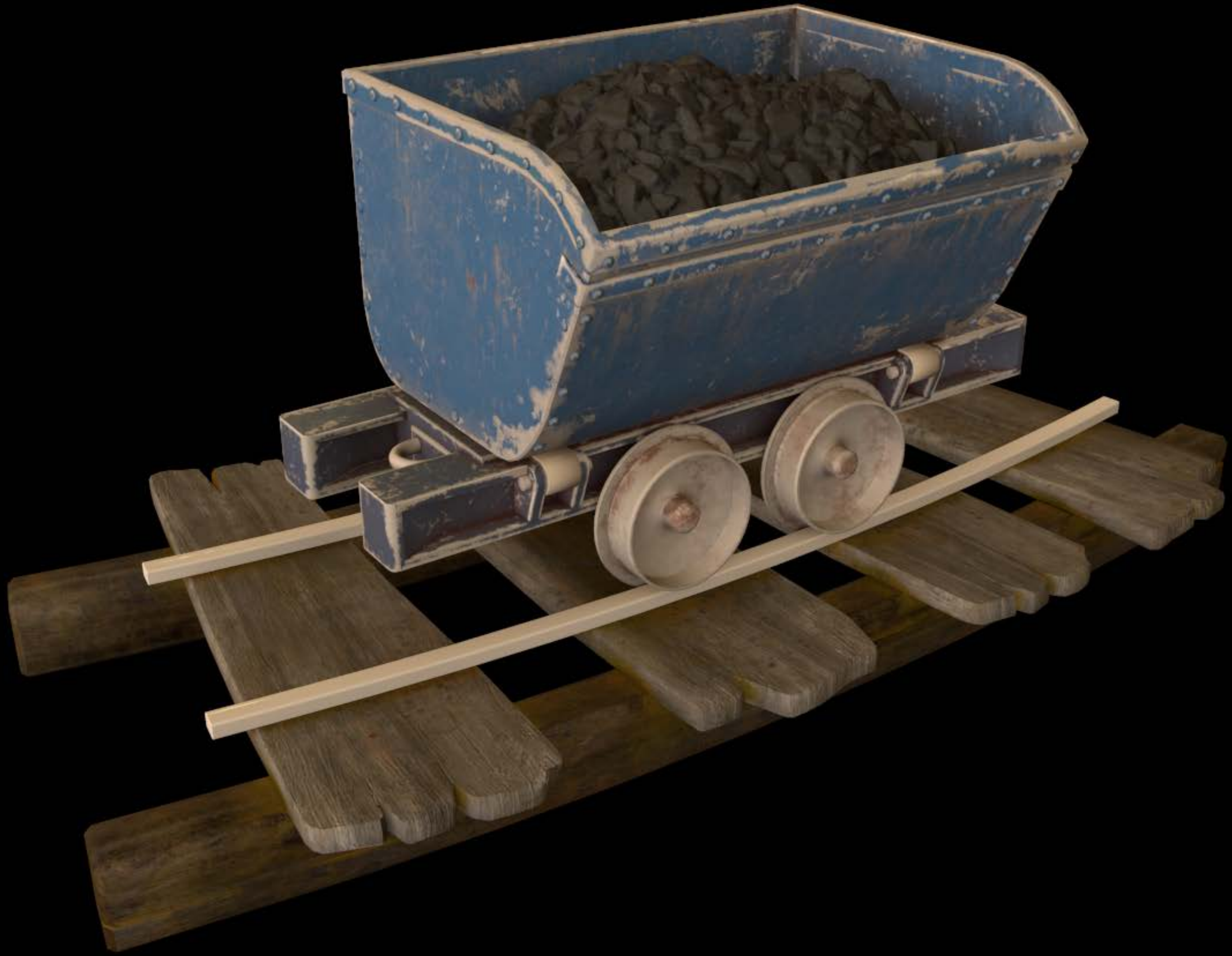
New physically based lighting model

`diffuse`, `metalness`, and `roughness` maps

```
let material = SCNMaterial()
material.lightingModelName = .physicallyBased
material.diffuse.contents = "albedo.png"
material.metalness.contents = "metalness.png"
material.roughness.contents = "roughness.png"
```

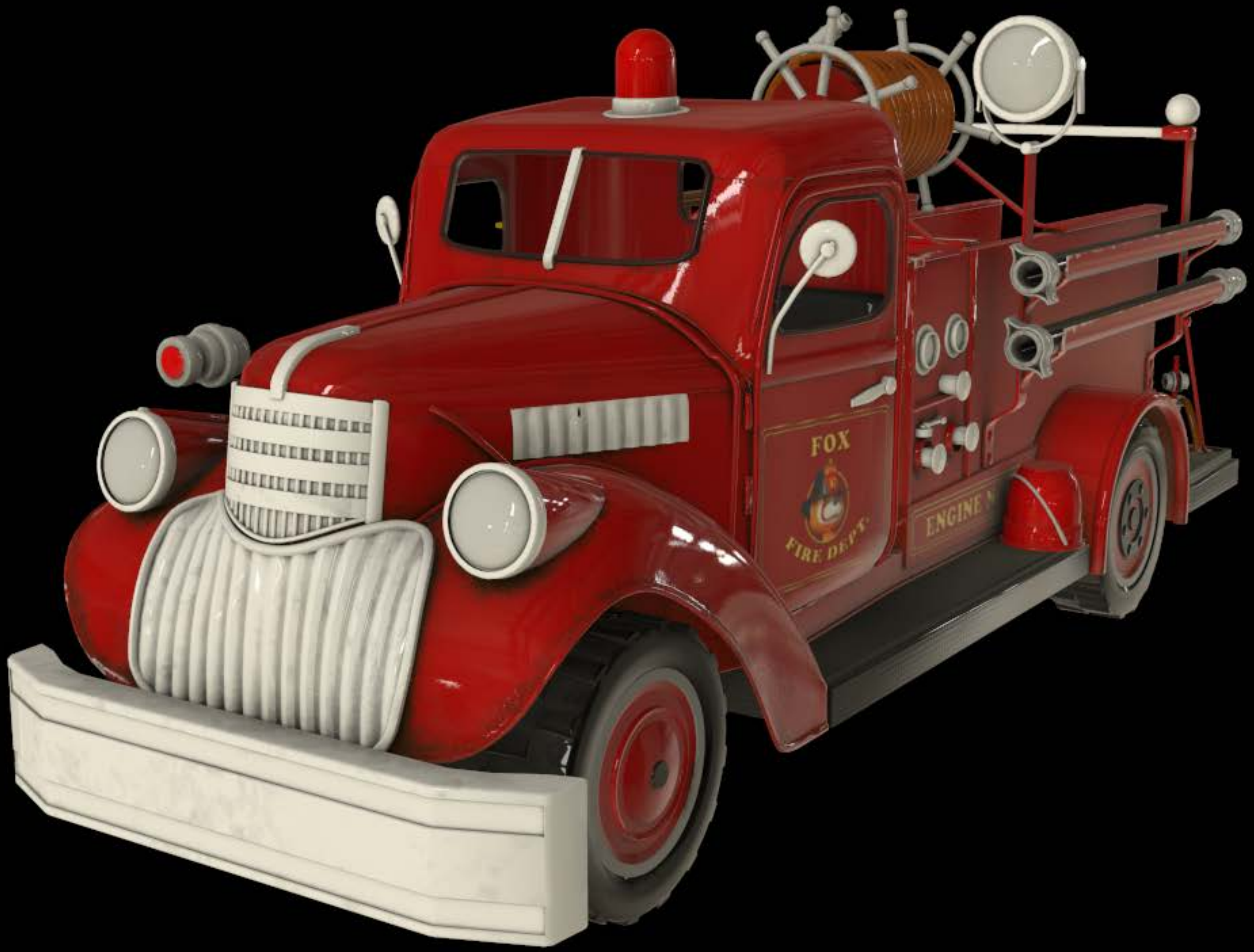


diffuse map

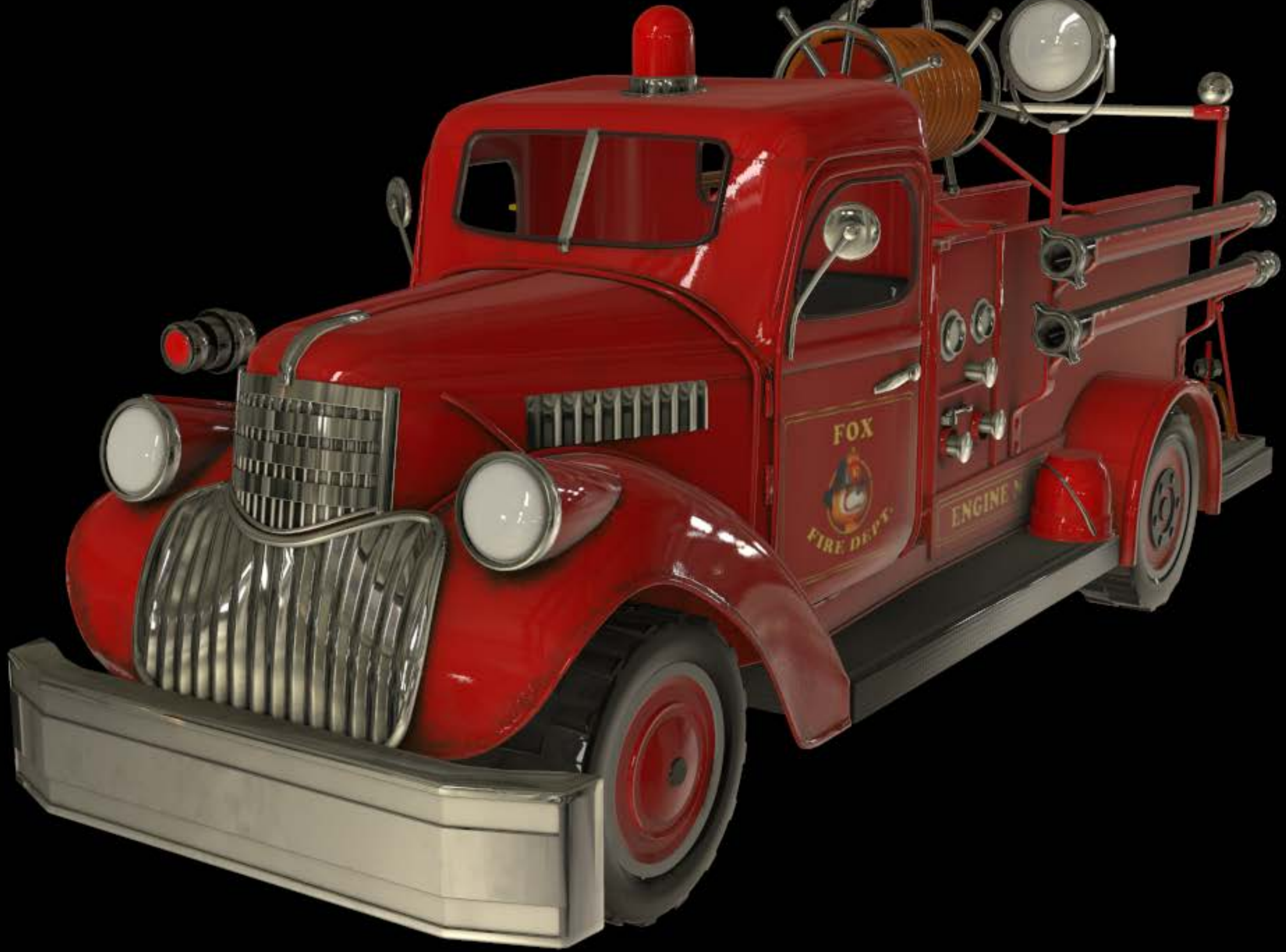


`diffuse` map, `roughness` map

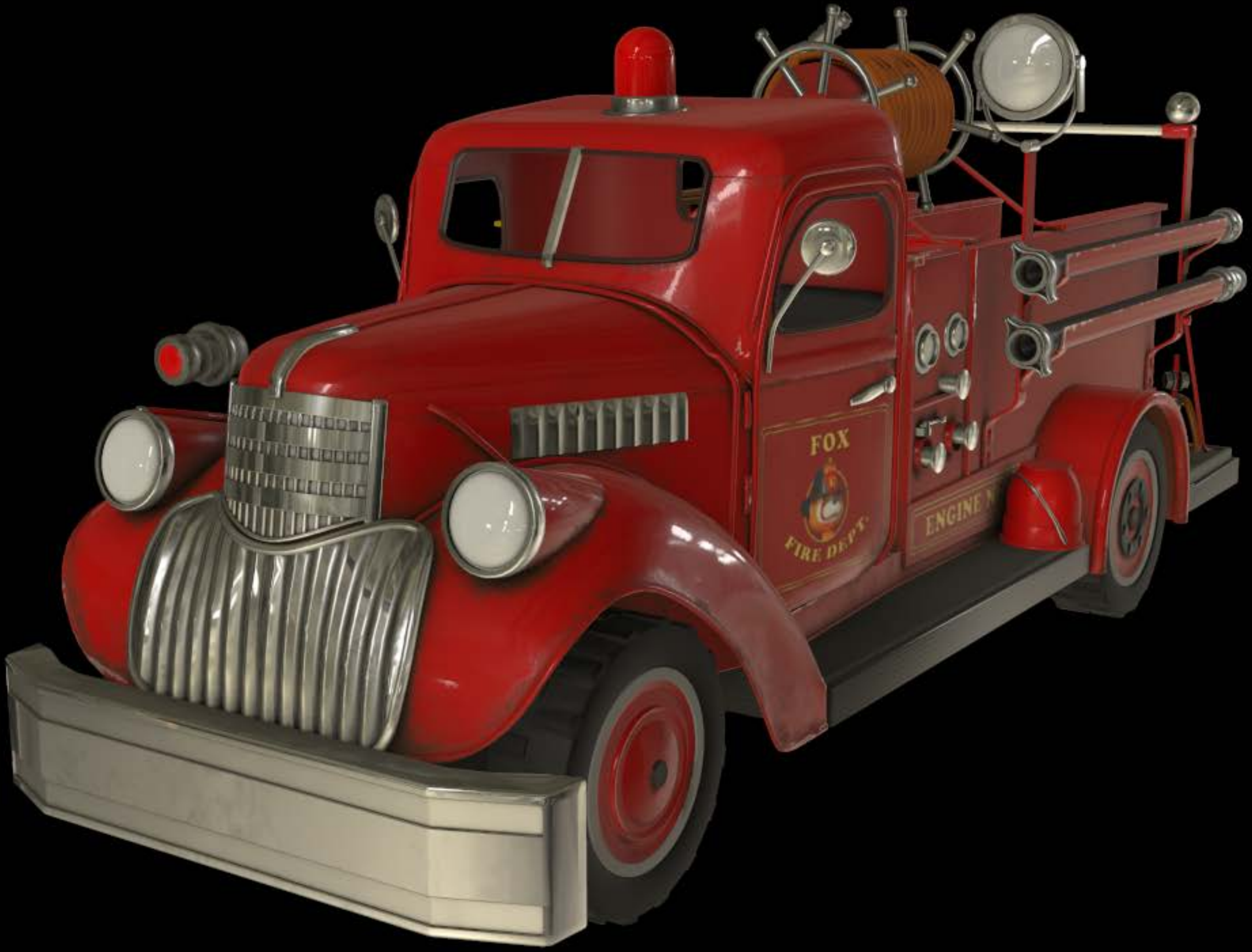
`diffuse` map, `roughness` map, `metalness` map



`diffuse` map



`diffuse` map, `metalness` map



`diffuse` map, `metalness` map, `roughness` map

Physically Based Materials

Material API

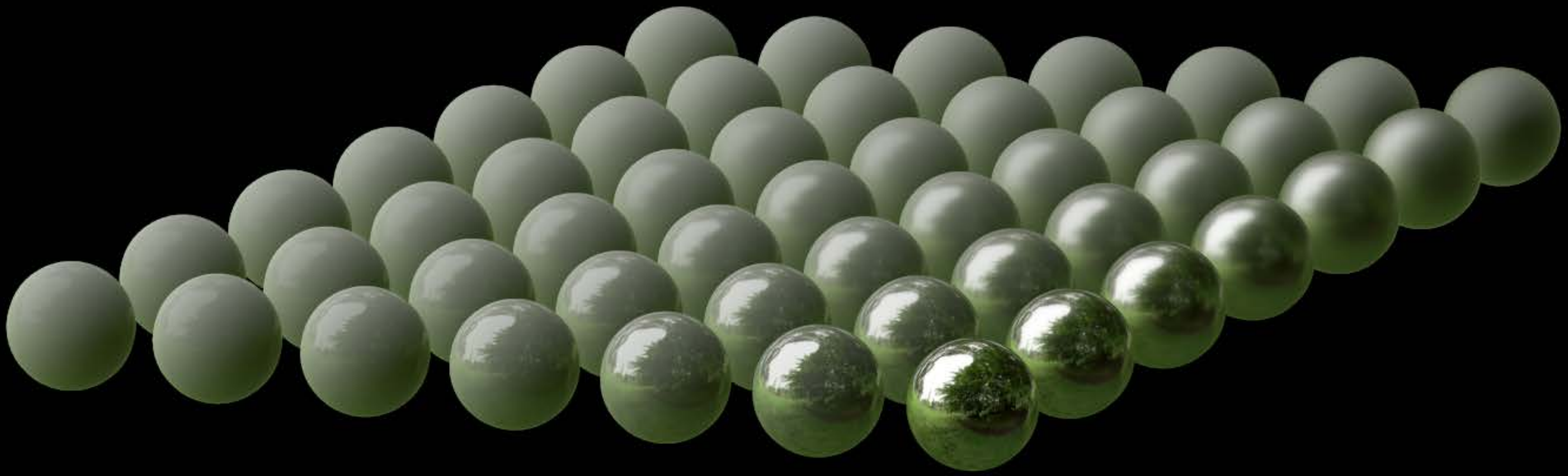
NEW

Use grayscale images for `metalness`, `roughness`, and `ambientOcclusion`

Use scalars for constant values

```
material.metalness.contents = "metalness.png"
material.roughness.contents = NSNumber(value: 0.5)
```

Physically Based Materials



Physically Based Rendering

Physically based materials

Physically based lights

Physically Based Lights

Physically Based Lights

Image based lighting

Light probes

Point lights

Physically Based Lights

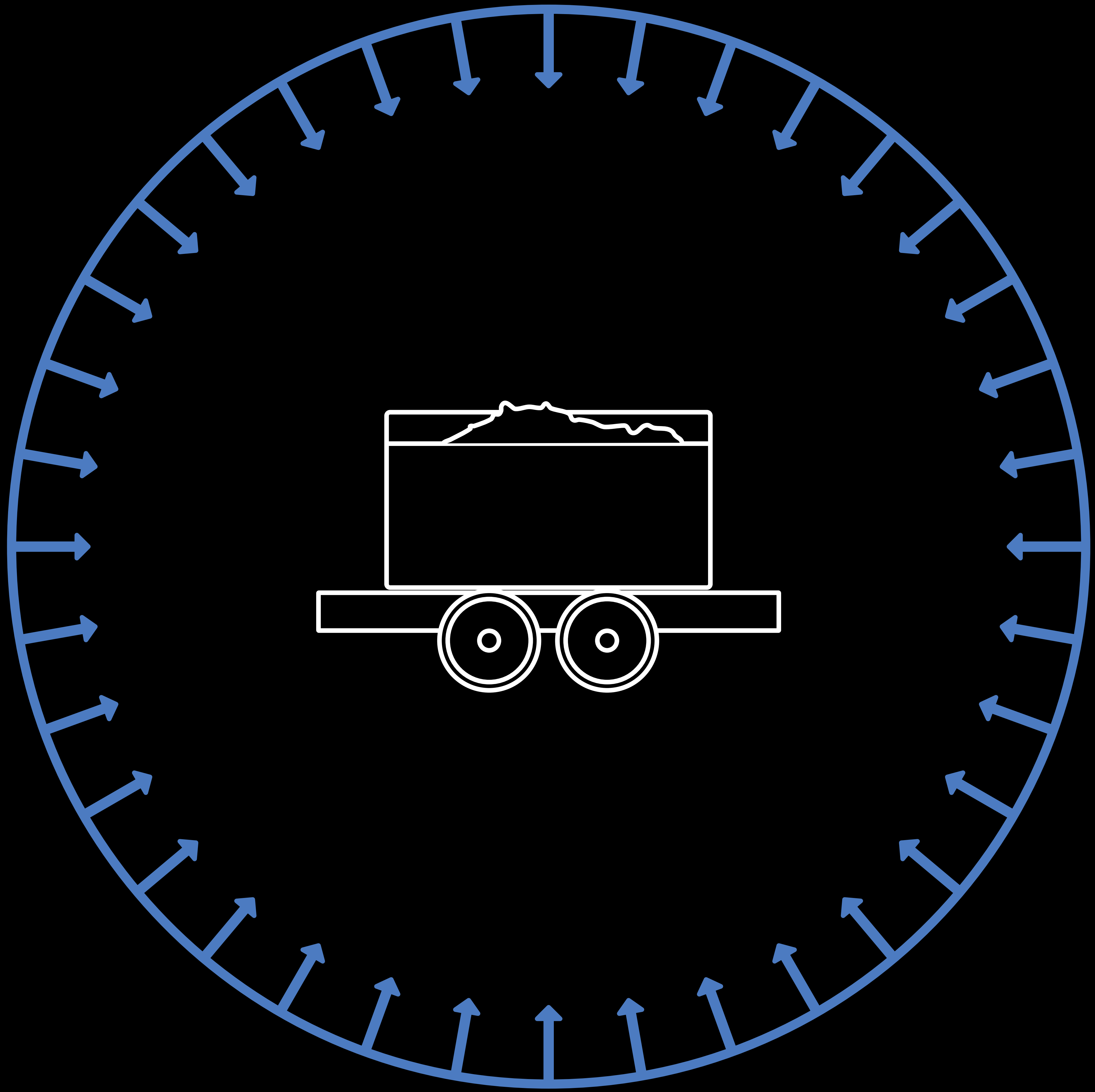
Image based lighting

Light probes

Point lights

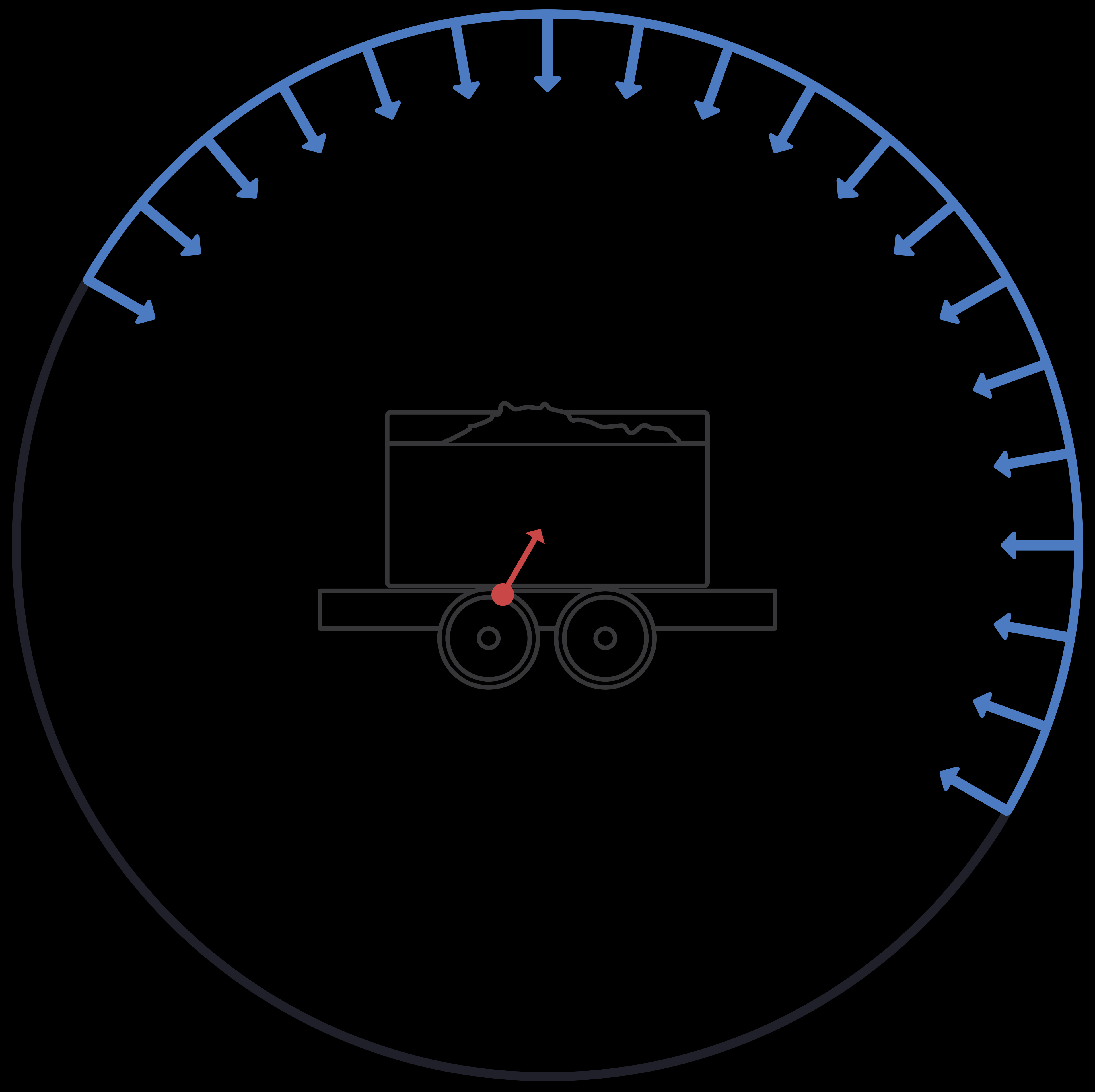
Physically Based Lights

Image based lighting



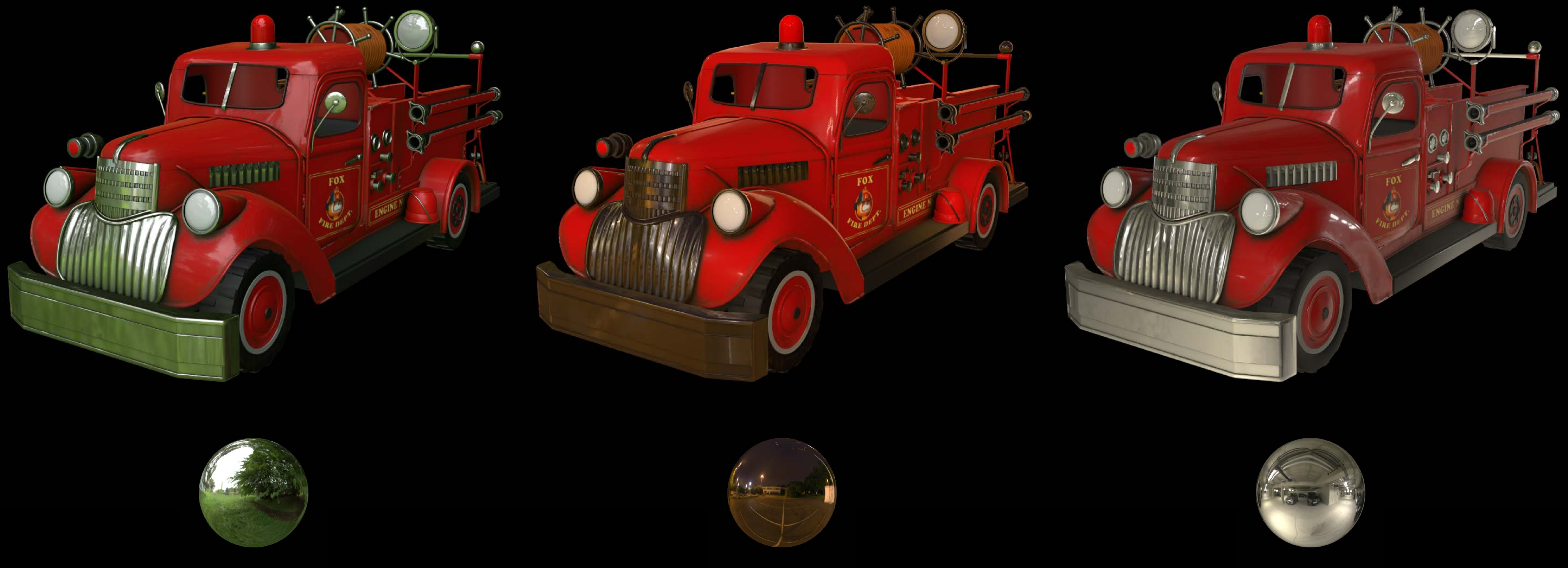
Physically Based Lights

Image based lighting



Physically Based Lights

Image based lighting



Physically Based Lights

Image based lighting

Cube map captures the environment

Lighting information is derived from cube map

Image based lighting can be used alone

Not mandatory to add lights in the scene

Physically Based Lights

Image based lighting

NEW

A single change affects the whole scene

```
let scene = SCNScene()
scene.lightingEnvironment.contents = "outside.exr"
```

Physically Based Lights

NEW

Image based lighting

A single change affects the whole scene

Works great with the `background` property

```
let scene = SCNScene()
scene.lightingEnvironment.contents = "outside.exr"
scene.background.contents = scene.lightingEnvironment.contents
```

FOX
FIRE DEPT.
ENGINE N

FOX
FIRE DEPT.
ENGINE N

Physically Based Lights

Image based lighting: Caveats

Captures the distant environment

Does not account for obstacles in the scene

Not suited for occluded objects

Physically Based Lights

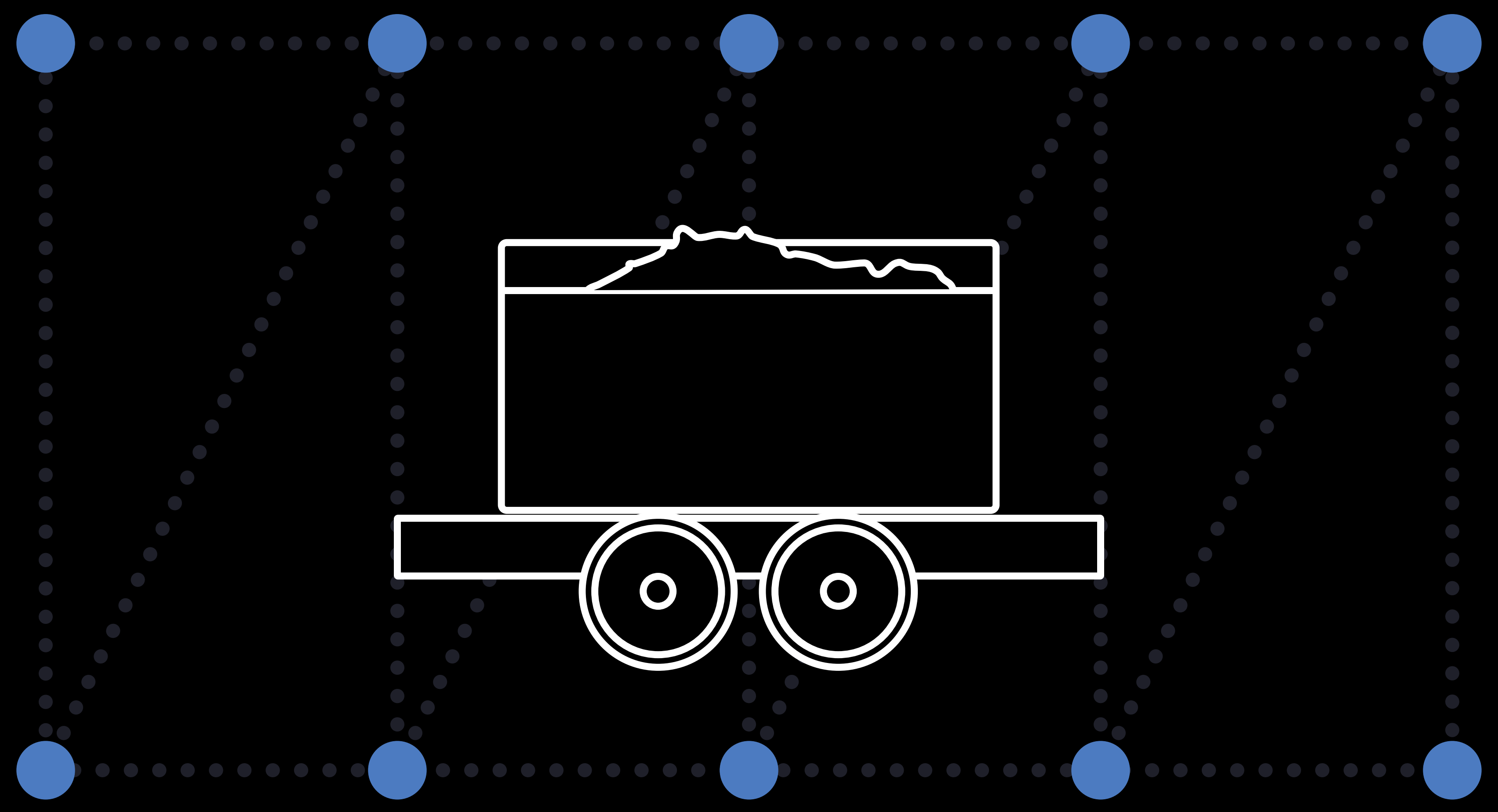
Image based lighting

Light probes

Point lights

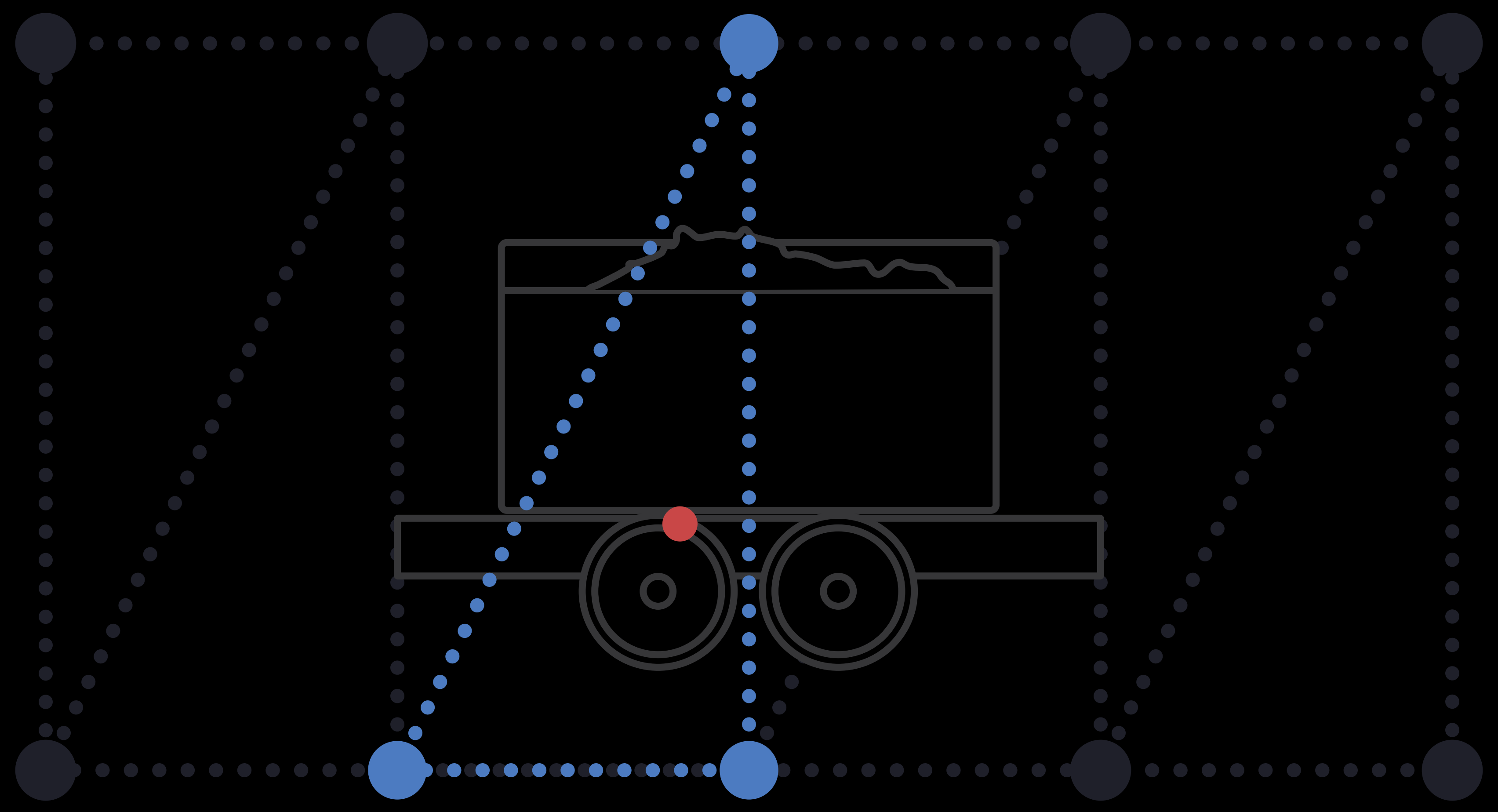
Physically Based Lights

Light probes



Physically Based Lights

Light probes



Physically Based Lights

Light probes

A special kind of light

Captures the local diffuse lighting

Account for obstacles in the scene

Lightweight

Efficient

Physically Based Lights

Light probes

NEW

A special kind of light: `SCNLightType.probe`

```
let light = SCNLight()
light.type = .probe
```

Physically Based Lights

NEW

Light probes

Can be placed programmatically or in Xcode

Static lighting information must be baked

```
public class SCNRenderer {
    public func updateProbes(_ probes: [SCNNode], atTime time: CFTimeInterval)
}
```

Physically Based Lights

Image based lighting

Light probes

Point lights

Physically Based Lights

Point lights

Work with physically based materials, too

Updated to be configured with real-world properties

```
public let SCNLightTypeOmni: String        // Omnidirectional light
public let SCNLightTypeDirectional: String // Directional light
public let SCNLightTypeSpot: String        // Spot light
```

Physically Based Lights

NEW

Point lights: Intensity

Expressed in lumens (lm)

```
let light = SCNLight()
light.intensity = 1500 // defaults to 1000 lm
```

Physically Based Lights

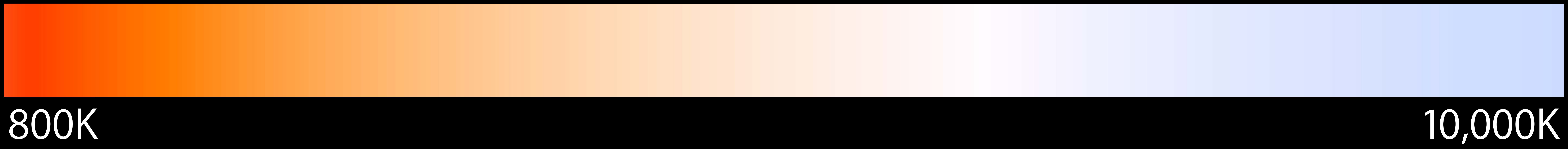
Point lights: Temperature

NEW

Expressed in Kelvin (K)

Modulates the light’s color

```swift
let light = SCNLight()
light.temperature = 5000
```



Physically Based Lights

NEW

Photometric lights

New kind of point light

```
public let SCNLightTypeOmni: String        // Omnidirectional light
public let SCNLightTypeDirectional: String // Directional light
public let SCNLightTypeSpot: String        // Spot light
public let SCNLightTypeIES: String         // IES light
```

Physically Based Lights

NEW

Photometric lights

New kind of point light

```
public let SCNLightTypeOmni: String        // Omnidirectional light
public let SCNLightTypeDirectional: String // Directional light
public let SCNLightTypeSpot: String        // Spot light
public let SCNLightTypeIES: String         // IES light
```

Physically Based Lights

Photometric lights

New kind of point light

Modeled after real-world lights

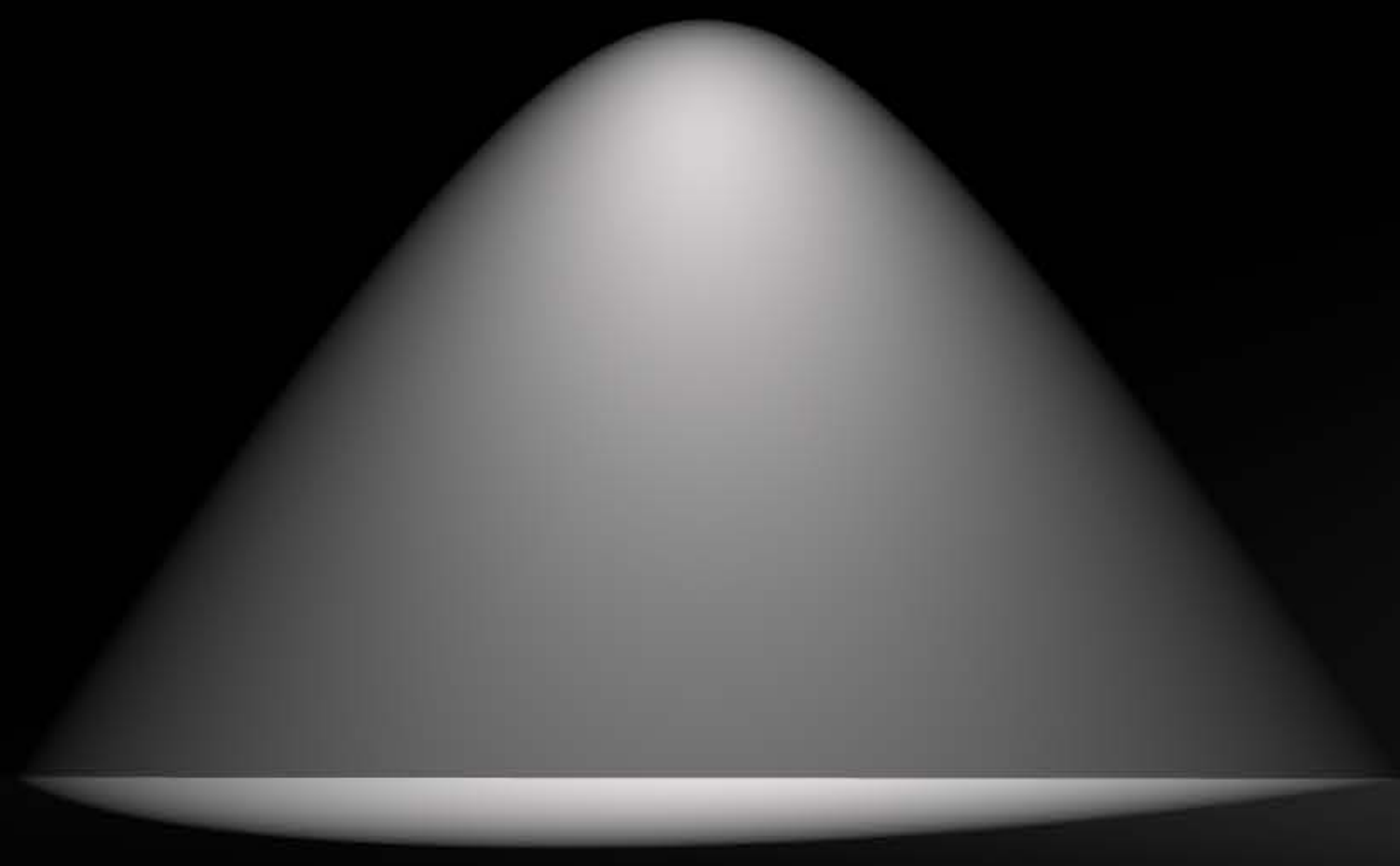
Custom attenuation shape

Physically Based Lights

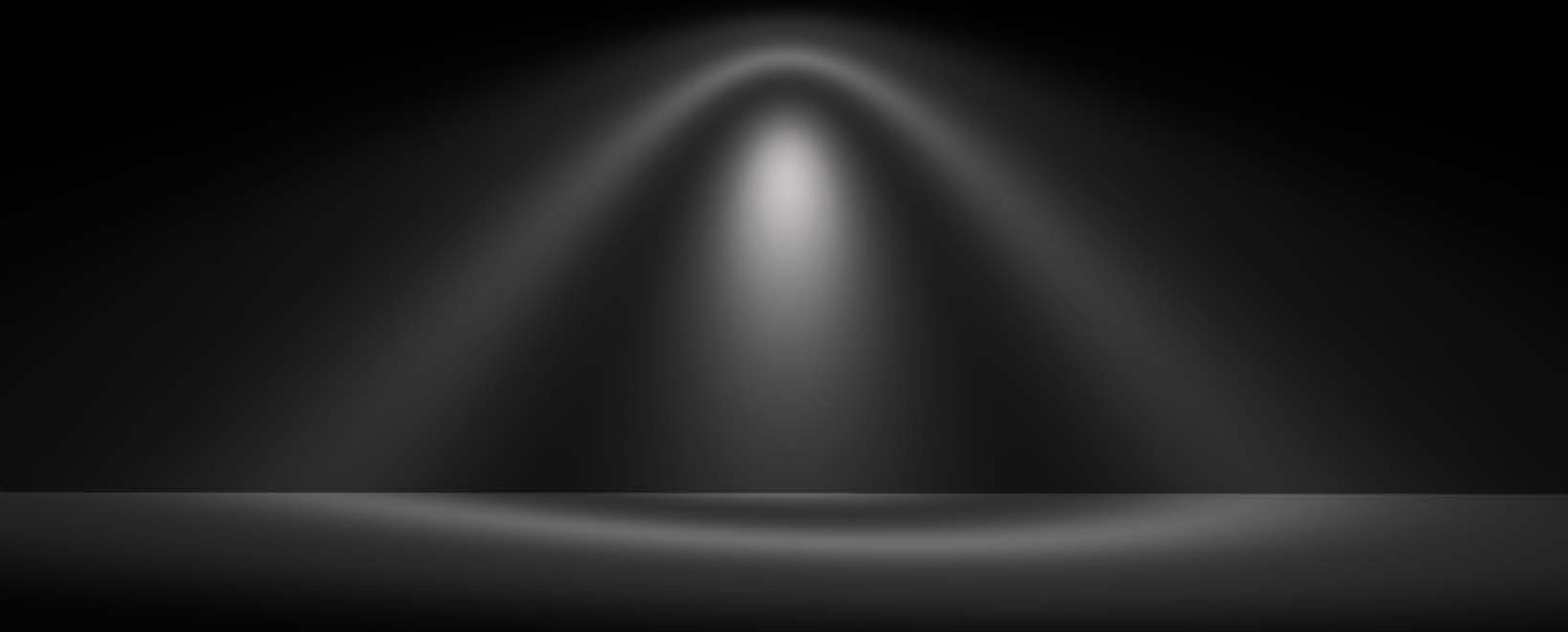
Photometric lights

Physically Based Lights

Photometric lights



Spot



IES

Physically Based Lights

NEW

Photometric lights

New kind of point light

Modeled after real-world lights

Custom attenuation shape

```
let light = SCNLight()
light.type = .IES
light.iesProfileURL = Bundle.main().urlForResource("spot", withExtension: "ies")
```

Physically Based Rendering

Recap

Physically based materials

Advanced lighting

- Image based lighting
- Light probes
- Point lights

Demo

Jean-Baptiste Bégué SceneKit Engineer
Sébastien Métrot SceneKit Engineer

Physically Based Rendering in Practice

Bob the Badger

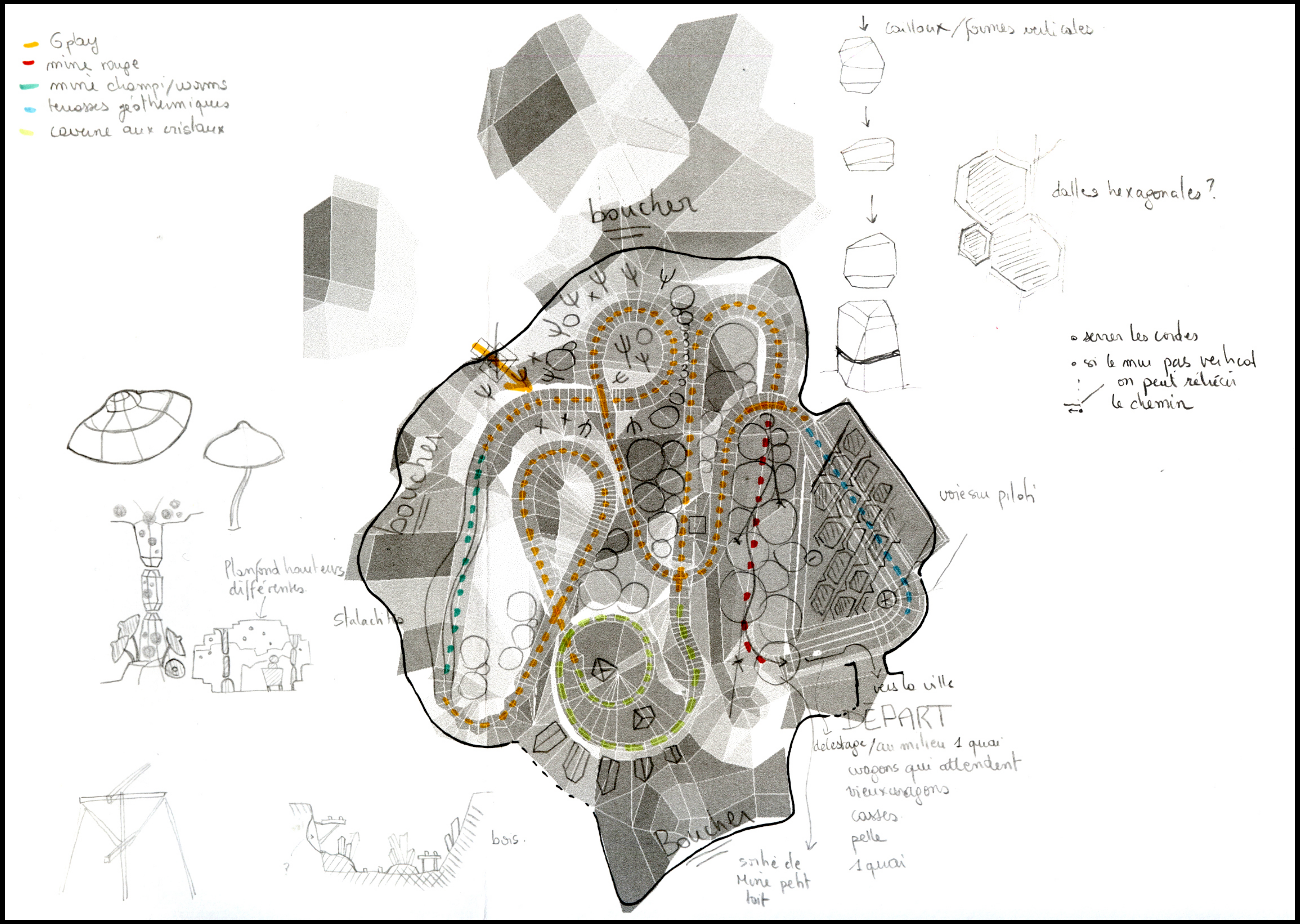
Sébastien Métrot SceneKit Engineer

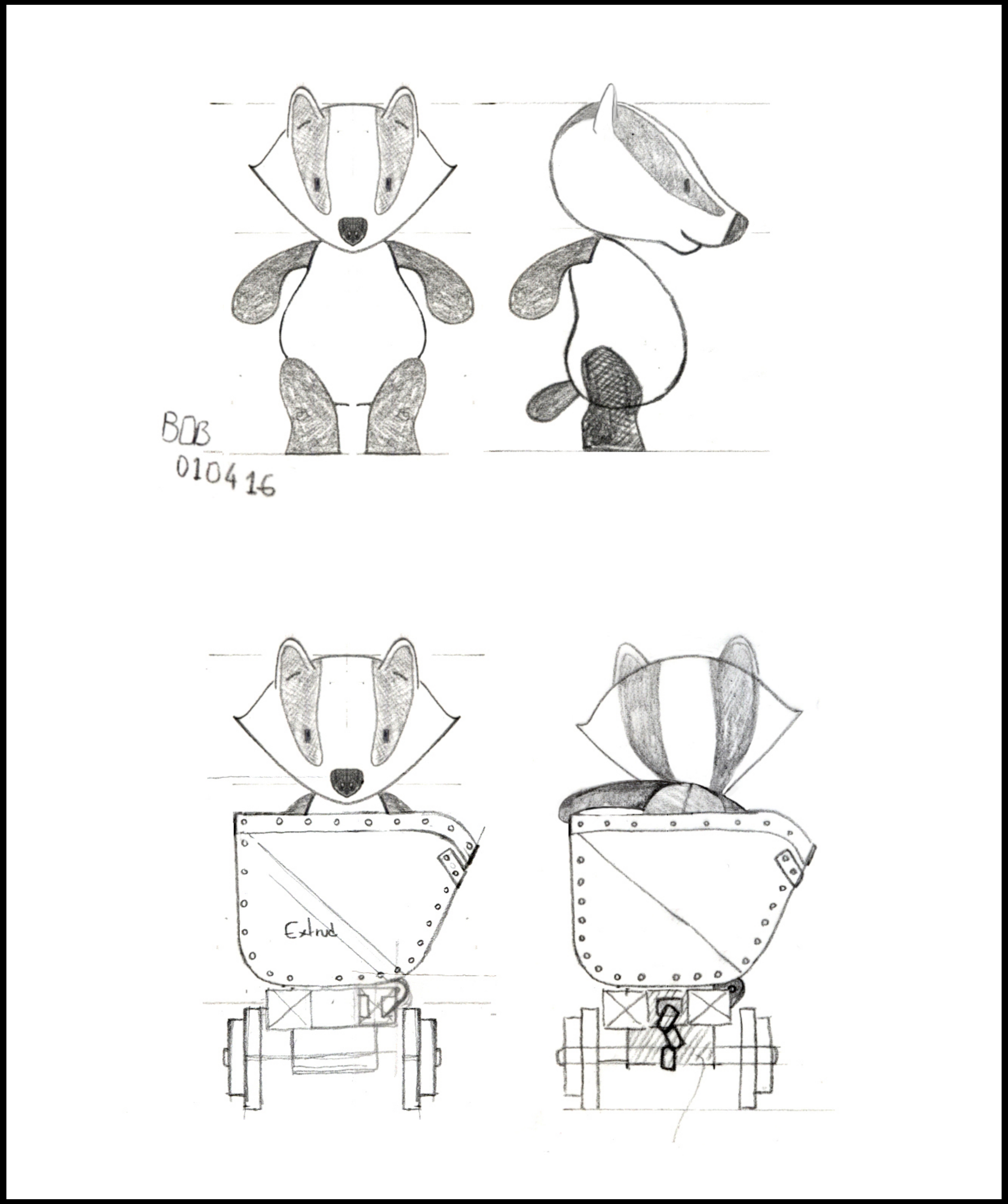
The demo

The demo
is a sample code!

Pre-Production

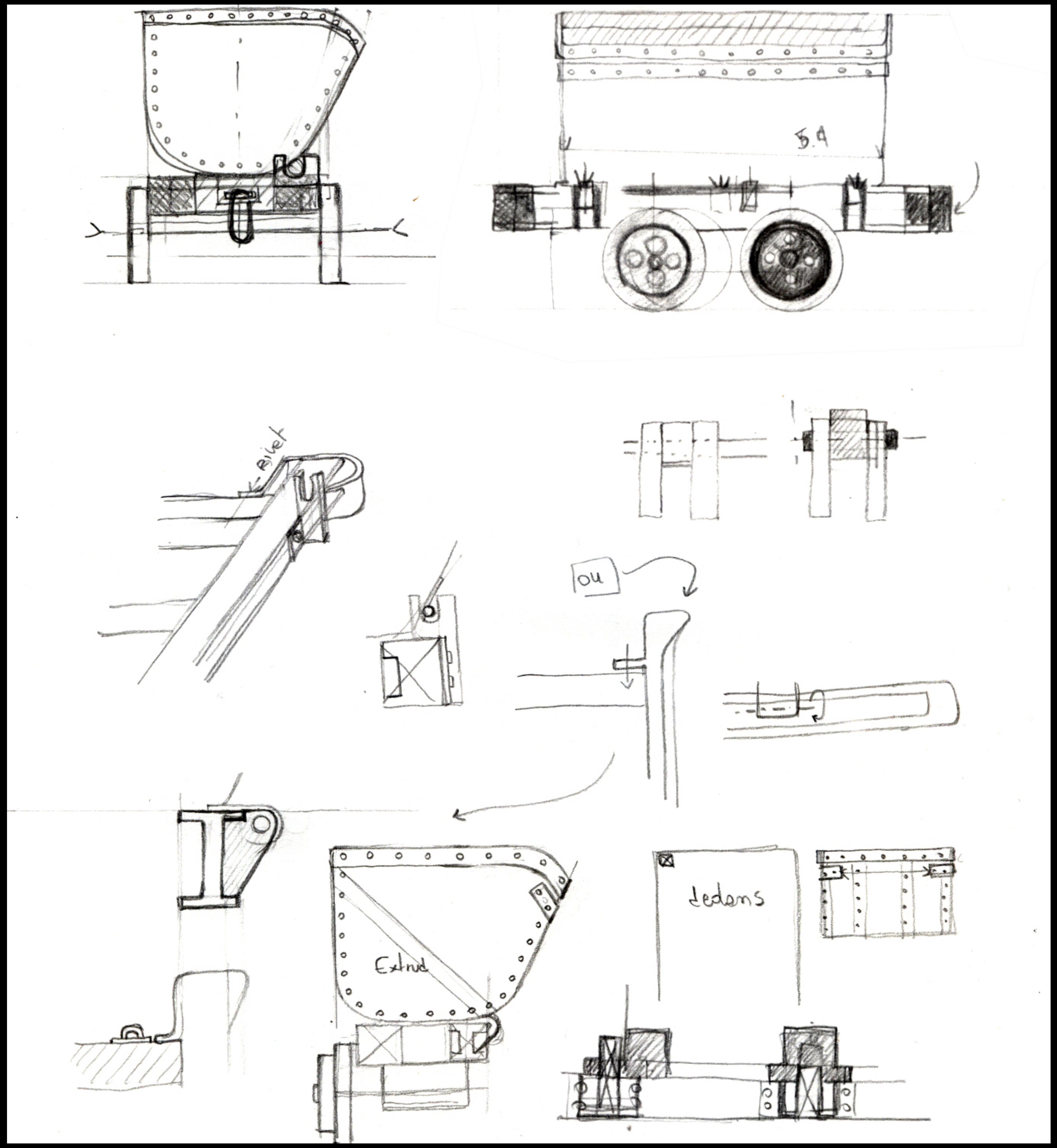
Drafts

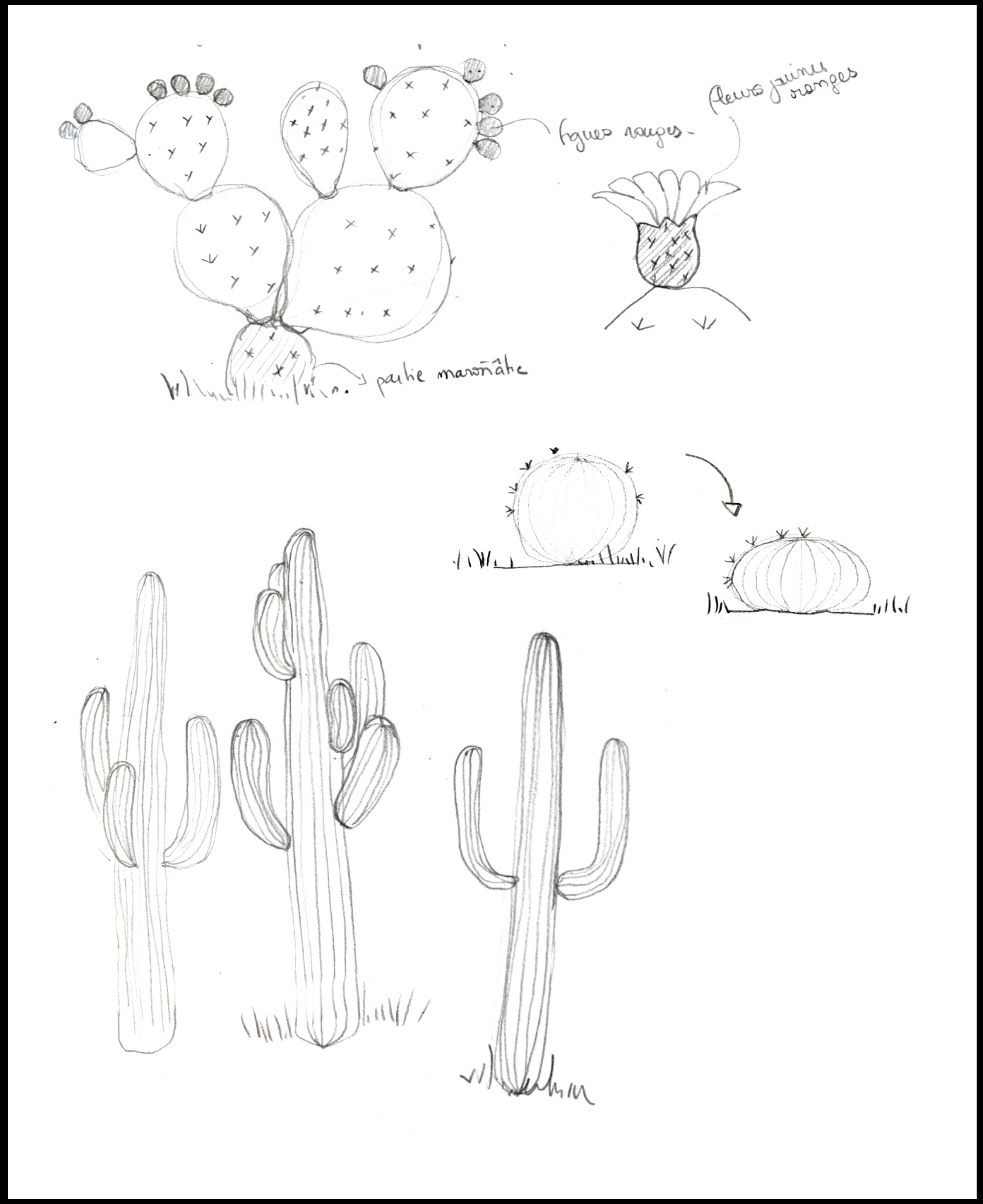




Pre-Production

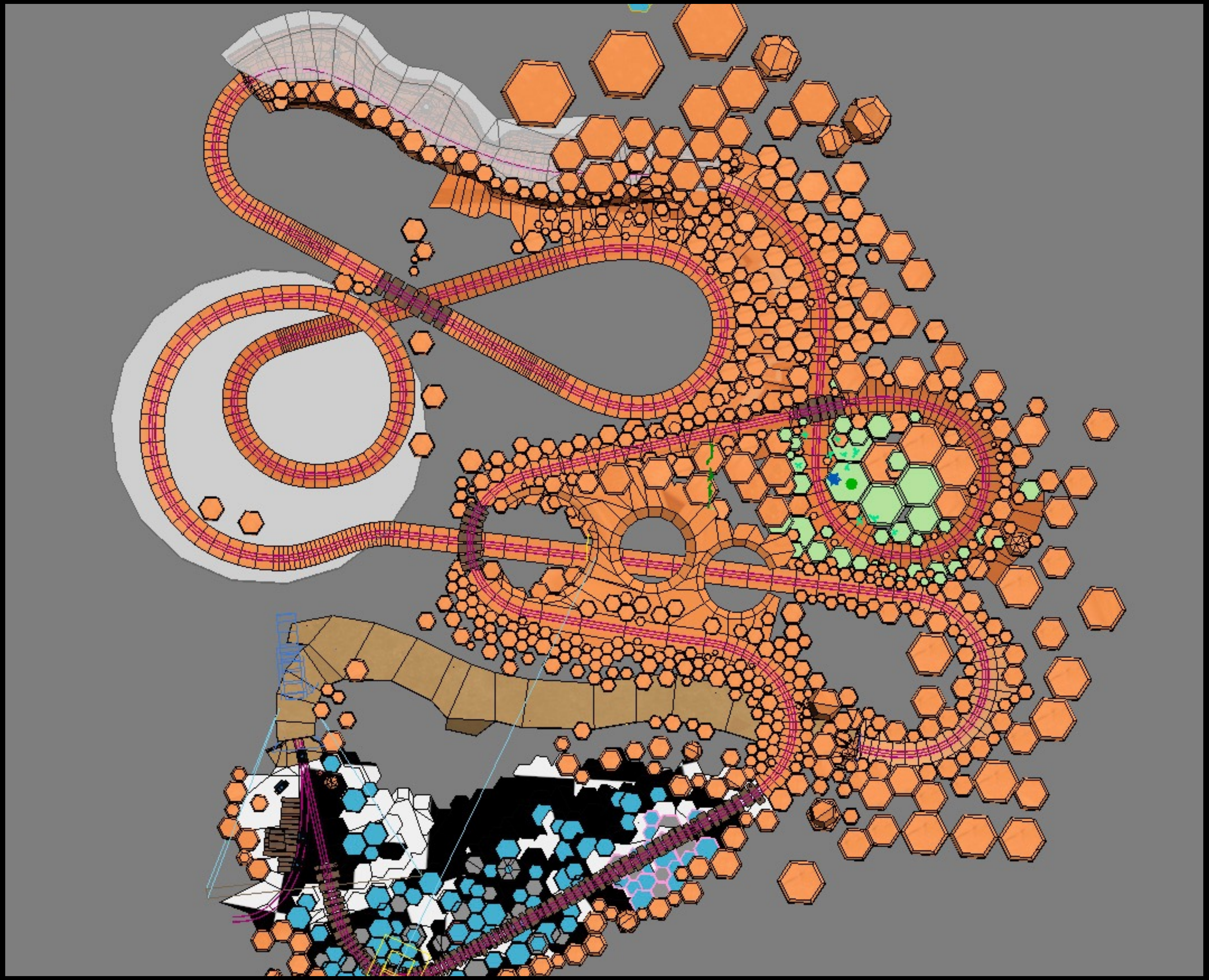
Drafts

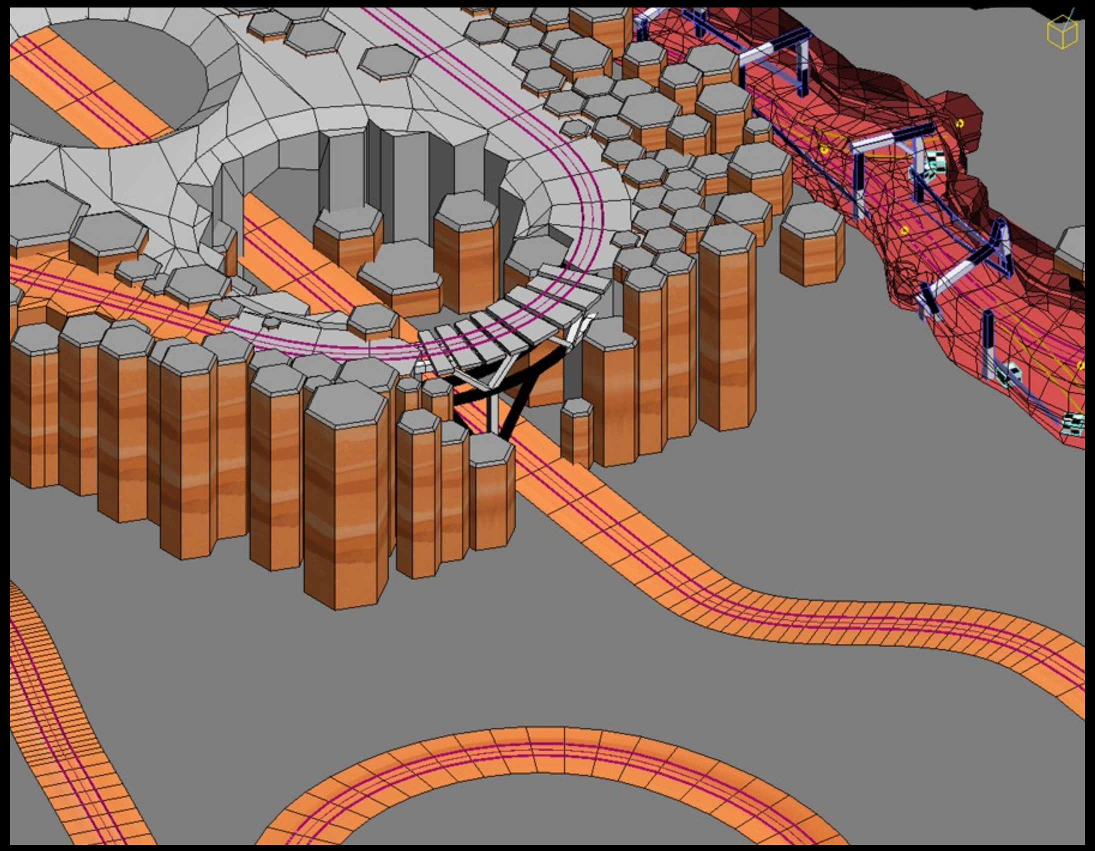


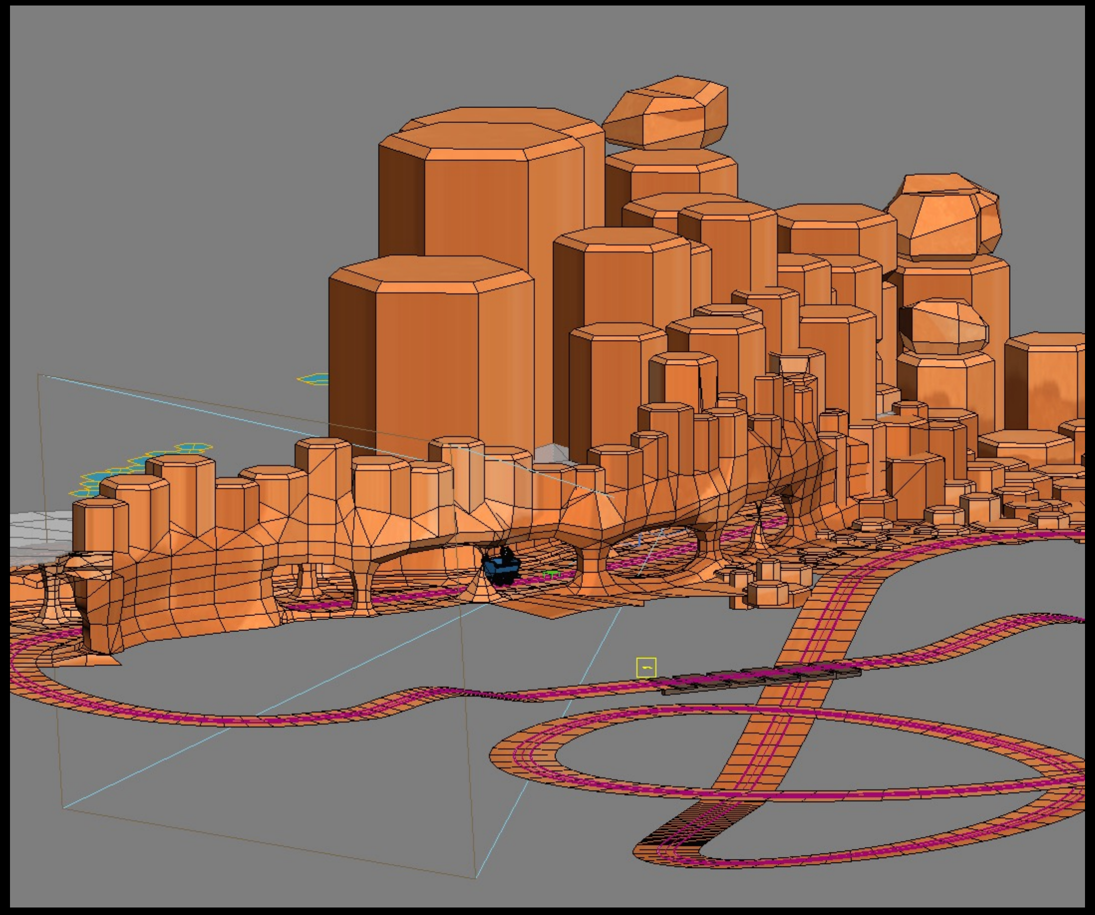


Production

Modeling







Workflow

Our artist exported models and PBR materials as DAE files

Custom tool written in SceneKit

- Import DAE file
- Convert units to meters
- Add light probes along the track

Lighting

Image based lighting

Light coming from the environment

Great for outdoor scenes

Reflections

Works with regular lights, too

Lighting

Image based lighting

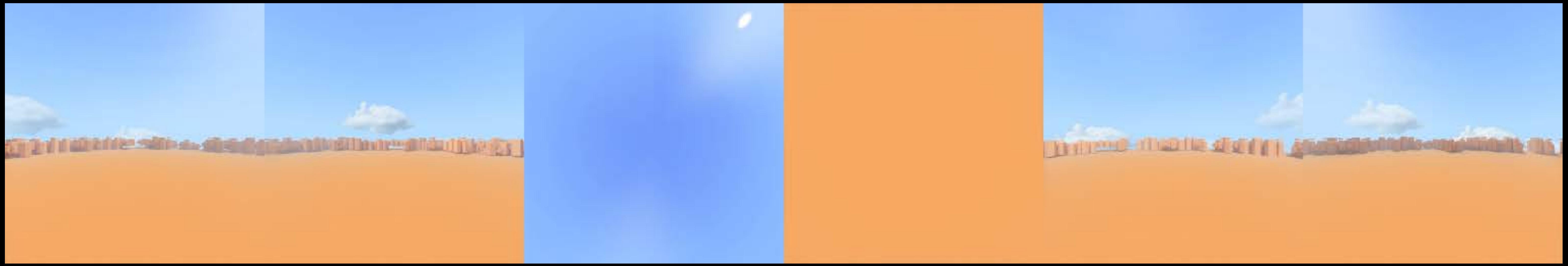
Light coming from the environment

Great for outdoor scenes

Reflections

Works with regular lights, too

Background Image



Lighting

Image based lighting

Light coming from the environment

Great for outdoor scenes

Reflections

Works with regular lights, too

Background Image

Lighting Environment

Lighting

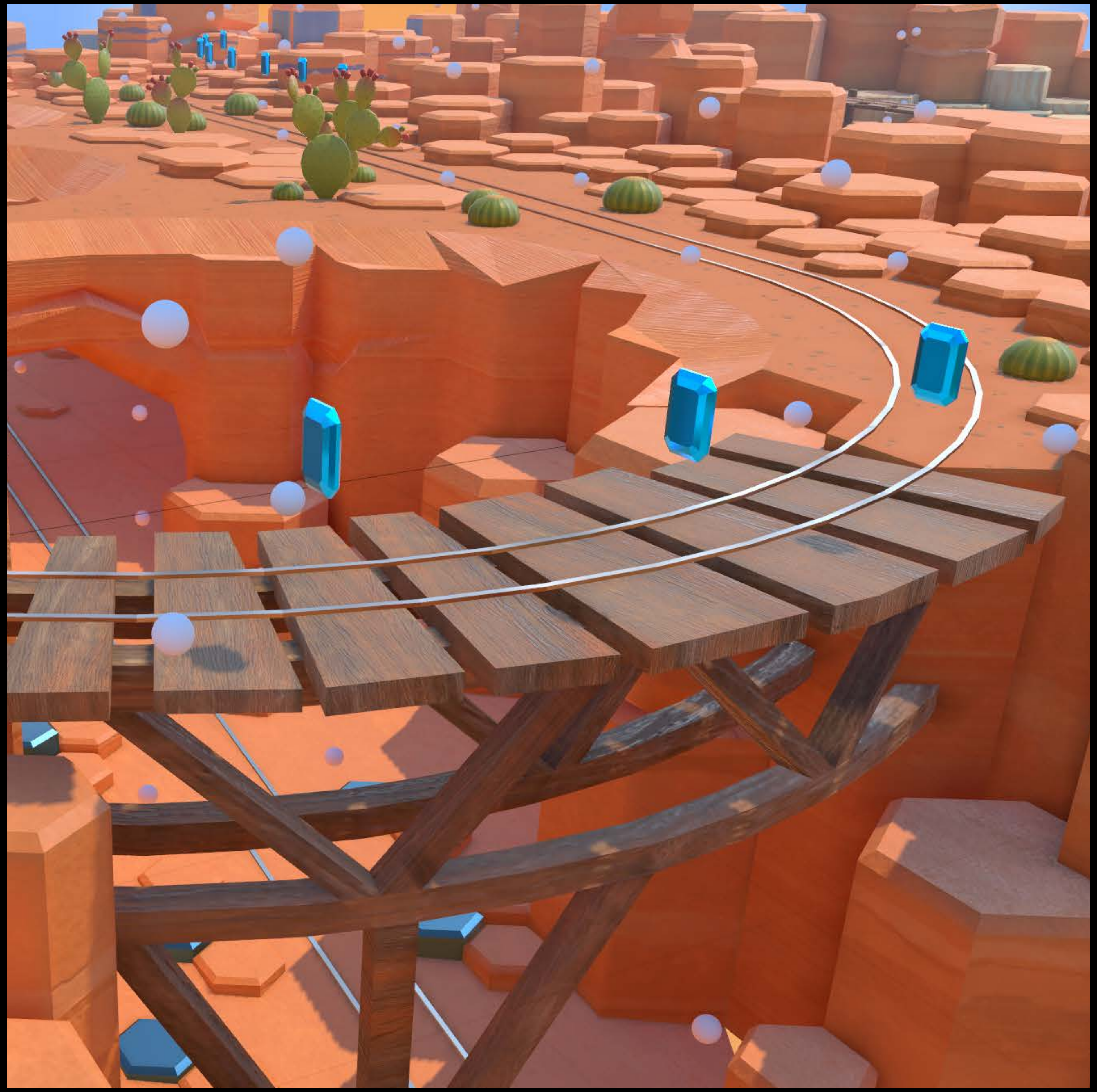
Light probes

Custom tool adds light probes along a path

They can be placed and computed in Xcode

Essential for the inside

May be optional for an outside-only scene



Lighting

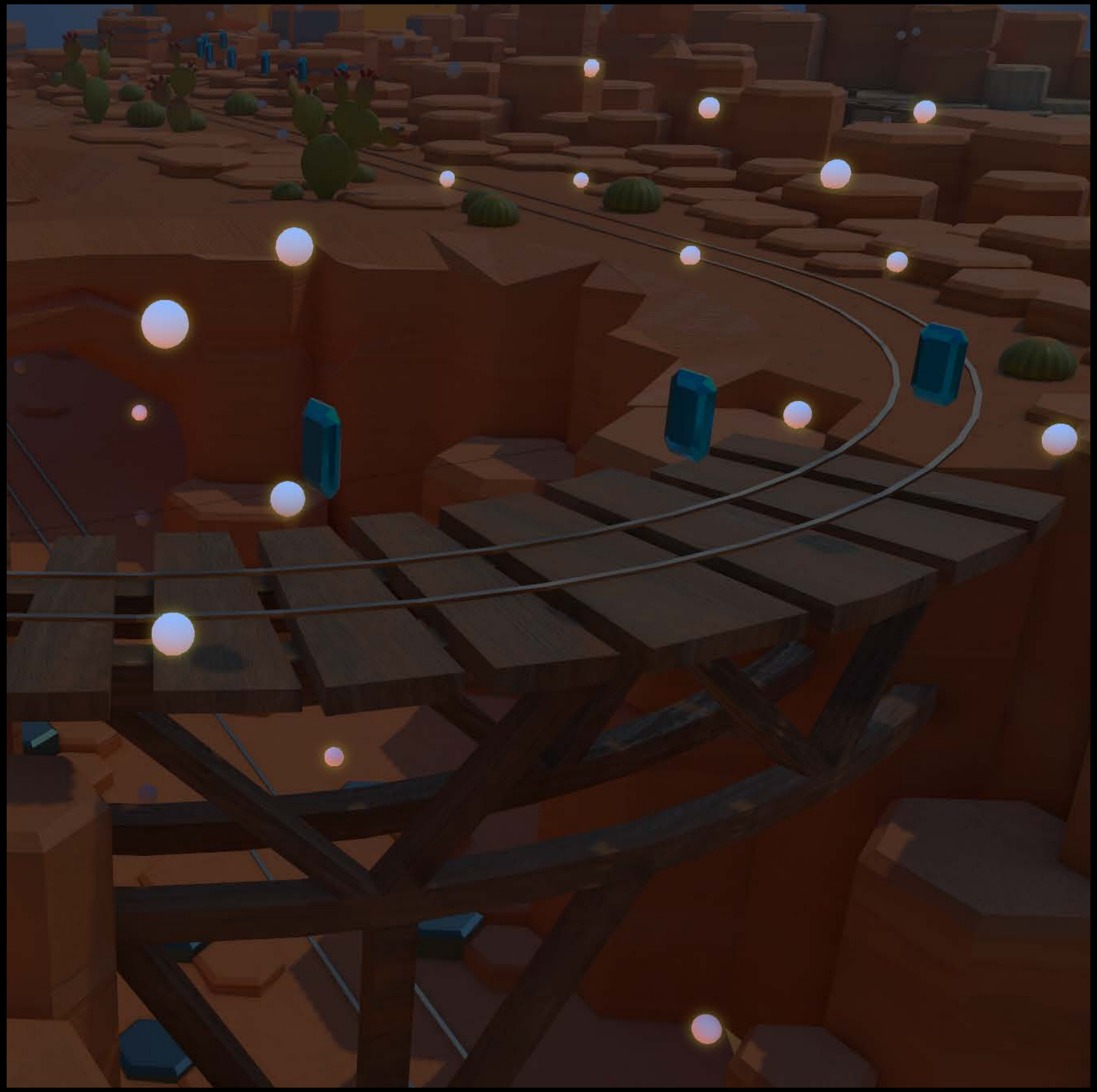
Light probes

Custom tool adds light probes along a path

They can be placed and computed in Xcode

Essential for the inside

May be optional for an outside-only scene



Lighting

Light maps

For the inside

Overrides IBL except for the specular component

```
let material = SCNMaterial()
material.selfIllumination.contents = "selfIllum.exr"
```

Lighting

Normal maps

Normal maps add detail to the models

```
let material = SCNMaterial()
material.normal.contents = "normal.png"
```

Lighting

Ambient occlusion maps

Ambient occlusion maps make global illumination more realistic

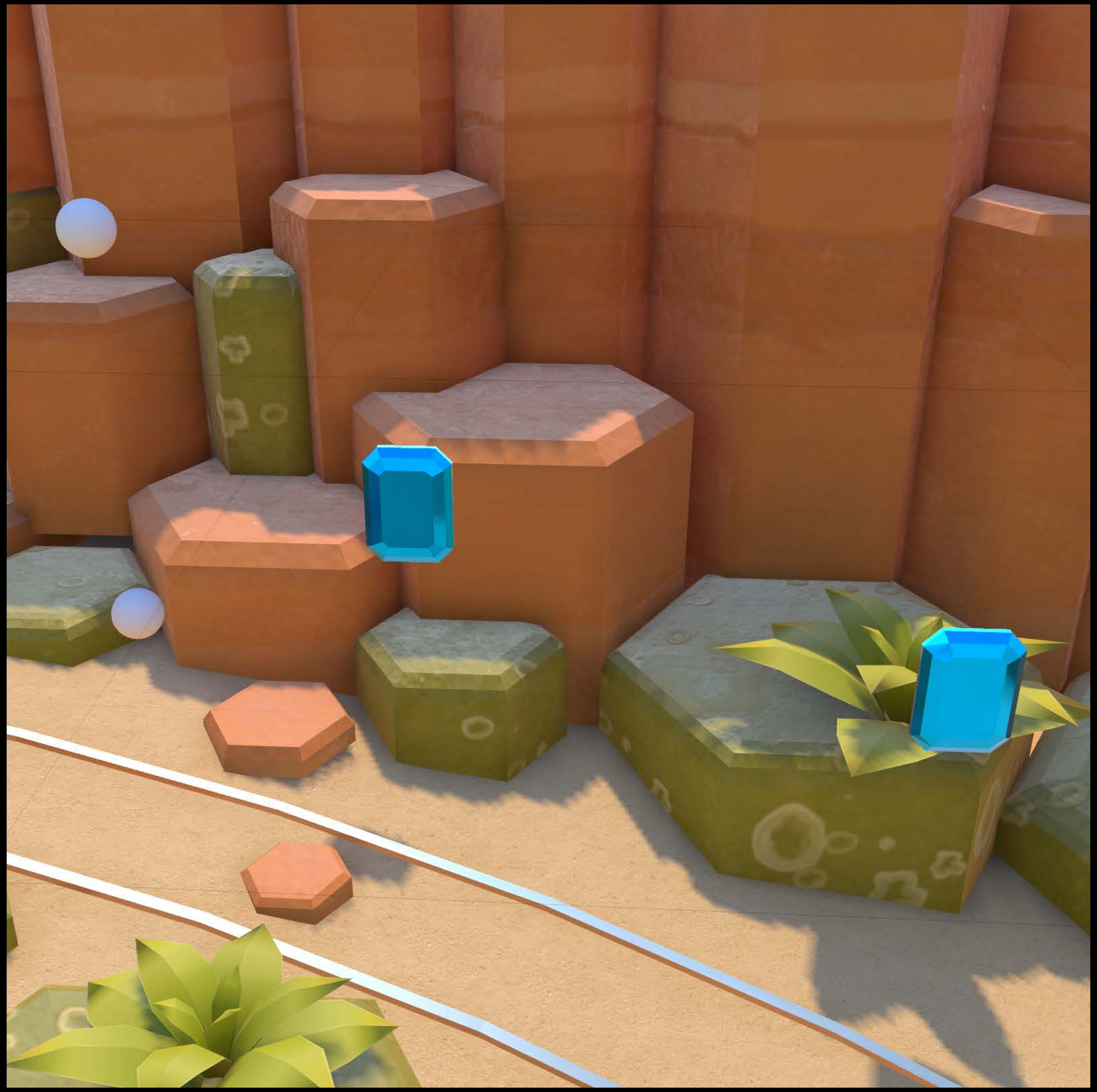
```
let material = SCNMaterial()
material.ambientOcclusion.contents = "ao.png"
```

Lighting

Point lights

One global dynamic light high above the scene

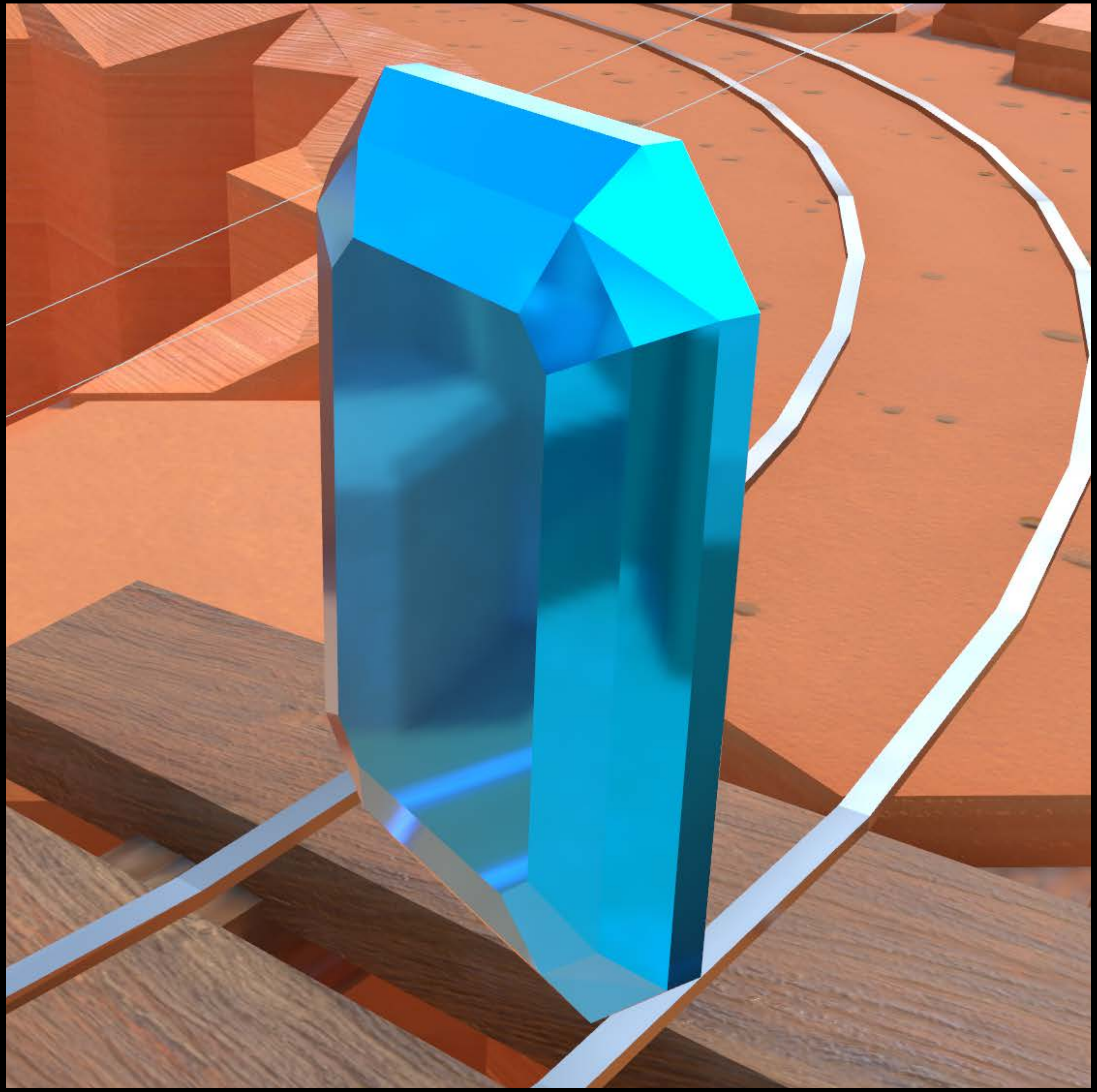
- Create shadows
- Improve global lighting

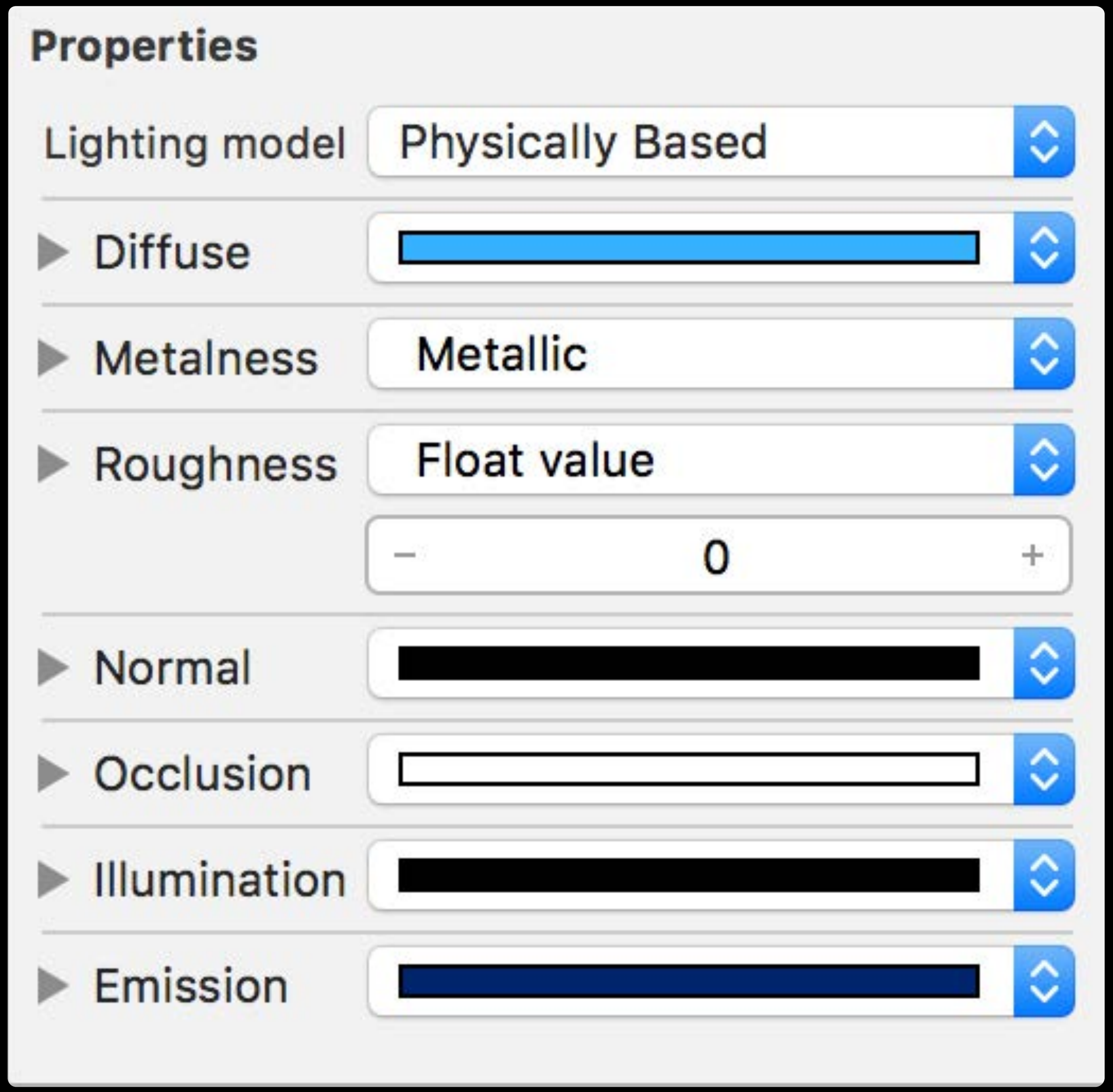


Materials

100% physically based materials!

Materials





Properties

Lighting model: Physically Based

Diffuse

Metalness: Metallic

Roughness: Float value

0

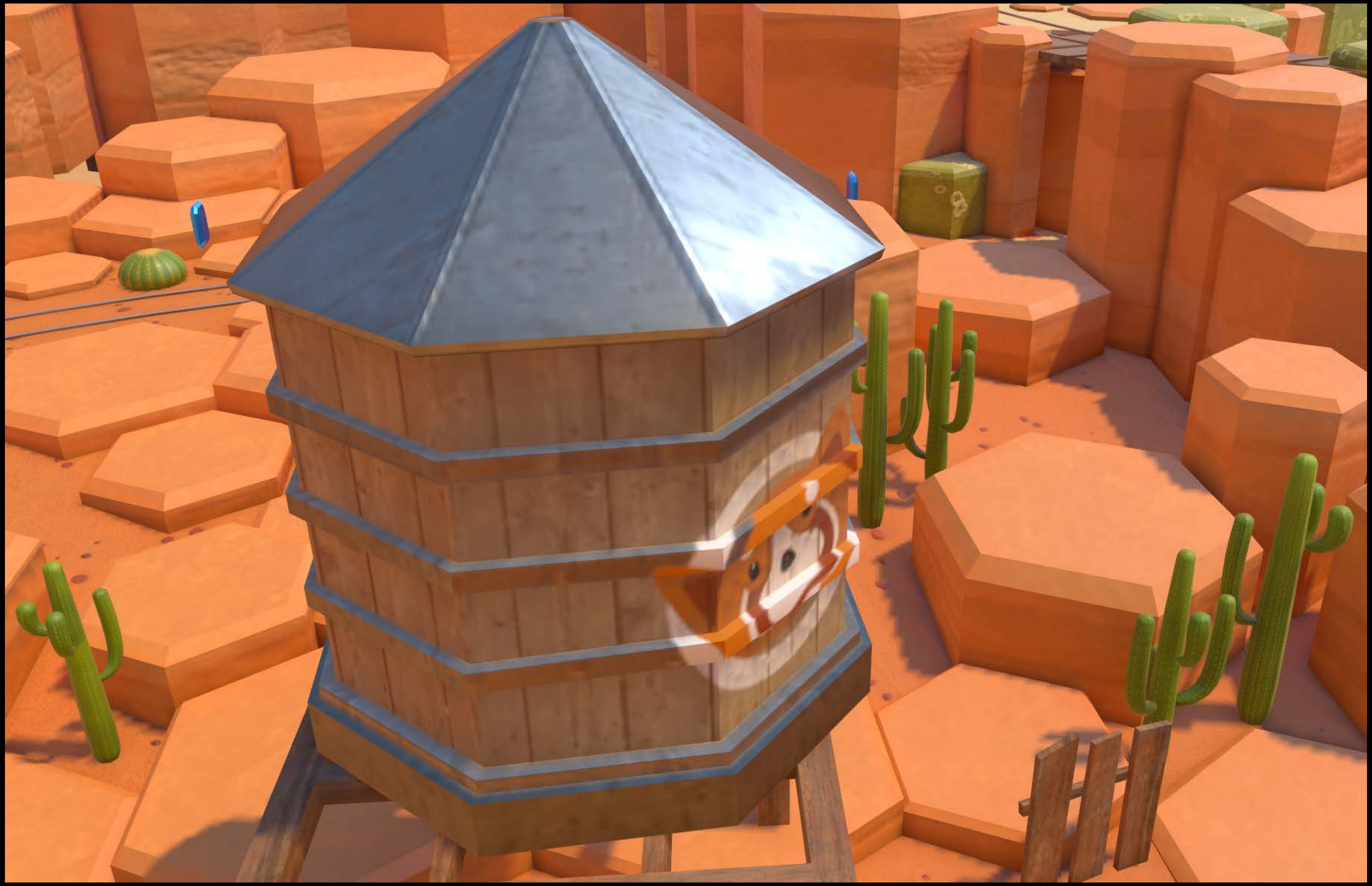
Normal

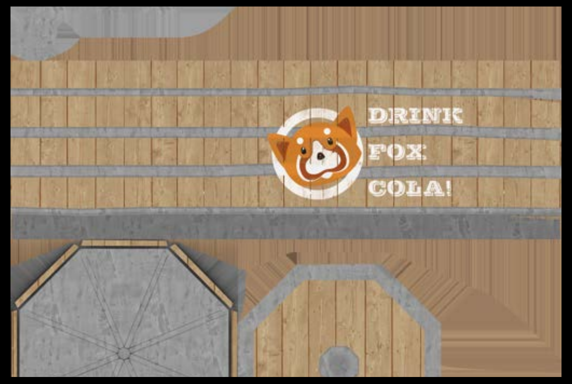
Occlusion

Illumination

Emission

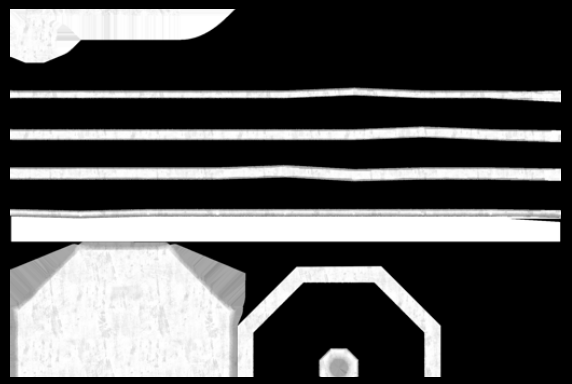
Materials



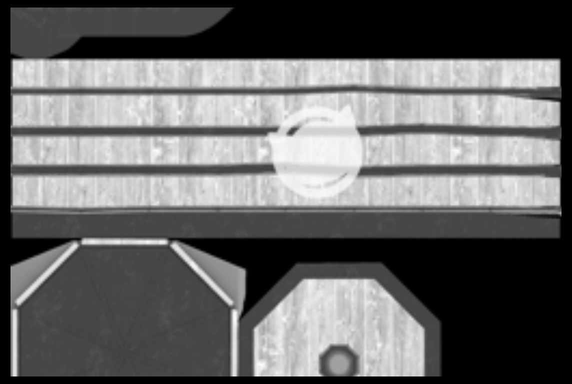


Diffuse

Normal



Metalness



Roughness

Physically Based Rendering

Summary

Physically based shading

SceneKit APIs for materials and lights

Xcode integration

Showcase demo and sample code

HDR Camera and Effects

HDR Camera

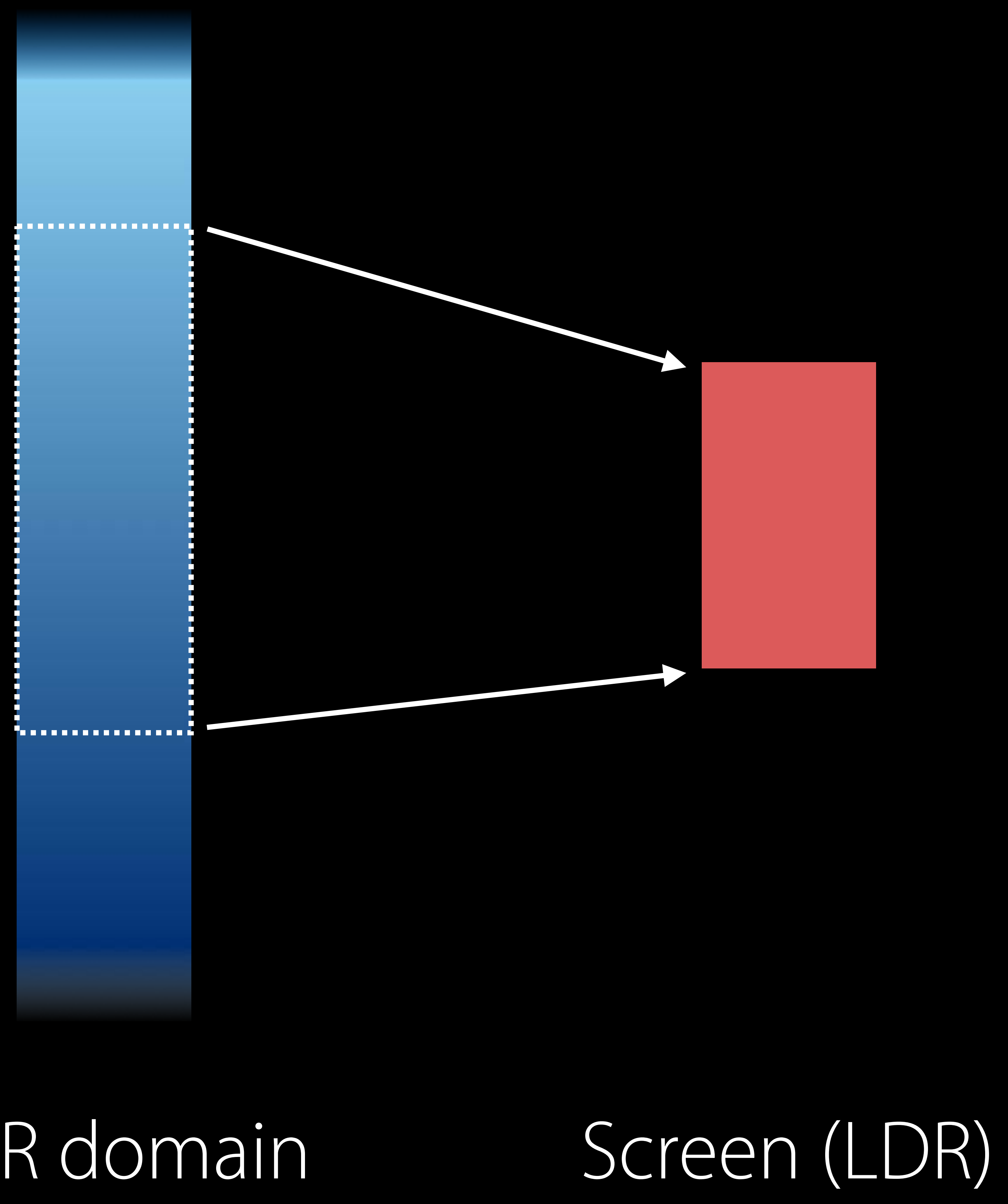
HDR is short for High Dynamic Range

Float components

Low dynamic range: 8 bits per components

HDR extends that range

Tone mapped to LDR screens



HDR Camera

NEW

Needed for High Dynamic Range contents

Can also be used with normal contents but realistic light ranges

```swift
let camera = SCNCamera()
camera.wantsHDR = true
```

HDR Camera

Tone mapping

NEW

Converts from HDR to LDR

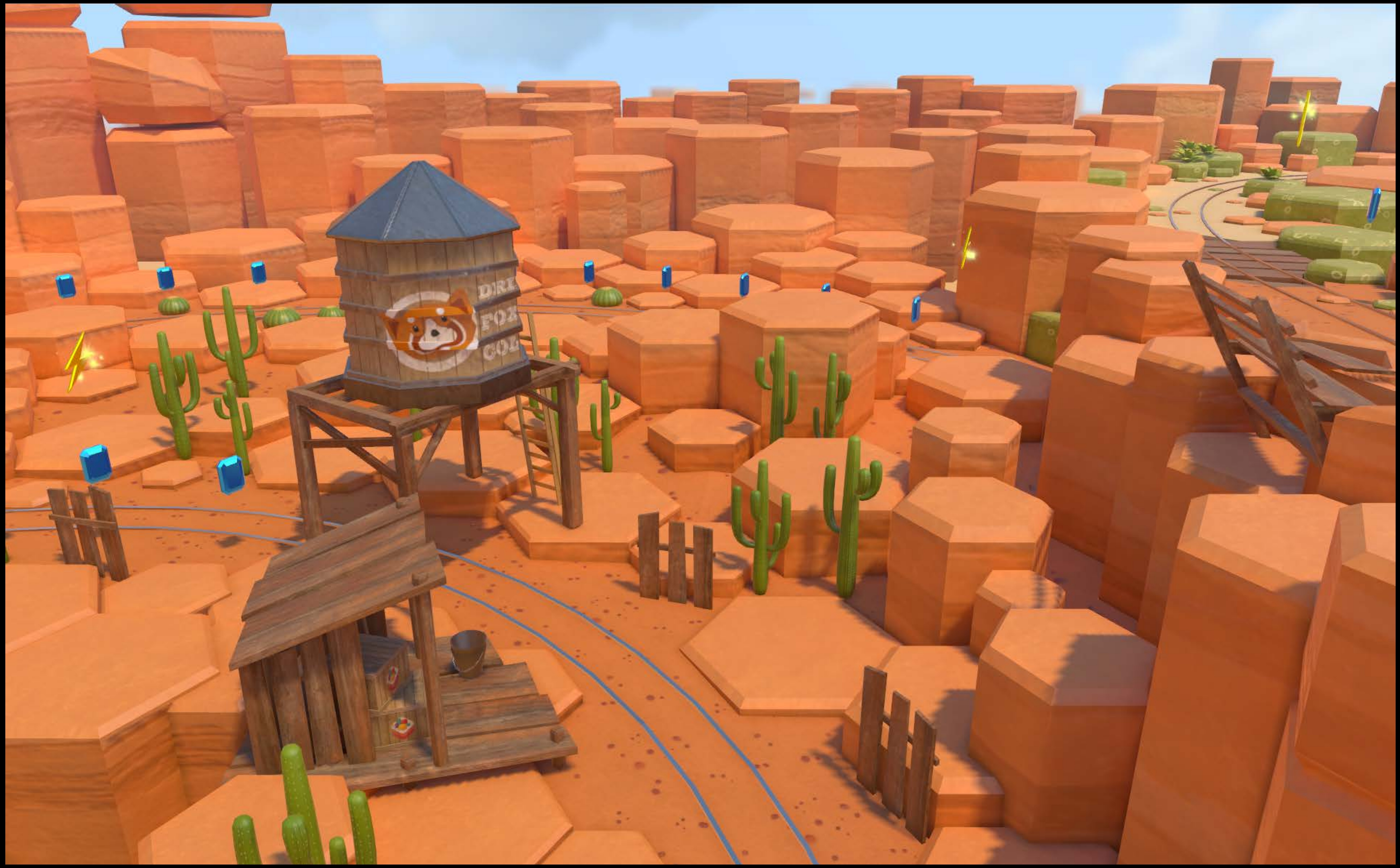
Automatic eye adaptation

Configurable (gray point, white point, min/max exposure)

```
camera.wantsExposureAdaptation = true
camera.averageGray = 0.5
camera.whitePoint = 0.5
camera.exposureOffset = 2.5
camera.minimumExposure = -20.0
camera.maximumExposure = 10.0
```

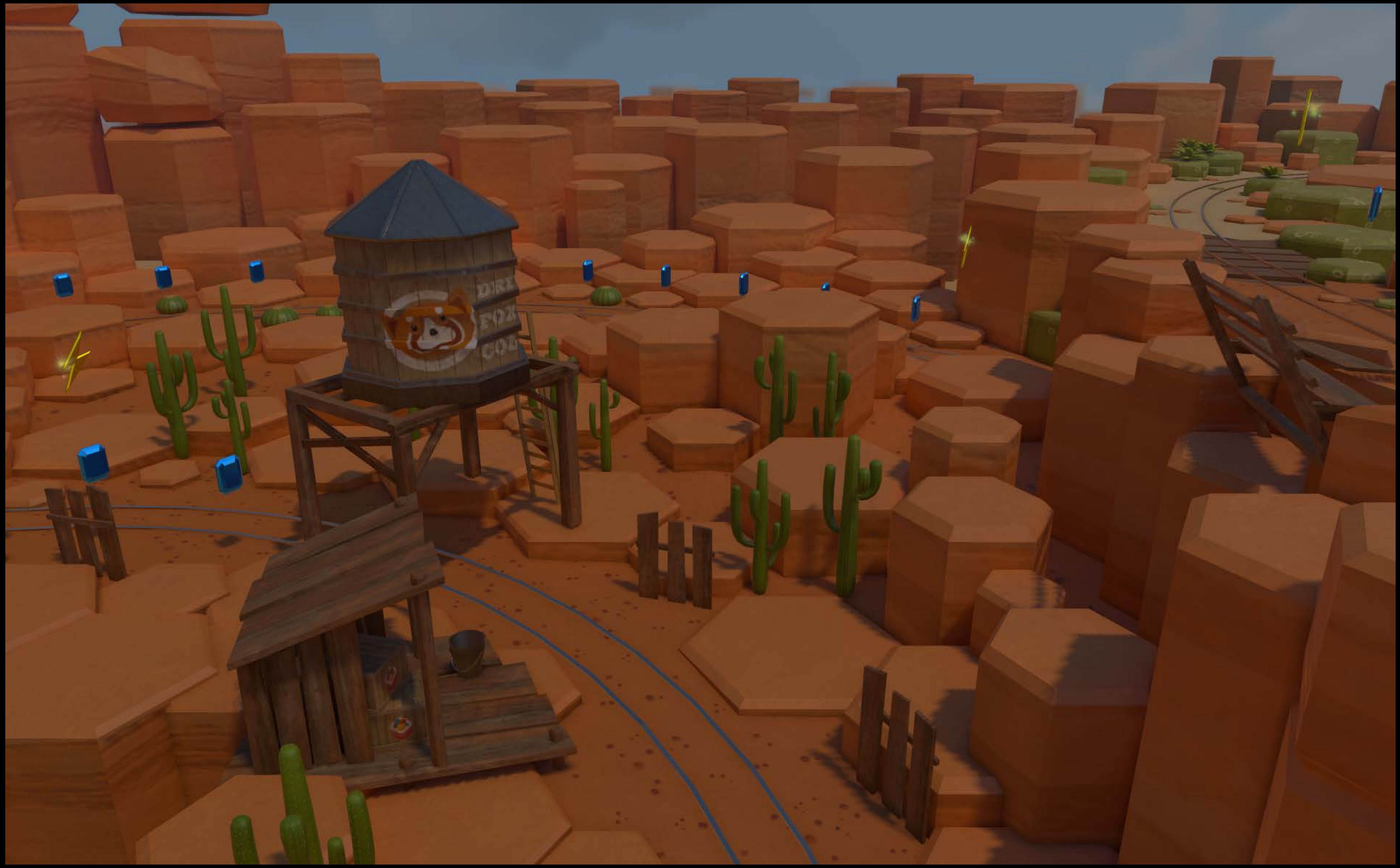
HDR Camera

Default exposure



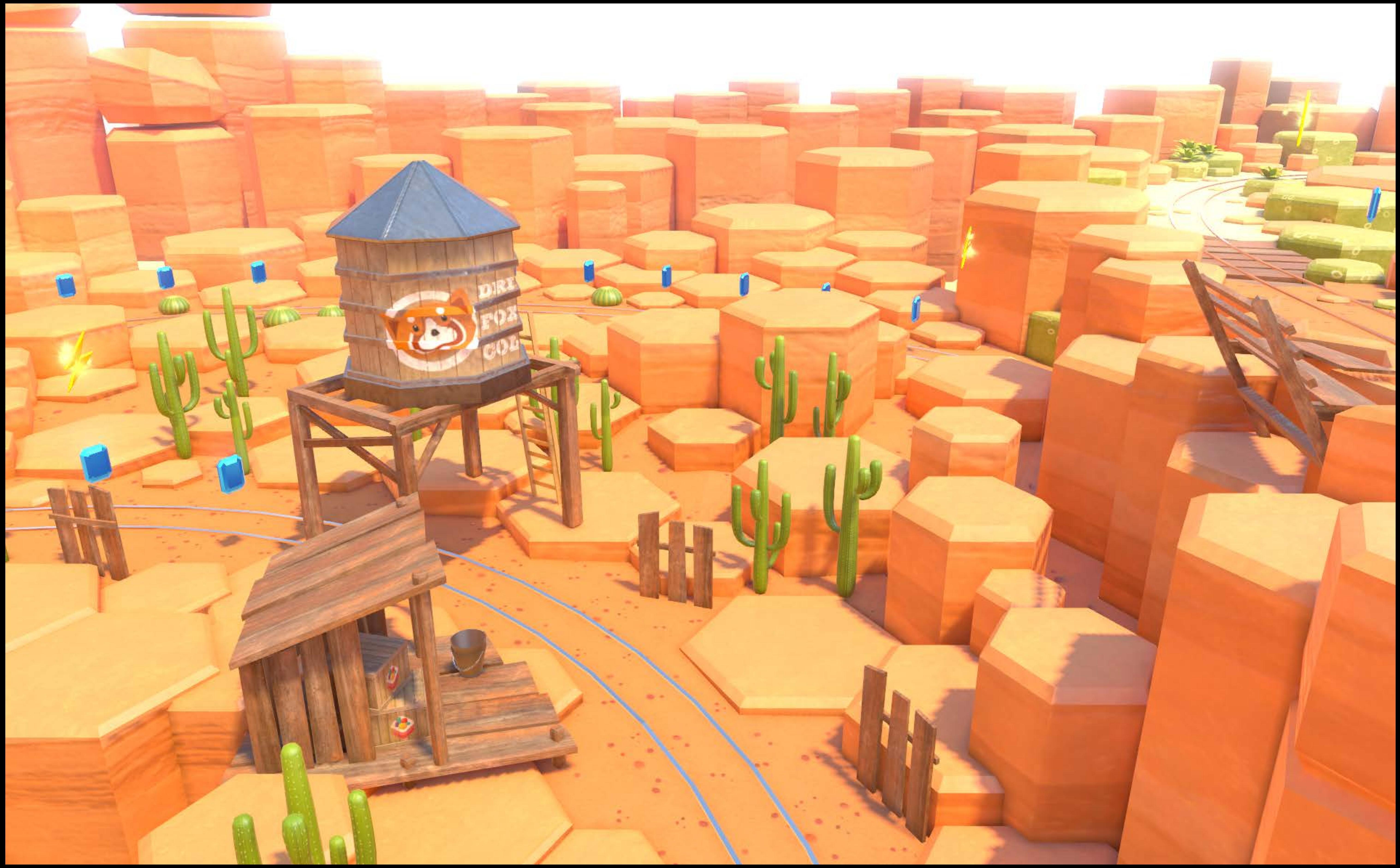
HDR Camera

Under exposure



HDR Camera

Over exposure



Effects

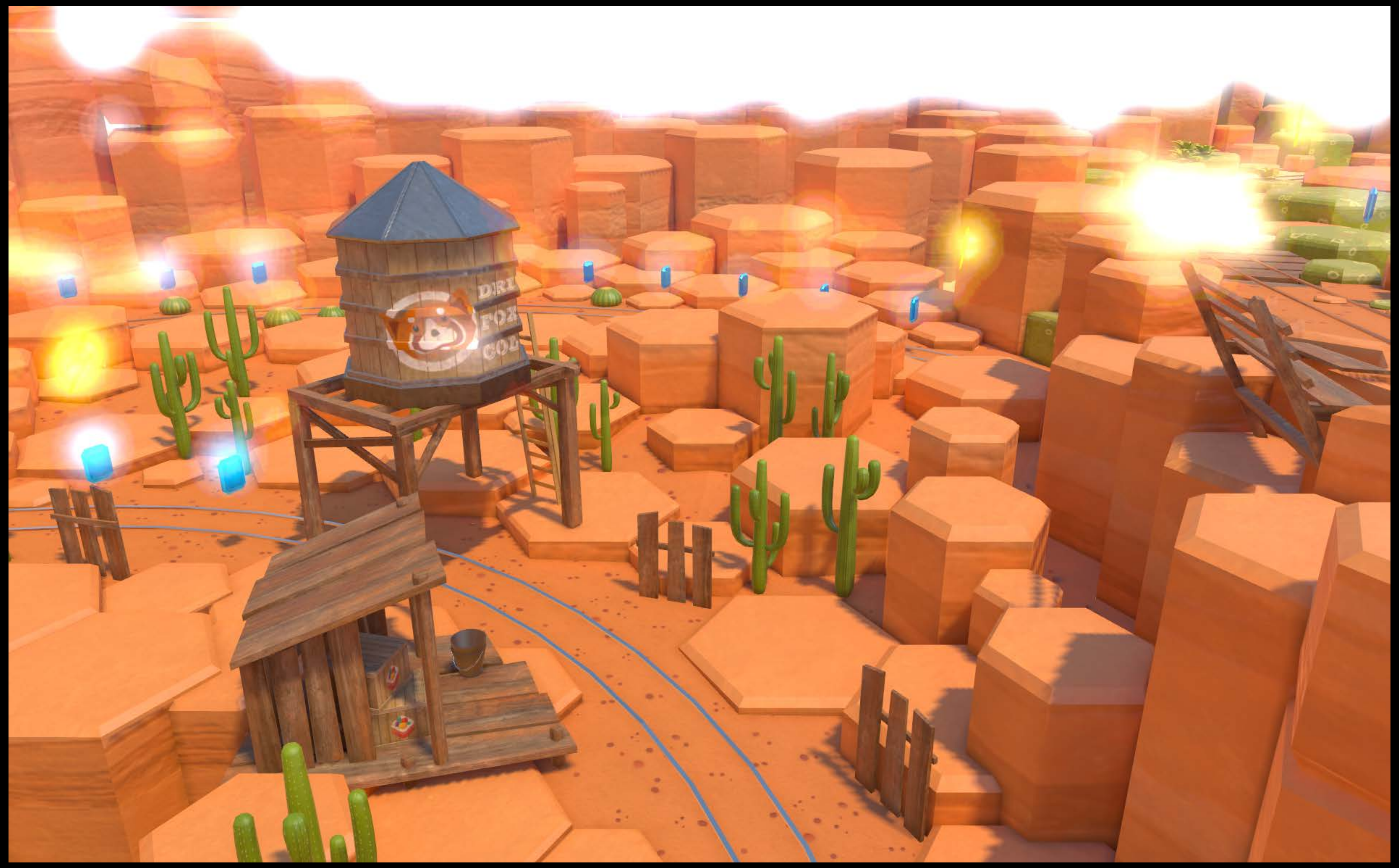
Bloom

NEW

High-intensity lights and reflections bleeding on the surrounding pixels

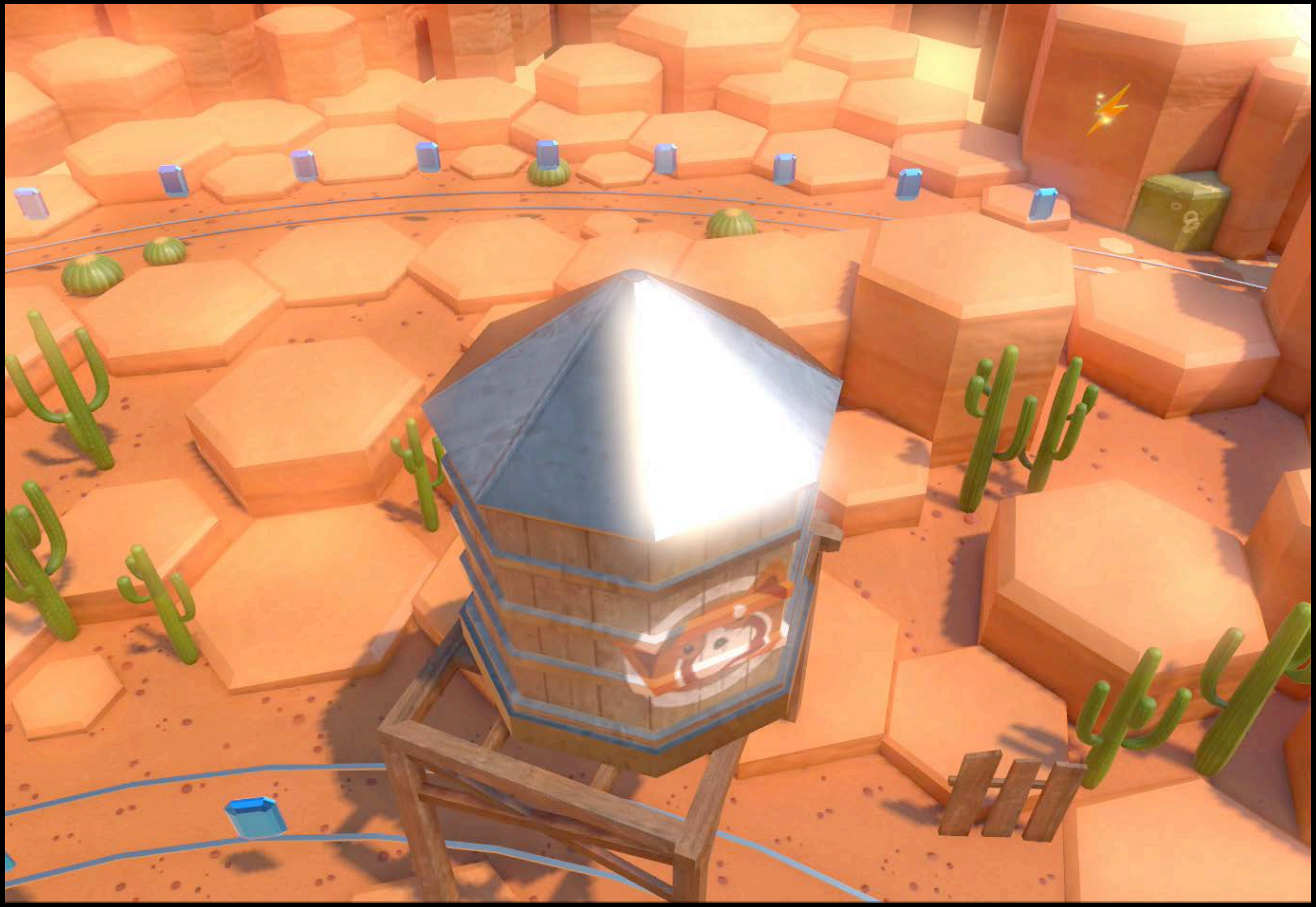
Simulates the effect of being blinded by looking at a bright light

```
camera.bloomThreshold = 0.5
camera.bloomIntensity = 1.5
camera.bloomBlurRadius = 2.5
```



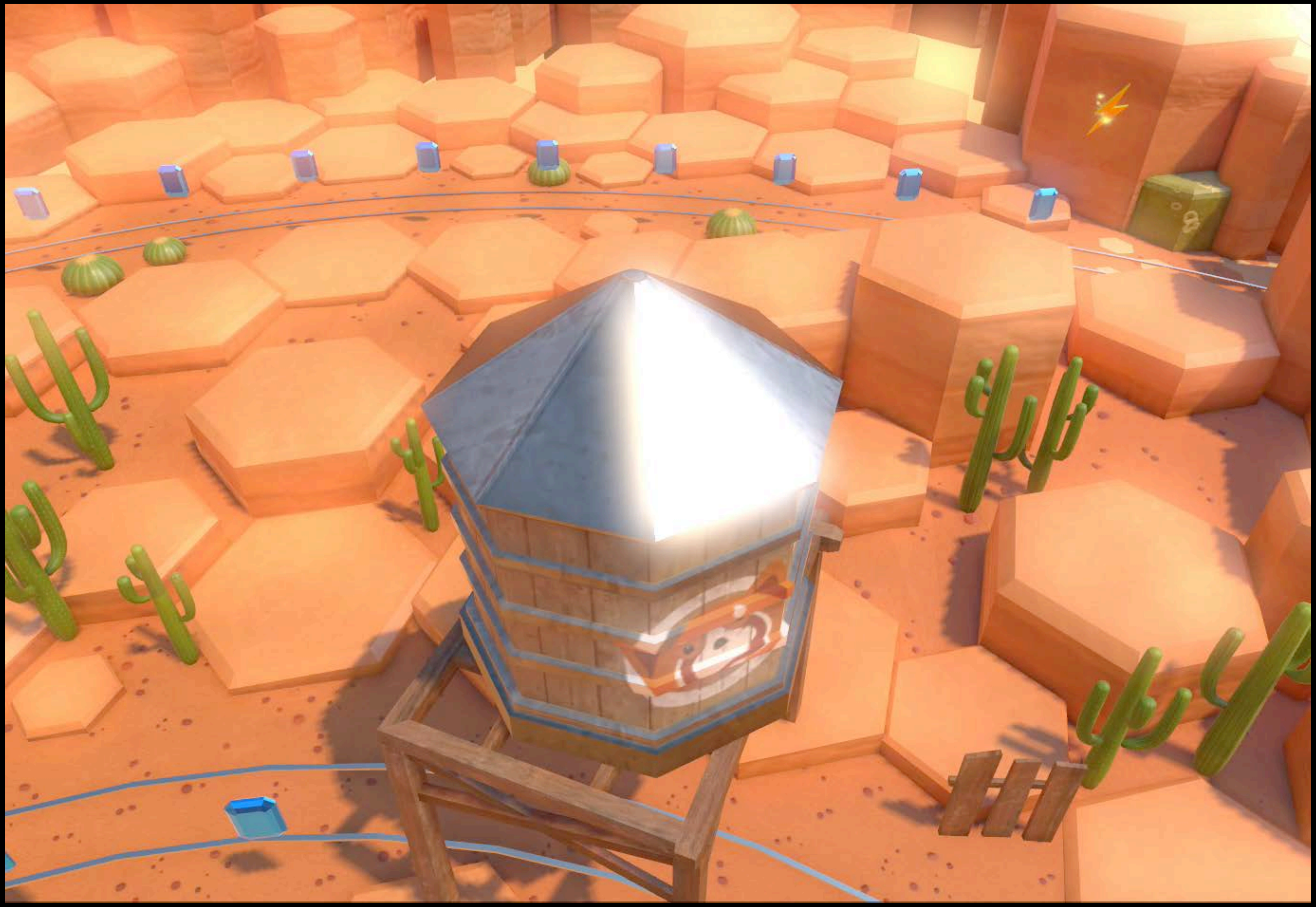
Effects

Bloom



Effects

Bloom



Effects

Motion blur

NEW

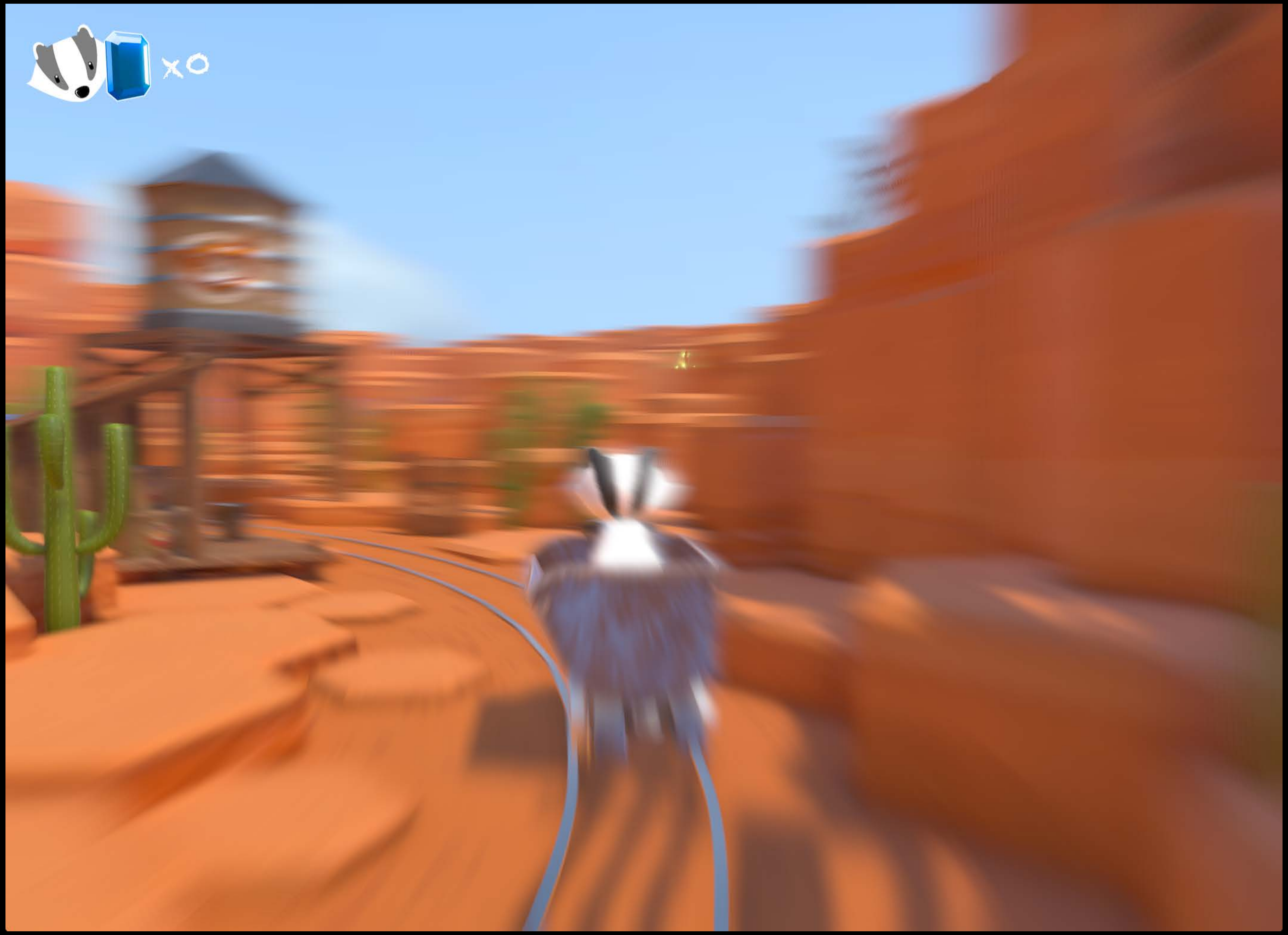
Smoothens camera movements

Some objects can be excluded from the motion blur

```
camera.motionBlurIntensity = 0.2
```

Effects

Motion blur



Effects

Motion blur

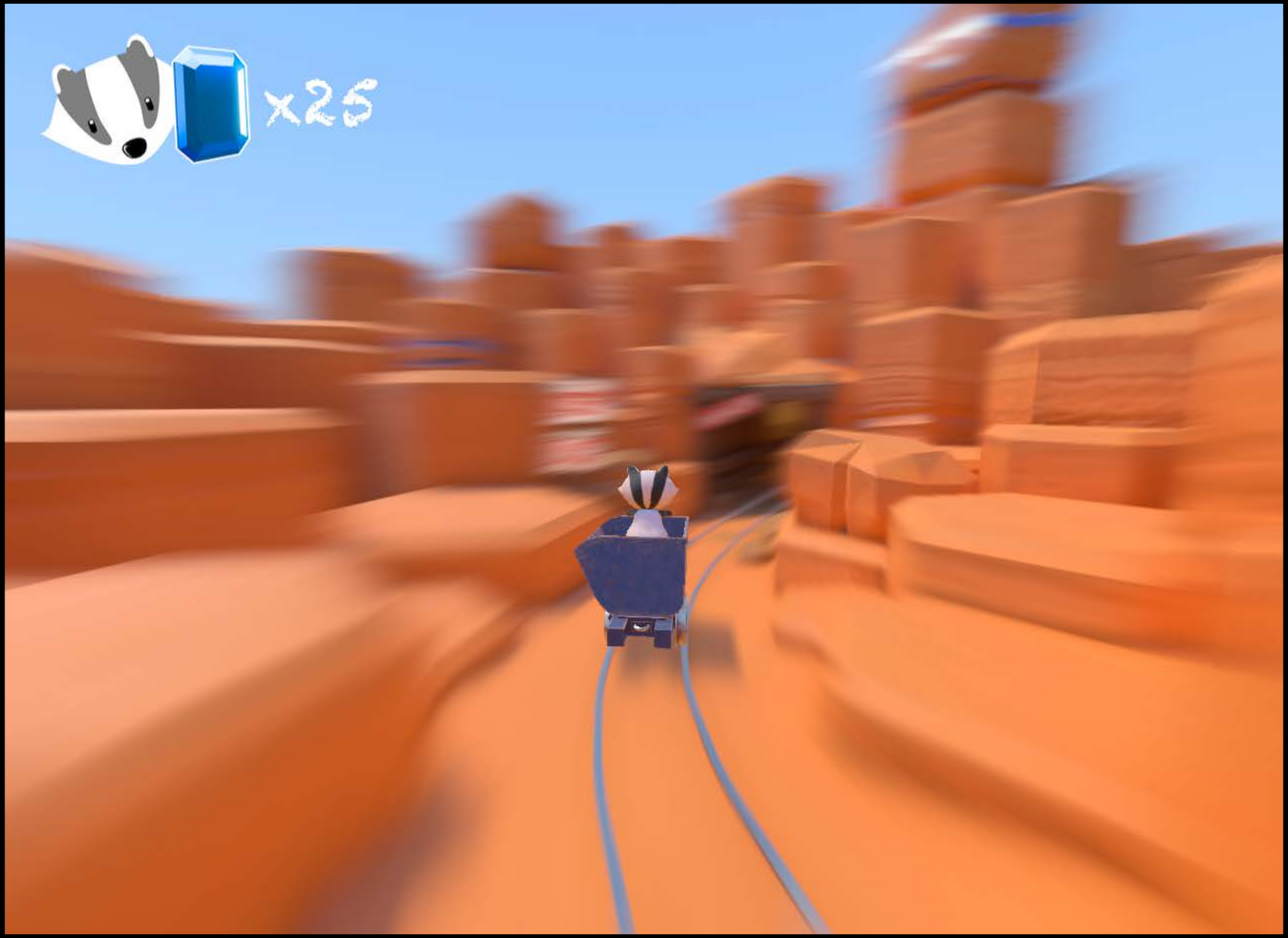
NEW

Use movability hint to exclude nodes from the motion blur

```
character.movabilityHint = .movable
```

Effects

Motion blur: Movability hint



Effects

Vignetting

NEW

Simulates the round shading aberrations of real camera lenses

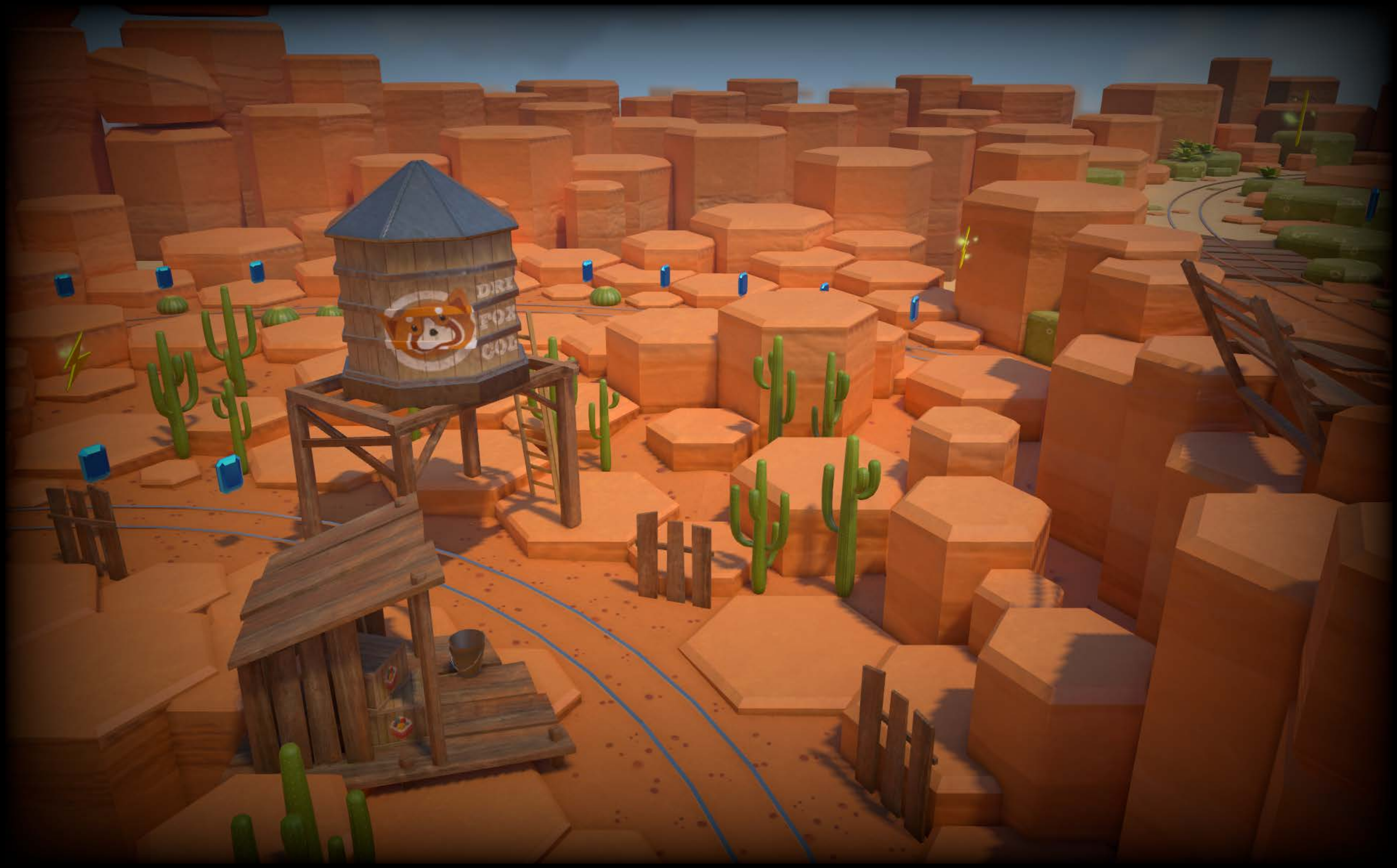
```
camera.vignettingPower = 0.2
camera.vignettingIntensity = 1.2
```

Effects

Vignetting

Effects

Vignetting



Effects

Color fringe

NEW

Simulates the chromatic aberrations happening in real lenses

```
camera.colorFringeStrength = 0.2
camera.colorFringeIntensity = 0.8
```

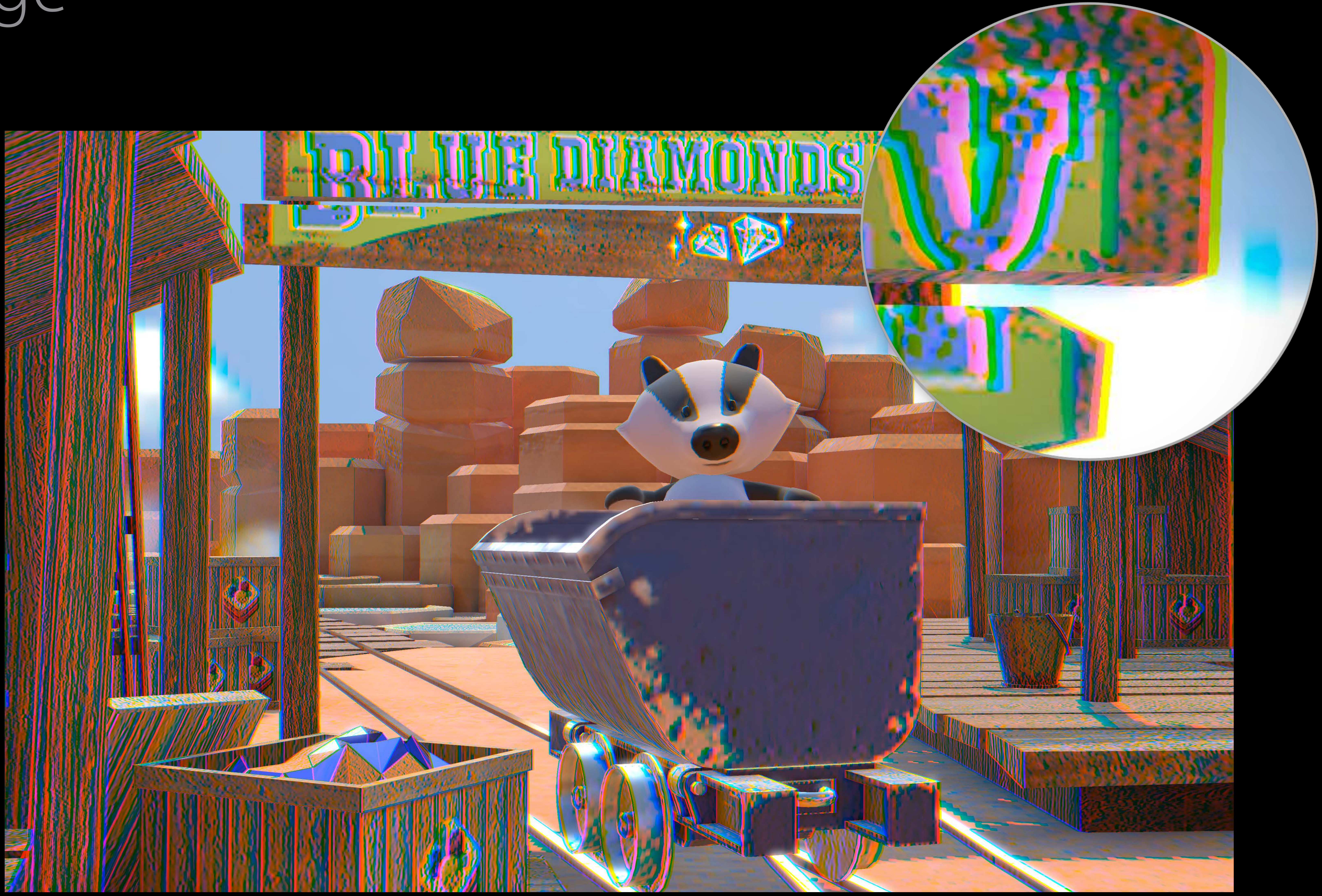
Effects

Color fringe

BLUE DIAMONDS VALLEY

Effects

Color fringe



Effects

Color correction

NEW

Saturation

- Easy black and white look
- Overblown colors

Contrast

- More intense look

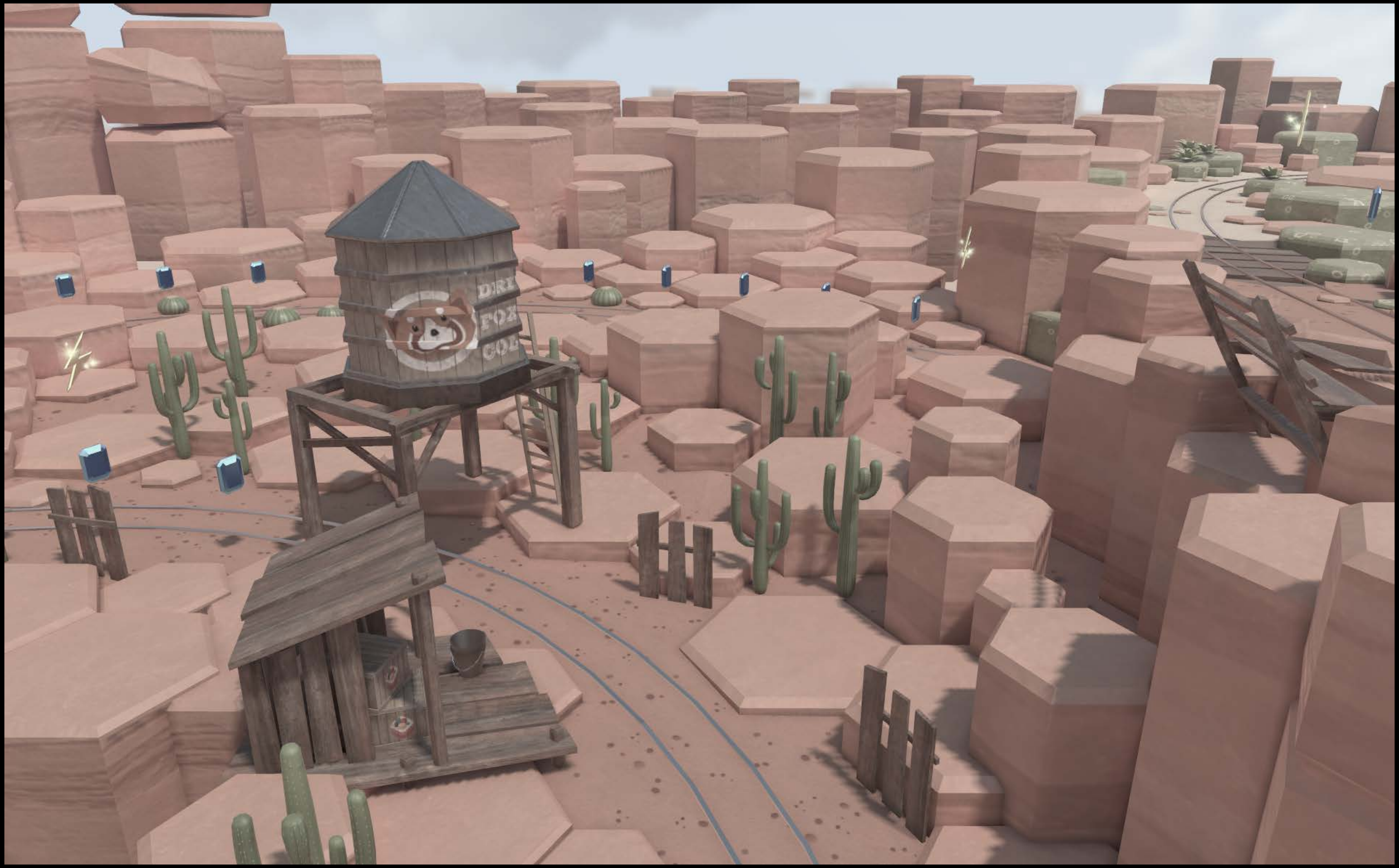
```
camera.saturation = 0.0
camera.contrast = 2.0
```

Effects

Default

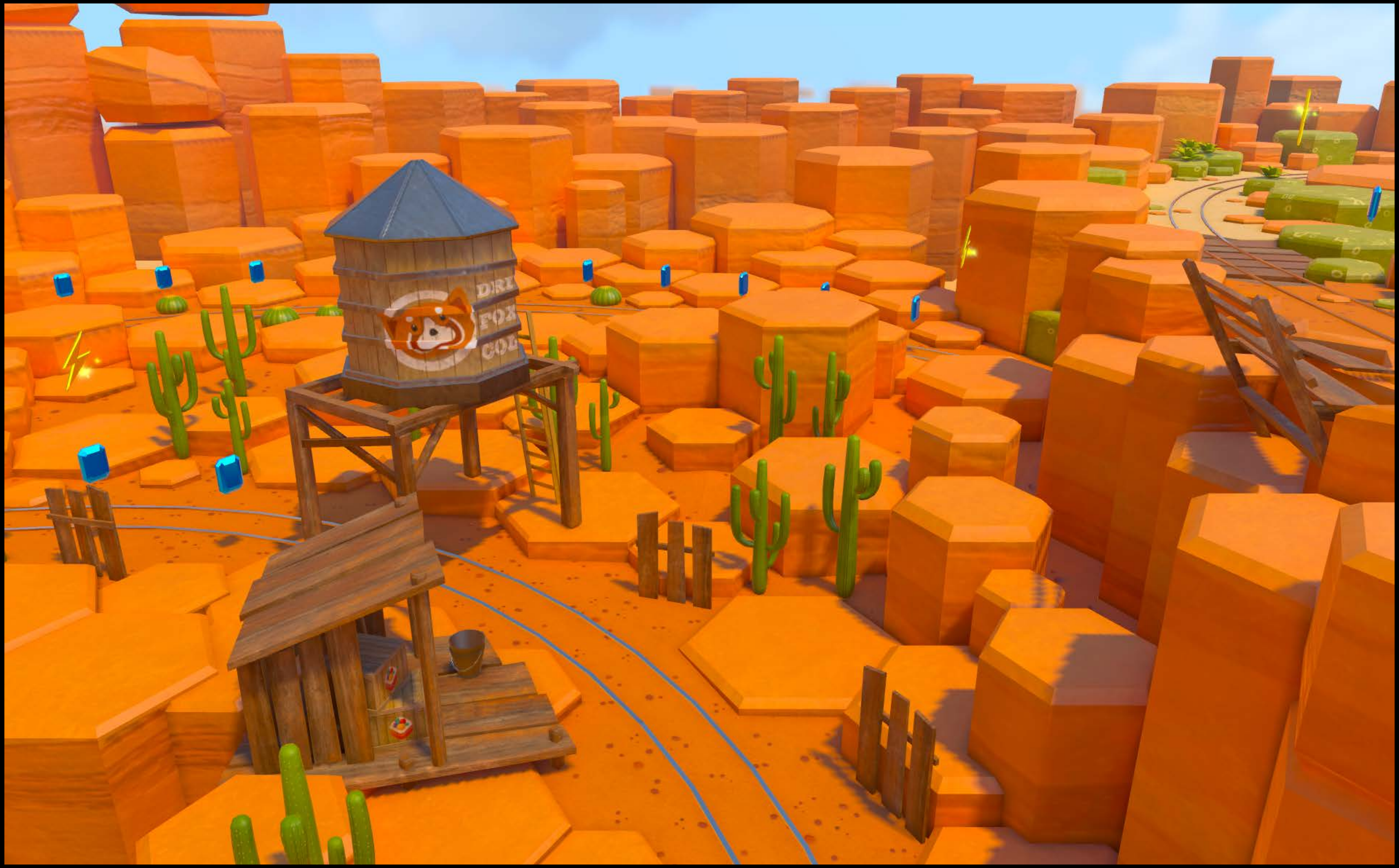
Effects

Desaturate



Effects

Saturate



Effects

Contrast

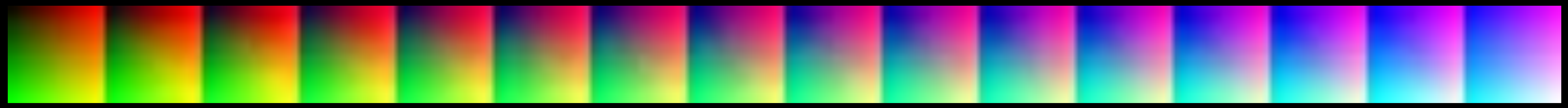
Effects

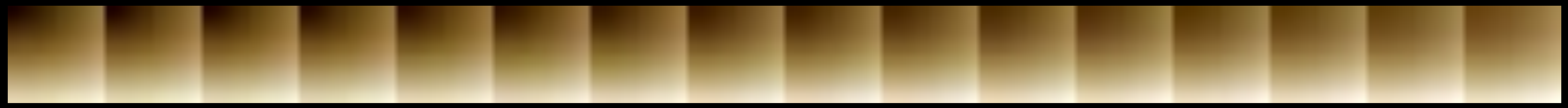
Color grading

NEW

Changes the mood of the rendering by applying a color profile

- 3D color cube
- Lookup table
- Stored as a strip of square images:





```
camera.colorGrading = "colorProfile.png"
```

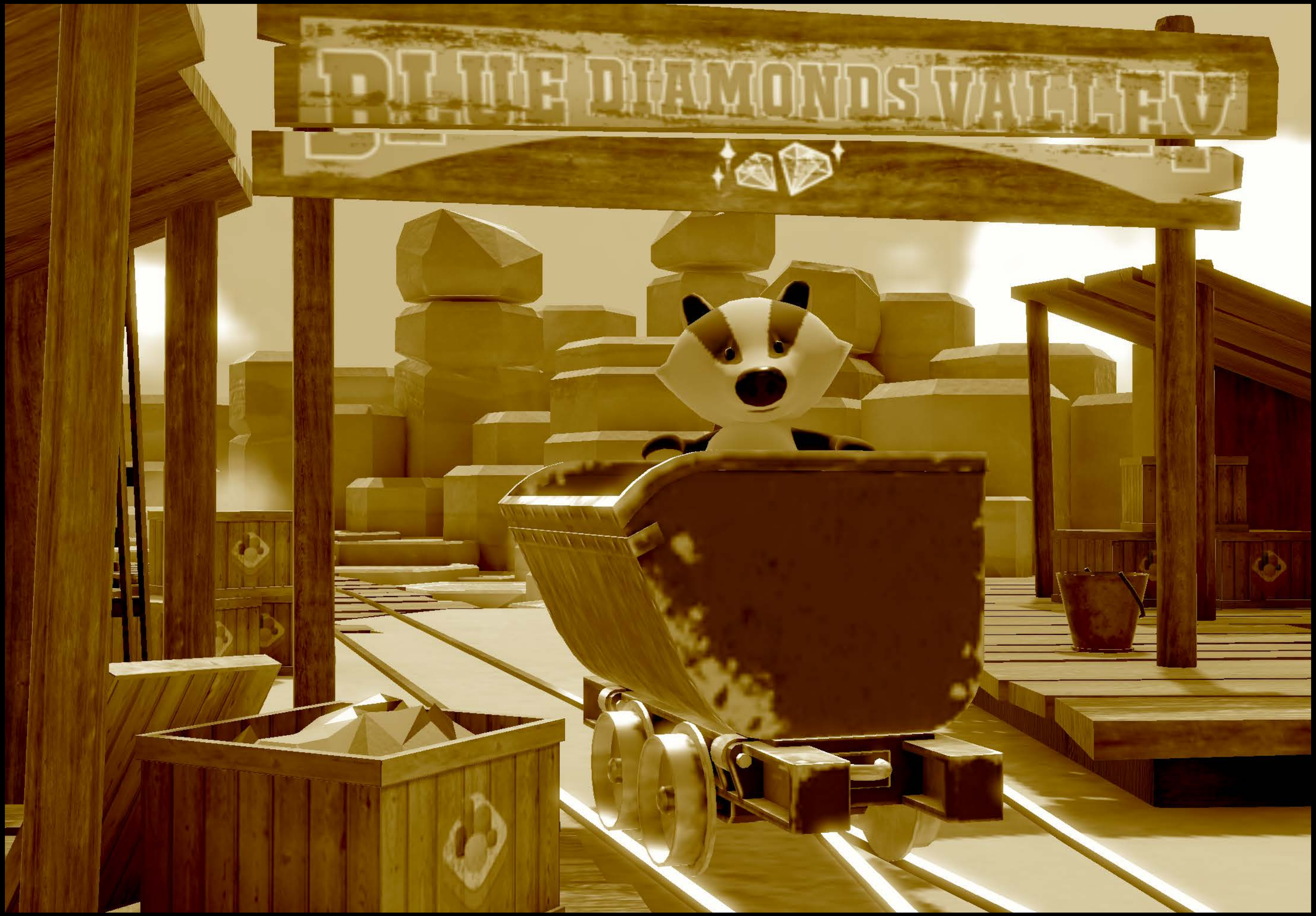
Effects

Color grading

BLUE DIAMONDS VALLEY

Effects

Color grading



HDR and Camera effects

Summary

Brand new HDR cameras and effects

- Configurable tone mapping and exposure
- Bloom
- Motion blur
- Vignetting
- Color fringe
- Saturation and contrast
- Color grading

I/O Improvements

Nick Porcino Model I/O Engineer

Primitives

Polygons

NEW

Easier to use

Automatic triangulation

Allow for much better subdivision

Opt-in when importing files

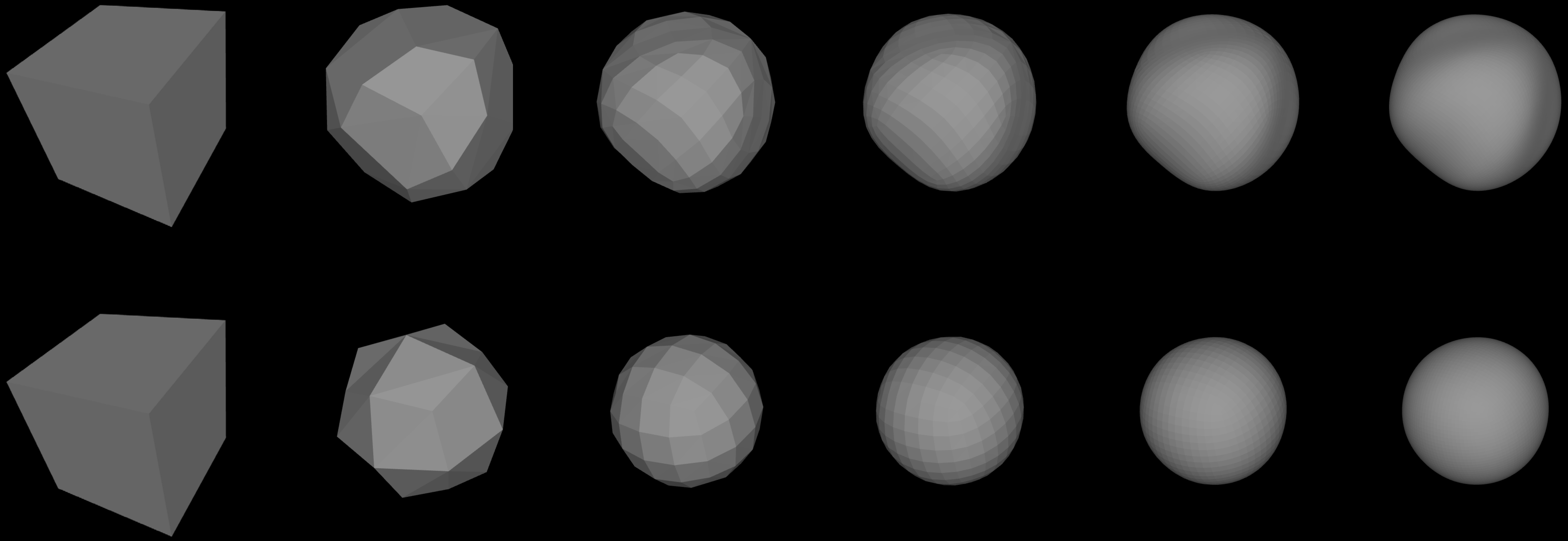
```
let loadingOptions = [.preserveOriginalTopology: true]
```

Subdivision Surfaces

OpenSubdiv 3

Faster

Better results



Triangles

Quads

Model I/O

Model I/O

3D data interchange

3D data interchange

- Between apps
- Between frameworks
- Standard file formats

Model I/O

USD

UNIVERSAL SCENE DESCRIPTION

PIXAR

Animation Studios

Model I/O

A new open standard

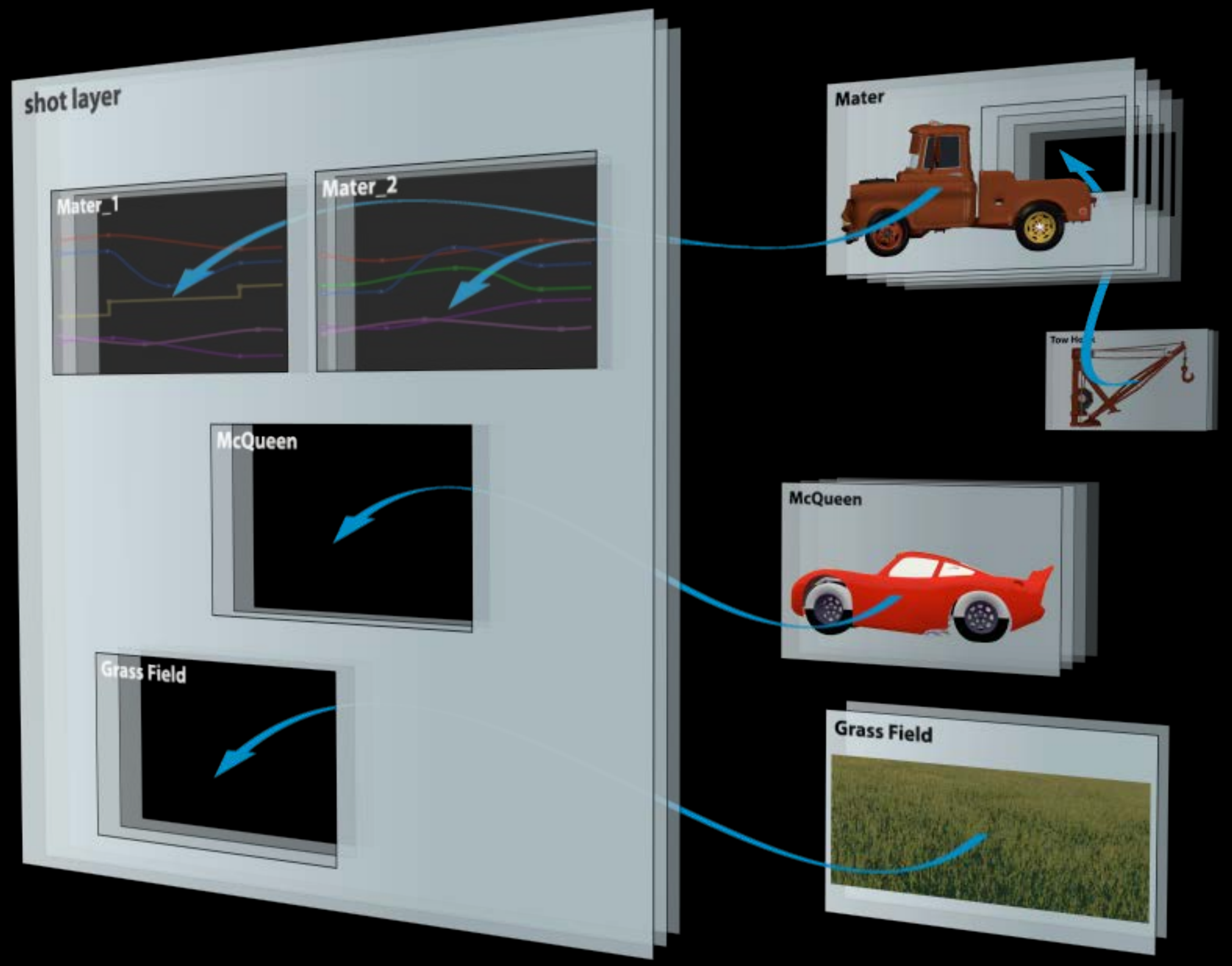
Years of practical production technologies

Data types specialized for scenes

File layering enables concurrent workflows

Universal Scene Description

Layers

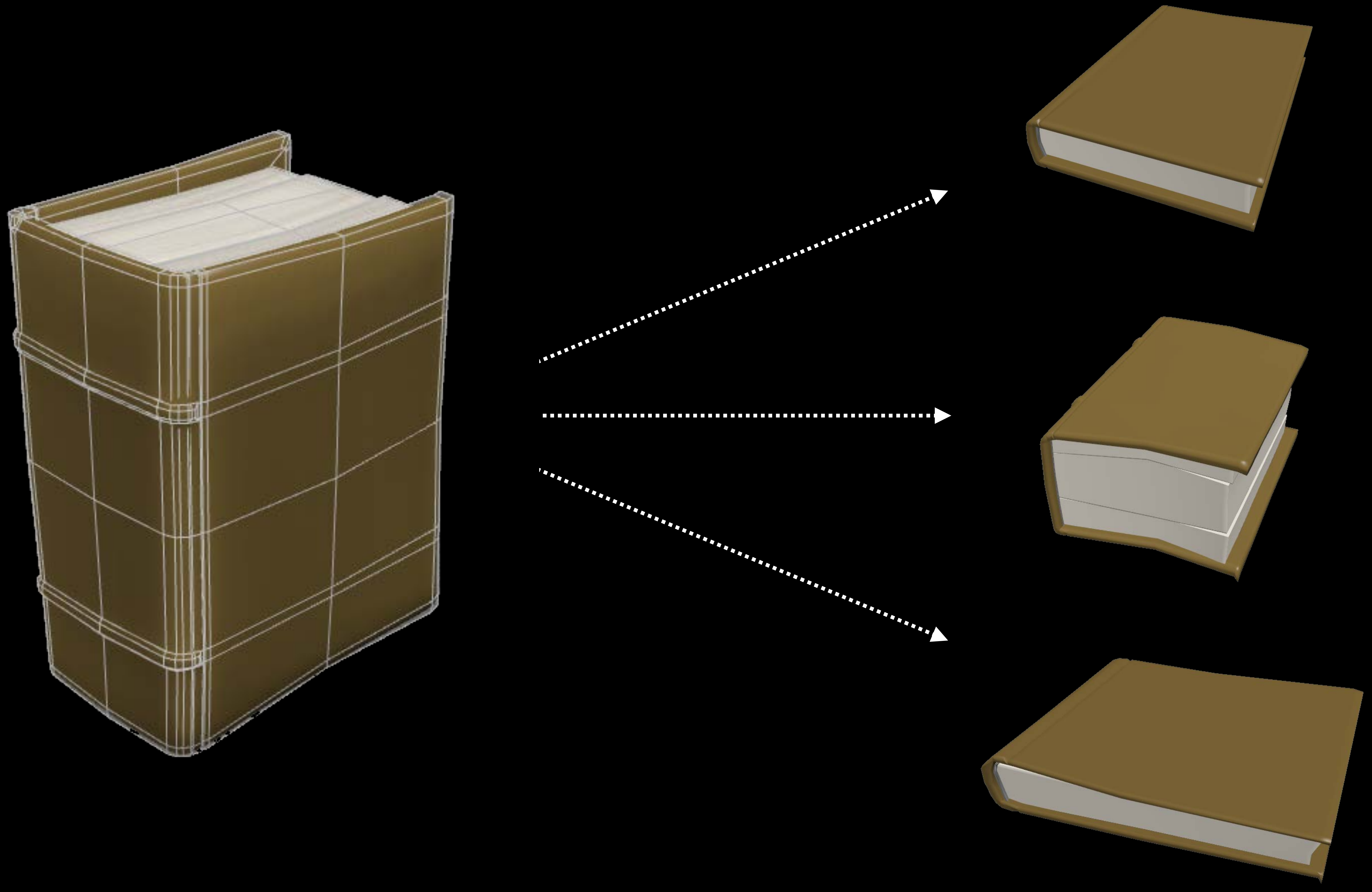


Universal Scene Description

Classes

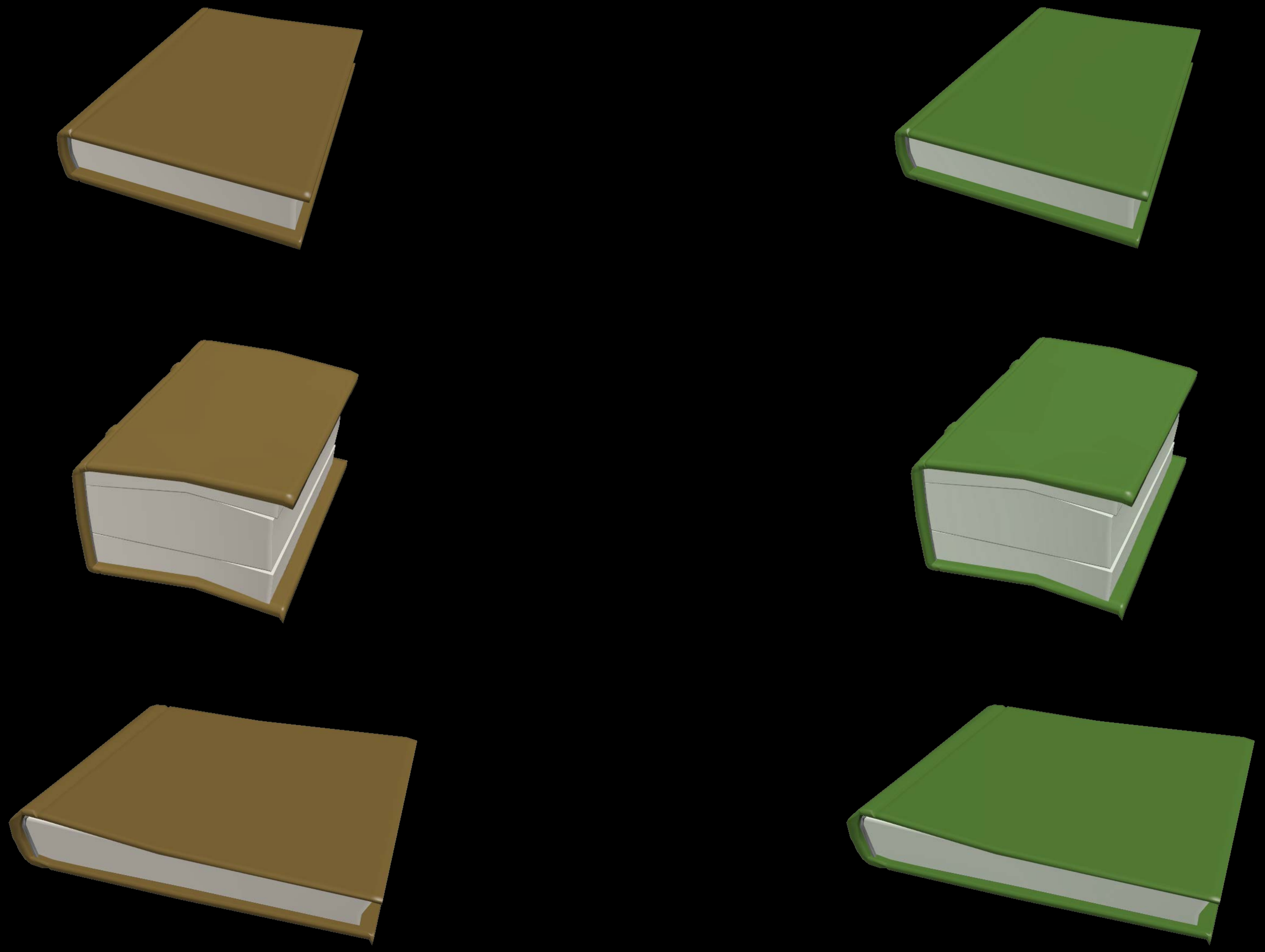
Universal Scene Description

Classes



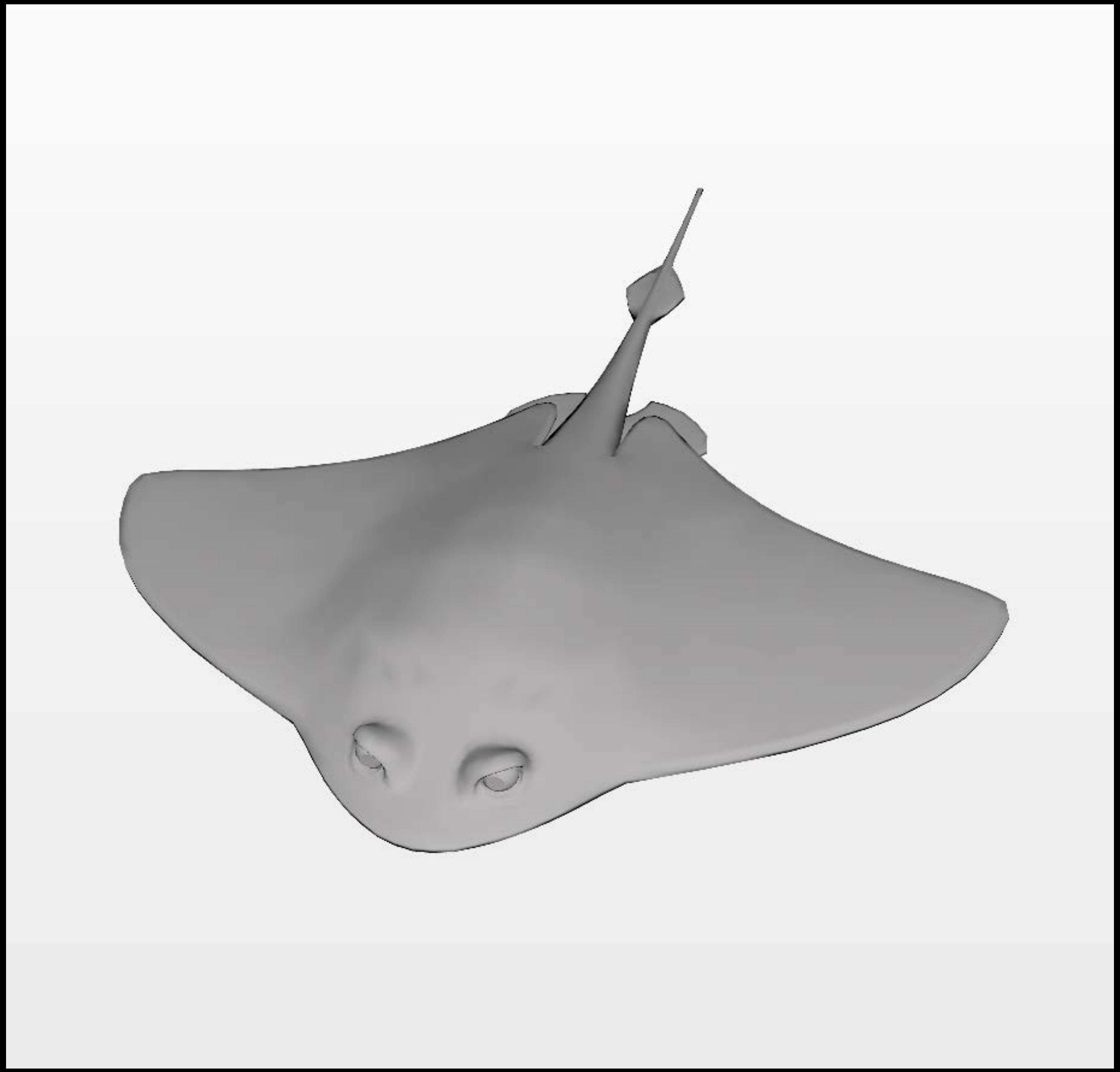
Universal Scene Description

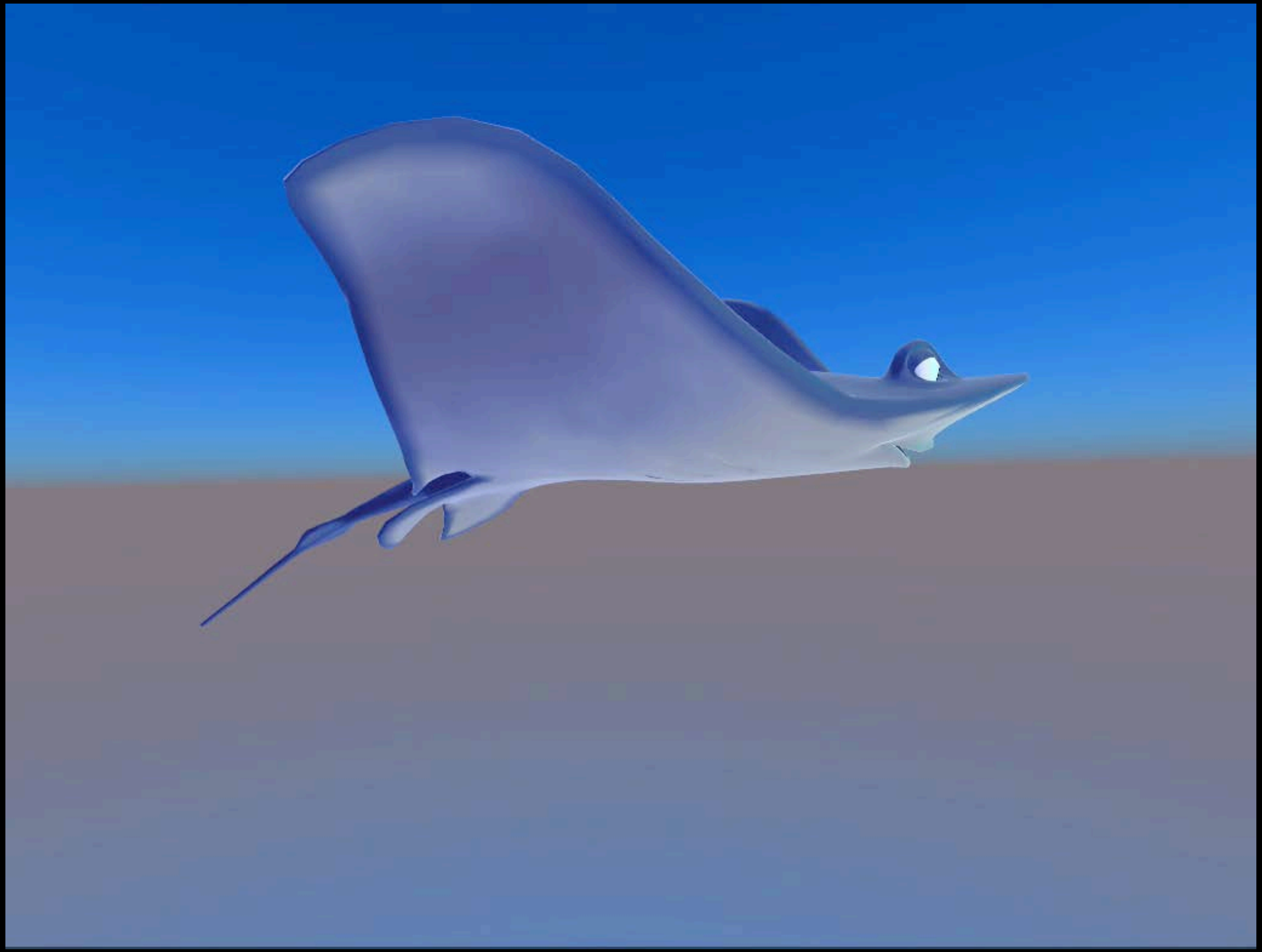
Variations



Universal Scene Description

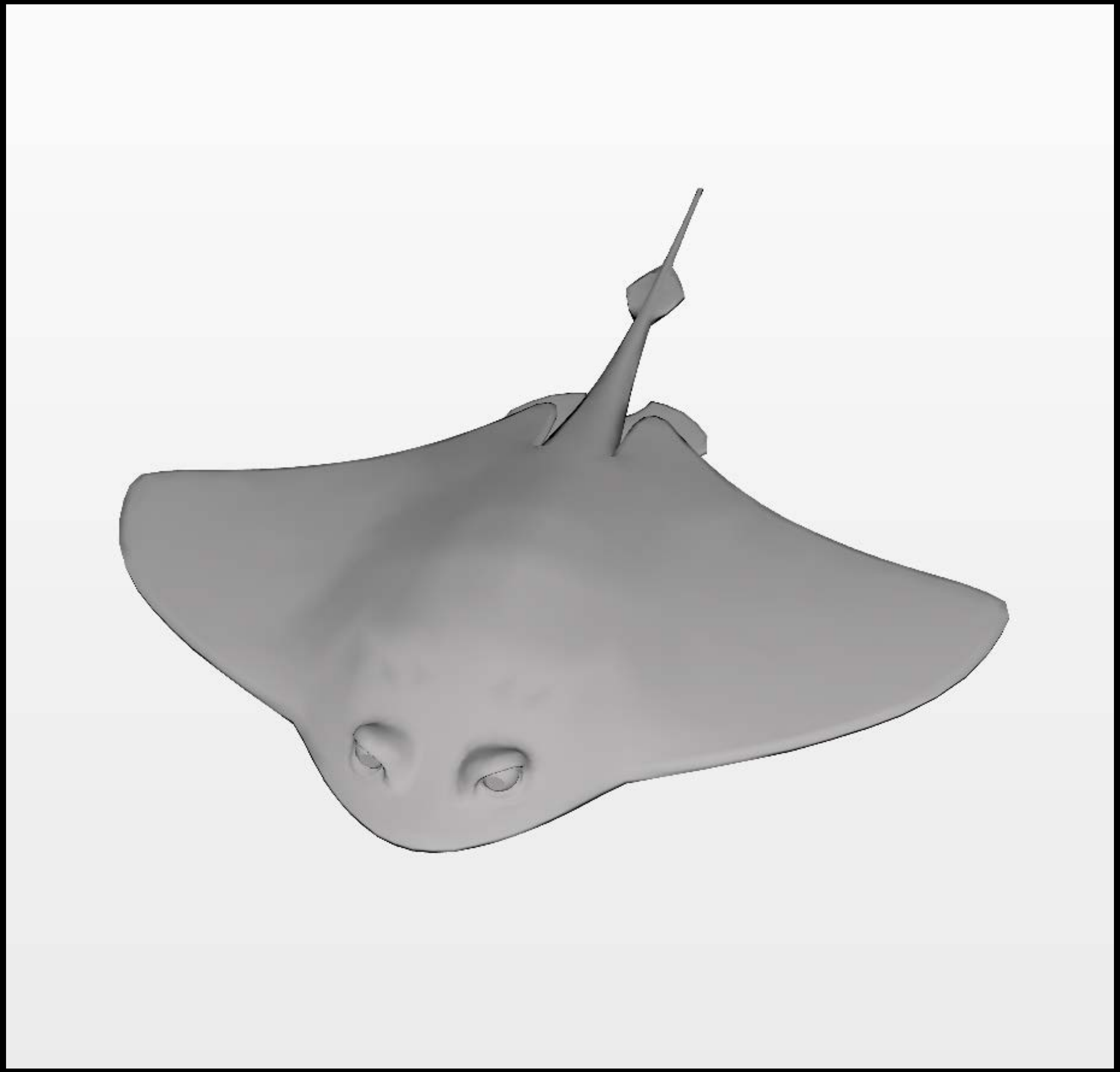
Capabilities

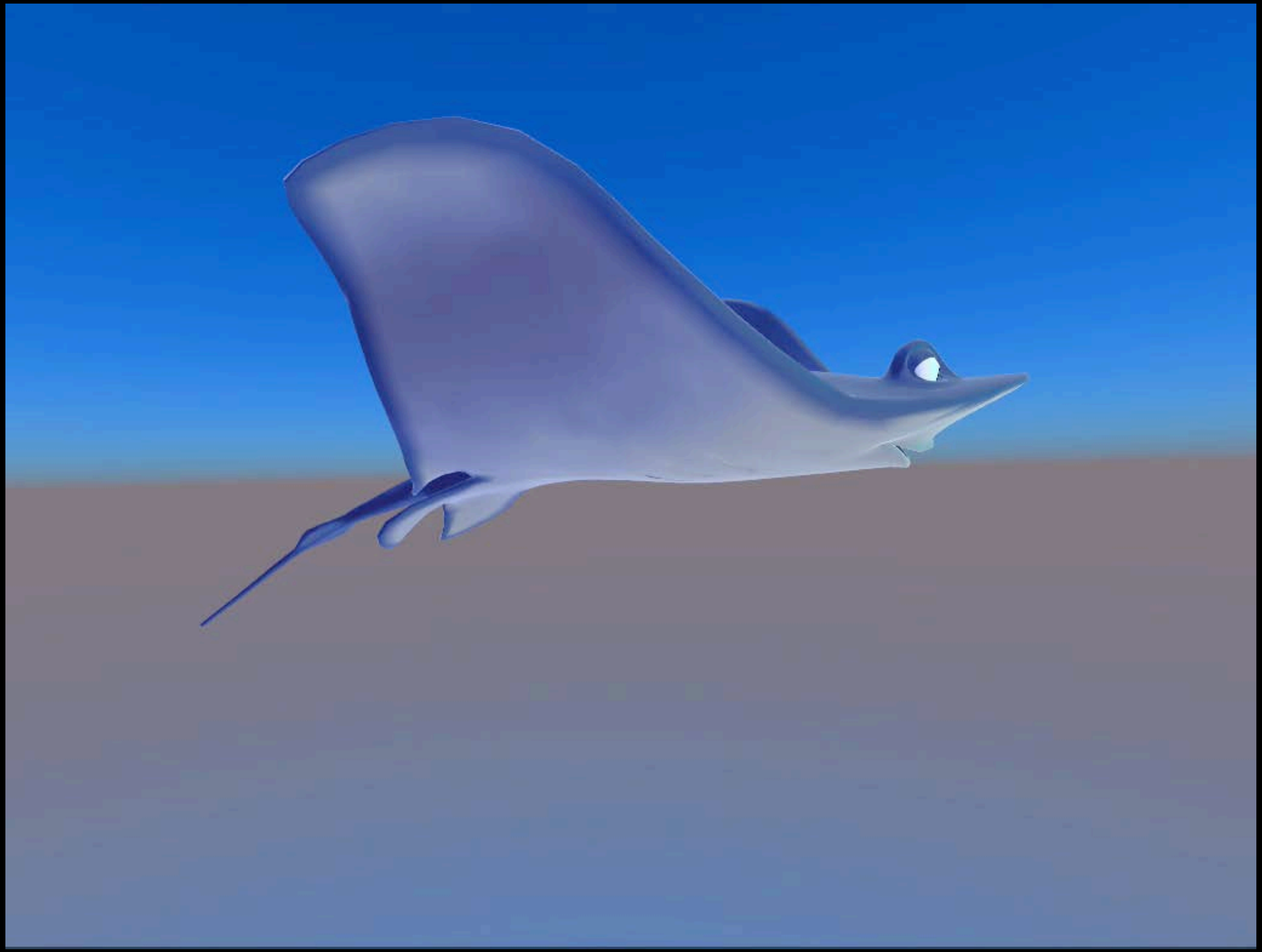




Universal Scene Description

Capabilities





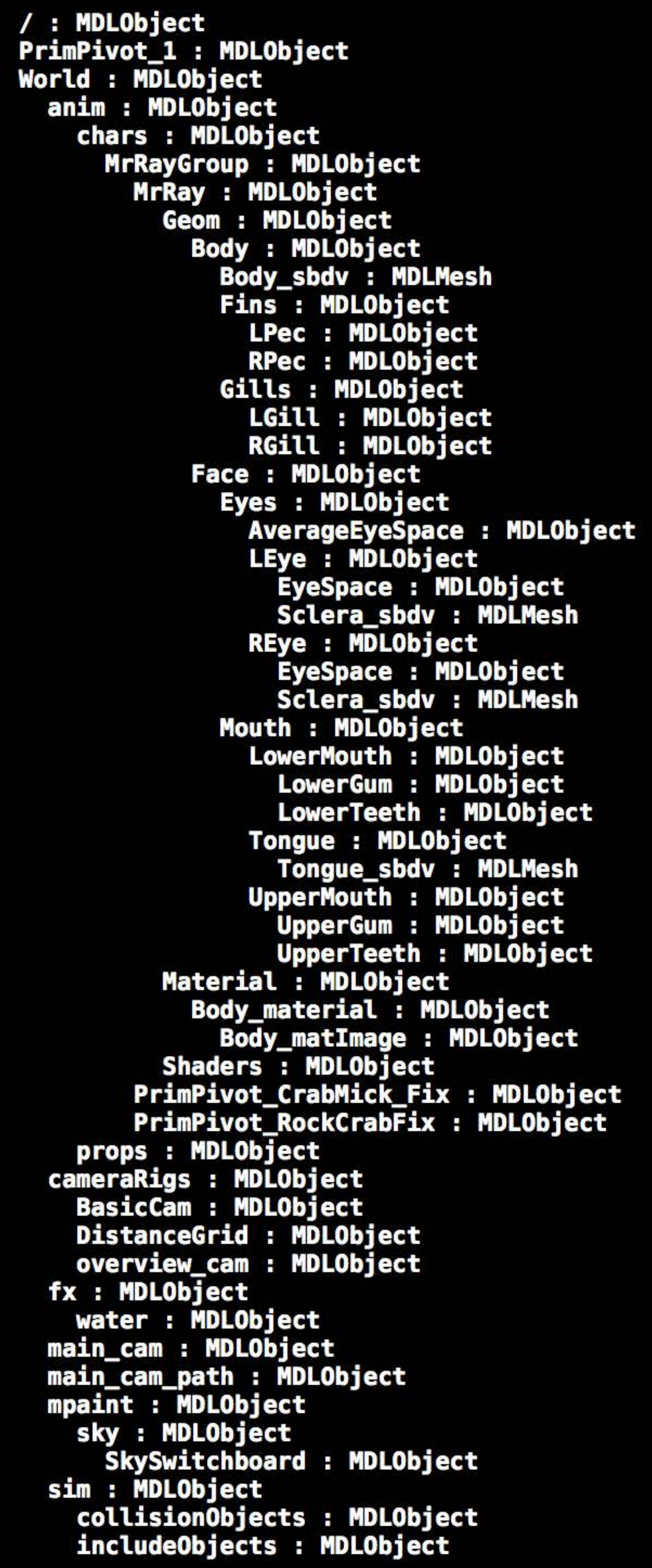
Universal Scene Description

Workflow

Integration across the system

Universal Scene Description

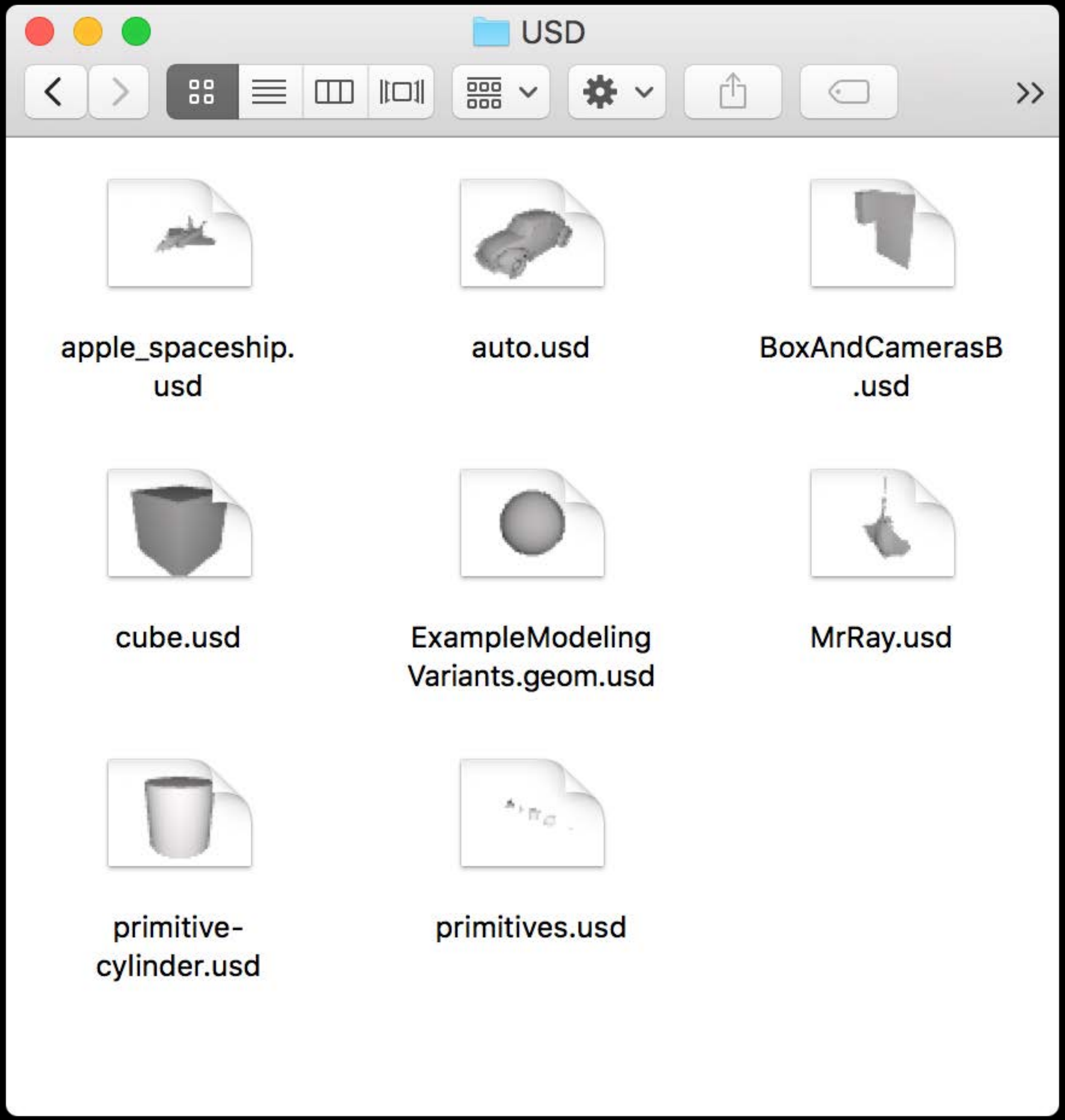
Model I/O



```
/ : MDLObject
PrimPivot_1 : MDLObject
World : MDLObject
  anim : MDLObject
    chars : MDLObject
      MrRayGroup : MDLObject
        MrRay : MDLObject
          Geom : MDLObject
            Body : MDLObject
              Body_sbdv : MDLMesh
              Fins : MDLObject
                LPec : MDLObject
                RPec : MDLObject
              Gills : MDLObject
                LGill : MDLObject
                RGill : MDLObject
            Face : MDLObject
              Eyes : MDLObject
                AverageEyeSpace : MDLObject
                LEye : MDLObject
                  EyeSpace : MDLObject
                  Sclera_sbdv : MDLMesh
                REye : MDLObject
                  EyeSpace : MDLObject
                  Sclera_sbdv : MDLMesh
              Mouth : MDLObject
                LowerMouth : MDLObject
                  LowerGum : MDLObject
                  LowerTeeth : MDLObject
                Tongue : MDLObject
                  Tongue_sbdv : MDLMesh
                UpperMouth : MDLObject
                  UpperGum : MDLObject
                  UpperTeeth : MDLObject
          Material : MDLObject
            Body_material : MDLObject
              Body_matImage : MDLObject
          Shaders : MDLObject
        PrimPivot_CrabMick_Fix : MDLObject
        PrimPivot_RockCrabFix : MDLObject
    props : MDLObject
  cameraRigs : MDLObject
    BasicCam : MDLObject
    DistanceGrid : MDLObject
    overview_cam : MDLObject
  fx : MDLObject
    water : MDLObject
  main_cam : MDLObject
  main_cam_path : MDLObject
  mpaint : MDLObject
    sky : MDLObject
      SkySwitchboard : MDLObject
  sim : MDLObject
    collisionObjects : MDLObject
    includeObjects : MDLObject
```

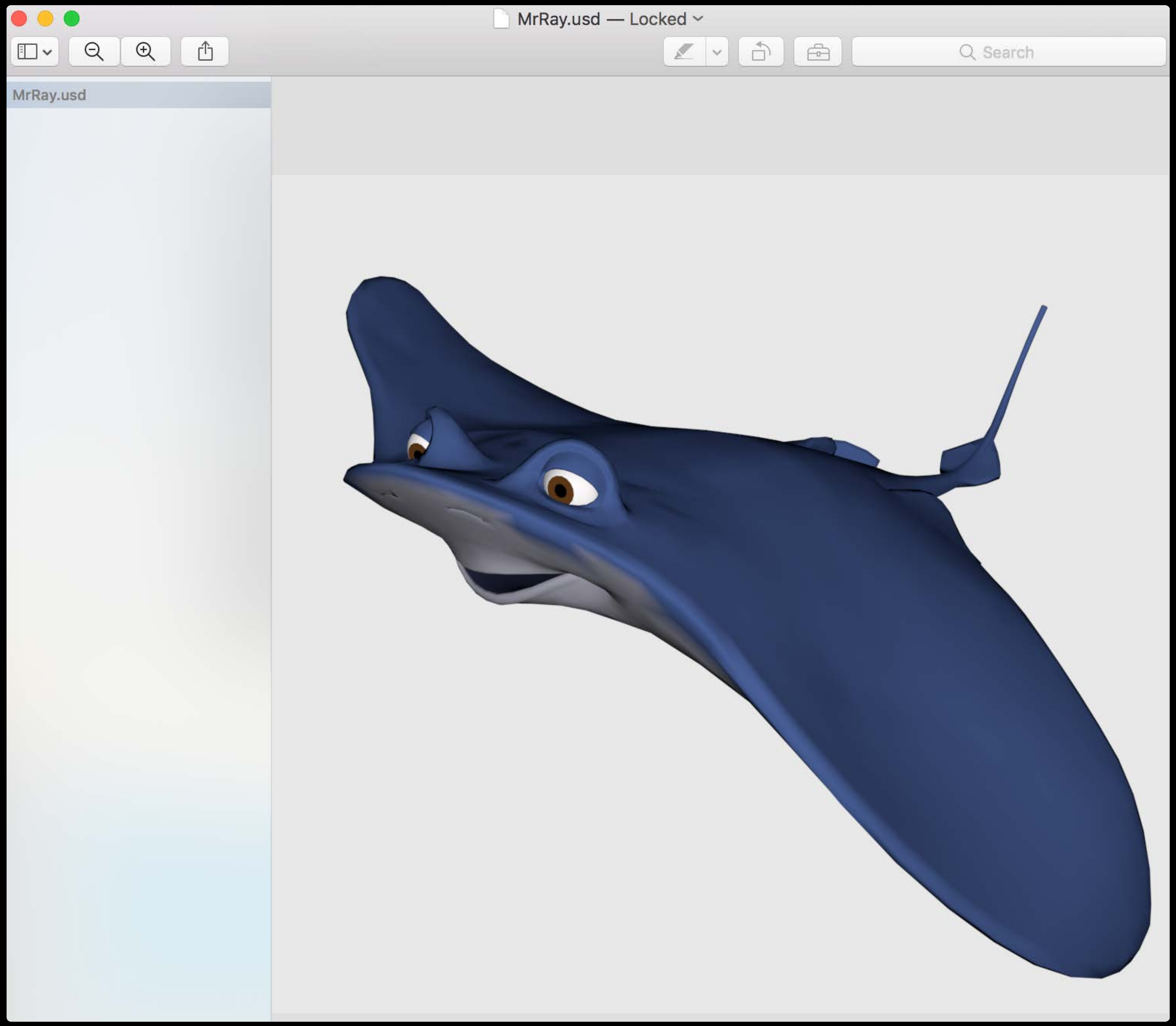
Universal Scene Description

Finder



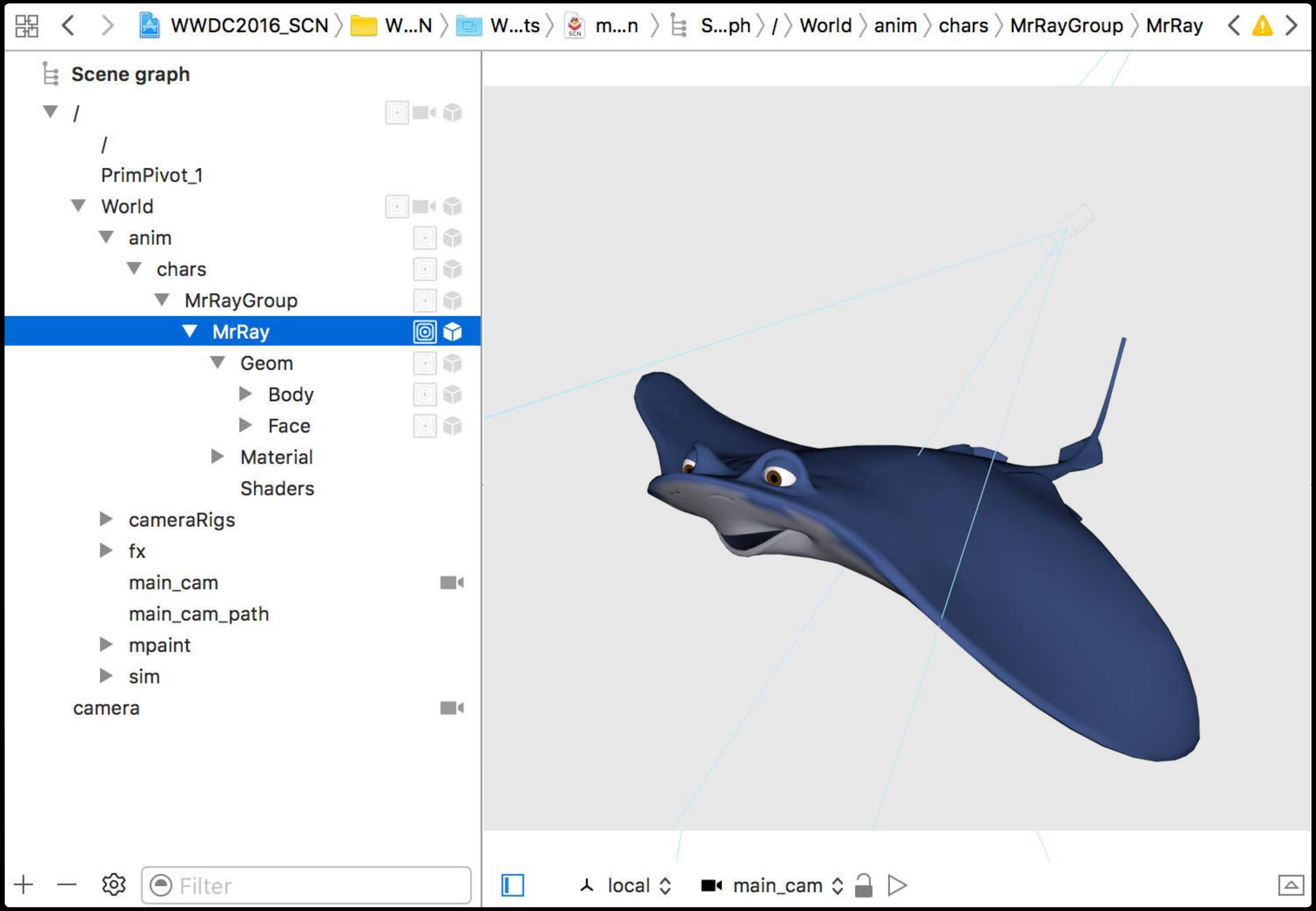
Universal Scene Description

Preview



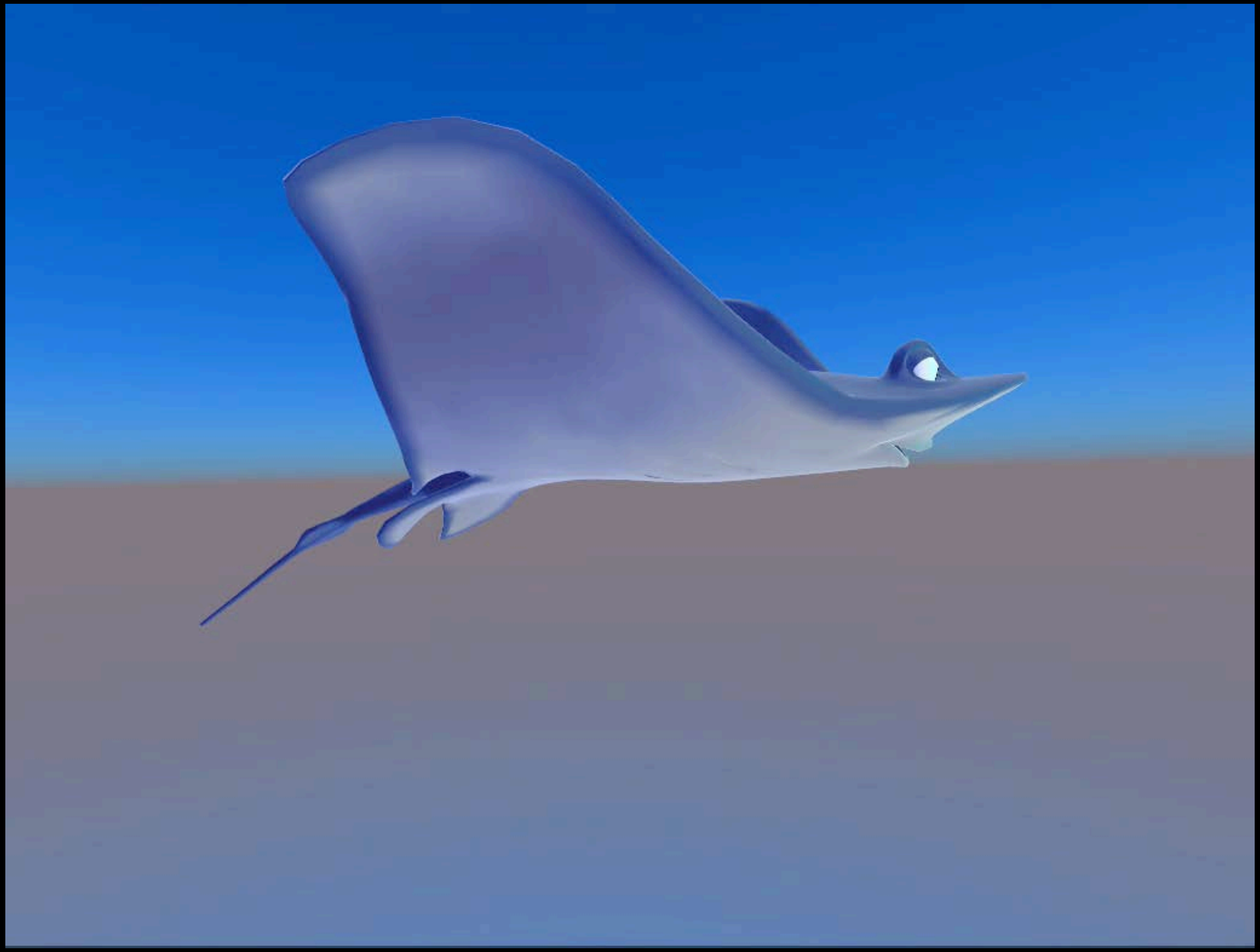
Universal Scene Description

Xcode



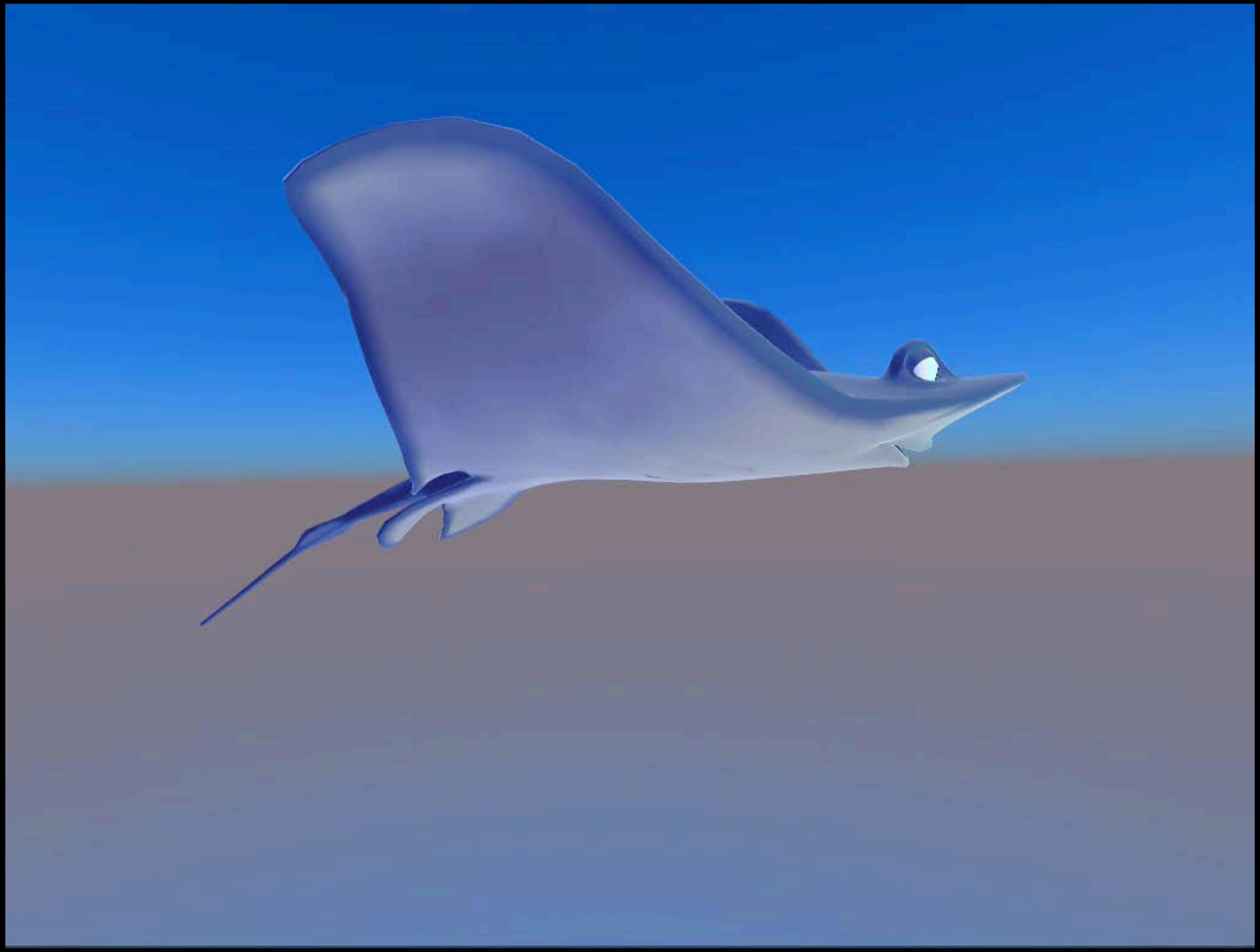
Universal Scene Description

SceneKit



Universal Scene Description

SceneKit



Universal Scene Description

Workflow

Plugins

Seamless movement of 3D data

- Between people
- Content creation programs
- Apps

Plugins and open source information are available from http://openusd.org

Summary

SceneKit available on all platforms

Physically based rendering

HDR camera and effects

Support for USD files

More Information

https://developer.apple.com/wwdc16/609

Related Sessions

Visual Debugging with Xcode	Presidio	Wednesday 4:00PM
Working with Wide Color	Mission	Thursday 1:40PM
Game Technologies for Apple Watch	Mission	Friday 3:00PM

Labs

SceneKit Lab	Graphics, Games, and Media Lab A	Thursday 3:00PM
Model I/O Lab	Graphics, Games, and Media Lab B	Thursday 3:00PM
watchOS Graphics and Games Lab	Graphics, Games, and Media Lab B	Friday 4:00PM

WWDC16