Working with USD

Denis Kovacs, GPU Software Engineer
Chloe Moore, GPU Software Engineer
iOS Integration

- Messages
- Mail
- Notes
- News
- Safari
- Files
- 3rd Party Apps
What is USD?
Workflows
USD essentials
USD highlights
What is USD?

Workflows

USD essentials

USD highlights
Universal Scene Description

Developed by Pixar
- 3D file format
- C++ library with Python bindings
  - Read/write
  - Composition engine
  - And much more...

Focus on speed, scalability, collaboration
USD File Types

- USDA: Plain text
- USDC: Binary
- USD: Plain text or binary
USD File Types

- USDA: Plain text
- USDC: Binary
- USD: Plain text or binary
- USDZ: Archive
What Is usdz?

Distribution format for USD
Compact single file
Optimized for sharing
Basis of AR Quick Look
Supported on iOS, macOS, and tvOS
Anatomy of usdz

Uncompressed archive
• 64-byte aligned files (memory-mapping)

Scene description file(s)
• .usda, .usd, .usdc, .usdz

Textures
• .png or .jpeg
3D Formats

3D Model

.obj
3D Formats

- 3D Model
  - .obj
- Scene Graph
  - .gltf, .fbx, .abc
3D Formats

- 3D Model
  - .obj
- Scene Graph
- Live Composition, Collaboration
- Scalable to complex scenes
- .gltf, .fbx, .abc
- .usd
What is USD?

Workflows

USD essentials

USD highlights
Converting 3D Content

- .obj
- .gltf
- .fbx
- .abc

Convert

.usdz

Deploy

Your App, AR Quick Look
usdzconvert

New command-line tool

.obj, .fbx, .gltf, .abc, etc.

Performs asset validation

Python based

Platform independent
% usdzconvert esprit.glTF
usdzconvert: converting file: esprit.glTF
usdzconvert: converted usdz file: esprit.usdz
usdARKitChecker: [Pass] esprit.usdz

% usdzconvert -h
[...]

% usdzconvert esprit.obj -diffuseColor d.png -normal n.png -metallic m.png -roughness r.png
[...]
% usdzconvert esprit.gltf
usdzconvert: converting file: esprit.gltf
usdzconvert: converted usdz file: esprit.usdz
usdARKitChecker: [Pass] esprit.usdz

% usdzconvert -h
[...]

% usdzconvert esprit.obj -diffuseColor d.png -normal n.png -metallic m.png -roughness r.png
[...]
% usdzconvert esprit.gltf
usdzconvert: converting file: esprit.gltf
usdzconvert: converted usdz file: esprit.usdz
usdARKitChecker: [Pass] esprit.usdz

% usdzconvert -h
[...]

% usdzconvert esprit.obj -diffuseColor d.png -normal n.png -metallic m.png -roughness r.png
[...]
% usdzconvert esprit.gltf
usdzconvert: converting file: esprit.gltf
usdzconvert: converted usdz file: esprit.usdz
usdARKitChecker: [Pass] esprit.usdz

% usdzconvert -h
[...]

% usdzconvert esprit.obj -diffuseColor d.png -normal n.png -metallic m.png -roughness r.png
[...]
Augmented Reality

Bethesda Gear

Nomatic

FURNI

usdz Tools

Download essential Python-based tools for generating, validating, and inspecting usdz files. Also includes a converter that creates usdz from other 3D file formats along with Pixar’s USD library and sample scripts.

Download usdz tools
usdz Tools

Download essential Python-based tools for generating, validating, and inspecting usdz files. Also includes a converter that creates usdz from other 3D file formats along with Pixar’s USD library and sample scripts.

Download usdz tools 📈
% usdcat model.usdz
[...]

% usdtree model.usdz
/
 `--Parent [def Xform] (kind = component)
    |--CubeChild [def Cube]
    |--SphereChild [def Sphere]
 `--Materials [def Scope]
     `--Glass [def Material]

% usdchecker model.usdz --arkit
Success!

% fixOpacity model.usdz
[...]
% usdcat model.usdz
[...]

% usdtree model.usdz
/
  `--Parent [def Xform] (kind = component)
     |   `--CubeChild [def Cube]
     |   `--SphereChild [def Sphere]
     `--Materials [def Scope]
         `--Glass [def Material]

% usdchecker model.usdz --arkit
Success!

% fixOpacity model.usdz
[...]
% usdcat model.usdz

% usdtree model.usdz
/
  `--Parent [def Xform] (kind = component)
    |--CubeChild [def Cube]
    |--SphereChild [def Sphere]
  `--Materials [def Scope]
    `--Glass [def Material]

% usdchecker model.usdz --arke
Success!

% fixOpacity model.usdz
[...]
% usdcat model.usdz

% usdtree model.usdz
/
  `--Parent [def Xform] (kind = component)
      |--CubeChild [def Cube]
      |--SphereChild [def Sphere]
  `--Materials [def Scope]
      `--Glass [def Material]

% usdchecker model.usdz --arkit
Success!

% fixOpacity model.usdz
[...]
% usdcat model.usdz

% usdtree model.usdz

/  
  `--Parent [def Xform] (kind = component)
        |--CubeChild [def Cube]
        |--SphereChild [def Sphere]
  `--Materials [def Scope]
        `--Glass [def Material]

% usdchecker model.usdz --arkit
Success!

% fixOpacity model.usdz

[...]
Demo

Chloe Moore, GPU Software Engineer
Convert  Create
Creating 3D Content

Convert

Deploy

Your App, AR Quick Look
Creating 3D Content

Content Creation Tool → Export 
  .obj  .gltf  .fbx  .abc → Convert 
  .usdz → Deploy 
  Your App, AR Quick Look
Creating 3D Content

Content Creation Tool → Export → .obj, .gltf, .fbx, .abc → Convert → .usdz → Deploy → Your App, AR Quick Look
Creating 3D Content

- Content Creation Tool
- Export .usd
- Archive .usdz
- Deploy Your App, AR Quick Look
Creating 3D Content

Content Creation Tool → Export → .usd → Archive → .usdz → Deploy → Your App, AR Quick Look
Creating 3D Content

Content Creation Tool → Export → .usd → Archive → .usdz → Deploy → Your App, AR Quick Look
Creating 3D Content

Content Creation Tool → Export → .usd

→ Archive → .usdz

→ Deploy → Your App, AR Quick Look
Creating 3D Content

Custom Python Pipeline → .usd → Archive → .usdz → Deploy → Your App, AR Quick Look
Exporting usdz from SceneKit

Create or load SCNScene
Export .usdz

```swift
let scene = SCNScene(...)
...
scene.write(to: baseURL.appendingPathComponent("file.usdz"), delegate: nil)
```
USD Features
USD Features

Subdivision Surface Instance Override

UsdPreviewSurfaceTime-Varying

Geometry SubsetVariant

Shader Graph Assembly ConnectionInterpolation

Skeletal AnimationMaterial Library Custom SchemaPayload

UsdLux SubLayerInherits Layer Offset Namespace Prim

SpecializesAPI  Schema AssetInfoHydra Purpose

Transform StackCustom Data Collection Default ValueSession Layer

Vertex AnimationKindList EditingPseudoRoot Stage

ProxyTimeCode

Asset ResolverRelationshipValue Clip FlattenTimeSample

Indexed Mesh Attribute

Layer Offset Namespace

Layer StackComposition Engine

Composition Engine

Instance Override

Value Resolution

Transform Stack

Custom Schema

Metadata

Stage

Asset Resolver
USD Features

- Subdivision Surface
- Subdivision Surface
- Instance Override
- UsdPreviewSurface
- Time-Varying Properties
- Geometry Subset
- Variant
- Shader Graph
- Assembly
- Interpolation
- Skeletal Animation
- Material Library
- Custom Schema
- Payload
- UsdLux
- SubLayer
- Layer Offset
- Namespace
- Inherits
- Layer Stack
- Edit Target
- PointInstancer
- Composition Engine
- Proxy
- Session Layer
- Value Resolution
- Transform Stack
- Custom Data
- Collection
- Session Layer
- Default Value
- Mesh
- Scene Graph
- Time-Varying Topology
- Instancing
- Mesh
- Reference
- Subtree Skinning
- Visibility
- Value Clip
- List Editing
- Kind
- TimeSample
- Flatten
- Asset Resolver
- Value Clip
- Crane File
- Edit Target
- Layer Stack
- Composition Engine
What is USD?

Workflows

USD essentials

USD highlights
USD Essentials

Basic File Structure

Scene Graph

Mesh Data

Materials
File Structure

#usda 1.0
def Xform "simpleMesh"
{
    def Mesh "cube"
    {
        
    }
}

# usda 1.0

def Xform "simpleMesh"
{
    def Mesh "cube"
    {
        point3f[] points = [(0,0,0), (1,0,0), (1,1,0), (0,1,0)]
        normal3f[] normals = [(0,0,1)]
    
    ...
    
    }
}
File Structure — Metadata

```python
#usda 1.0

upAxis = "Y"

def Xform "simpleMesh"
{
    def Mesh "cube"
    {
        point3f[] points = [(0,0,0), (1,0,0), (1,1,0), (0,1,0)]
        normal3f[] normals = [(0,0,1)]
    }
    ...
}
}```
File Structure — Metadata

```python
#usda 1.0
(
    upAxis = "Y"
)

def Xform "simpleMesh"
    kind = "component"
)
{
    def Mesh "cube"
    {
        point3f[] points = [(0,0,0), (1,0,0), (1,1,0), (0,1,0)]
        normal3f[] normals = [(0,0,1)]

        ...
    }
}
# File Structure — Metadata

```python
#usda 1.0
(
    upAxis = "Y"
)
def Xform "simpleMesh" (  
    kind = "component"
)
{
    def Mesh "cube"
    {
        point3f[] points = [(0,0,0), (1,0,0), (1,1,0), (0,1,0)]
        normal3f[] normals = [(0,0,1)] (  
            interpolation = "uniform"
        )
        ...
    }
}
```
Object Access Through Paths

```usda
#usda 1.0
(
    upAxis = "Y"
)

def Xform "simpleMesh" ( kind = "component"
)
{

def Mesh "cube" /simpleMesh/cube
{
    point3f[] points = [(0,0,0), (1,0,0), (1,1,0), (0,1,0)]
    normal3f[] normals = [(0,0,1)] ( interpolation = "uniform"
    )
    ...
}
}
Object Access Through Paths

```python
#usda 1.0

upAxis = "Y"

def Xform "simpleMesh" (kind = "component")
{
    def Mesh "cube"
    {
        point3f[] points = [(0,0,0), (1,0,0), (1,1,0), (0,1,0)]
        normal3f[] normals = [(0,0,1)] (interpolation = "uniform"
                      /simpleMesh/cube.normals)
        ...
    }
}
```
USD Essentials

Basic File Structure

Scene Graph

Mesh Data

Materials
Scene graph defines object hierarchy

Transforms on a parent also affect its children
Scene graph defines object hierarchy

Transforms on a parent also affect its children
Scene graph defines object hierarchy

Transforms on a parent also affect its children
Scene graph defines object hierarchy
Transforms on a parent also affect its children
Scene Graph

Scene graph defines object hierarchy

Transforms on a parent also affect its children
Scene Graph

def Xform "Parent"
{
  def Cube "Cube"
  {
  }
  def Sphere "Sphere"
  {
  }
  def Scope "Materials"
  {
    def Material "Glass"
    {
    }
  }
}

Parent
  Cube
  Sphere
USD Essentials

Basic File Structure

Scene Graph

Mesh Data

Materials
Mesh Data

Mesh attributes

Mesh connectivity

Positions

Normals

Texture Coordinates
Mesh Data

Mesh attributes

Mesh connectivity
def Mesh "Tetrahedron"
{
    point3f[] points = [(-1,-1,1), (-1,1,-1), (1,-1,-1), (1,1,1)]
    int[] faceVertexCounts = [3, 3, 3, 3]
    int[] faceVertexIndices = [1,2,0, 3,2,1, 3,0,2, 3,1,0]

    normal3f[] normals = [(-0.58,-0.58,-0.58), (0.58,0.58,-0.58),
                          (0.58,-0.58,0.58), (-0.58,0.58,0.58)] (interpolation = "uniform")

    float2[] primvars:st = [(0.75,0.43), (0.5,0.87), (0.25,0.43),
                             (1,0.87), (0,0.87), (0.5,0)] (interpolation = "faceVarying")

    int[] primvars:st:indices = [0,1,2, 3,1,0, 4,2,1, 5,0,2]

    uniform token subdivisionScheme = "none"
}
def Mesh "Tetrahedron"
{
    point3f[] points = [(-1,-1,1), (-1,1,-1), (1,-1,-1), (1,1,1)]
    int[] faceVertexCounts = [3, 3, 3, 3]
    int[] faceVertexIndices = [1,2,0, 3,2,1, 3,0,2, 3,1,0]

    normal3f[] normals = [(-0.58,-0.58,-0.58), (0.58,0.58,-0.58),
                            (0.58,-0.58,0.58), (-0.58,0.58,0.58)]
                            (interpolation = "uniform")

    float2[] primvars:st = [(0.75,0.43),(0.5,0.87),(0.25,0.43),
                            (1,0.87),(0,0.87),(0.5,0)]
                            (interpolation = "faceVarying")

    int[] primvars:st:indices = [0,1,2, 3,1,0, 4,2,1, 5,0,2]

    uniform token subdivisionScheme = "none"
}
def Mesh "Tetrahedron"
{
    point3f[] points = [(-1,-1,1), (-1,1,-1), (1,-1,-1), (1,1,1)]

    int[] faceVertexCounts = [3, 3, 3, 3]
    int[] faceVertexIndices = [1,2,0, 3,2,1, 3,0,2, 3,1,0]

    normal3f[] normals = [(-0.58,-0.58,-0.58), (0.58,0.58,-0.58),
                         (0.58,-0.58,0.58), (-0.58,0.58,0.58)]
        (interpolation = "uniform"
    )

    float2[] primvars:st = [(0.75,0.43),(0.5,0.87),(0.25,0.43),
                            (1,0.87),(0,0.87),(0.5,0)]
        (interpolation = "faceVarying"
    )

    int[] primvars:st:indices = [0,1,2, 3,1,0, 4,2,1, 5,0,2]

    uniform token subdivisionScheme = "none"
}

def Mesh "Tetrahedron"
{
    point3f[] points = [(-1,-1,1), (-1,1,-1), (1,-1,-1), (1,1,1)]
    int[] faceVertexCounts = [3, 3, 3, 3]
    int[] faceVertexIndices = [1,2,0, 3,2,1, 3,0,2, 3,1,0]

    normal3f[] normals = [(-0.58,-0.58,-0.58), (0.58,0.58,-0.58),
                         (0.58,-0.58,0.58), (-0.58,0.58,0.58)] (interpolation = "uniform")

    float2[] primvars:st = [(0.75,0.43),(0.5,0.87),(0.25,0.43),
                            (1,0.87),(0,0.87),(0.5,0)] (interpolation = "faceVarying")
}

int[] primvars:st:indices = [0,1,2, 3,1,0, 4,2,1, 5,0,2]

uniform token subdivisionScheme = "none"
def Mesh "Tetrahedron"
{
    point3f[] points = [(-1,-1,1), (-1,1,-1), (1,-1,-1), (1,1,1)]
    int[] faceVertexCounts = [3, 3, 3, 3]
    int[] faceVertexIndices = [1,2,0, 3,2,1, 3,0,2, 3,1,0]

    normal3f[] normals = [(-0.58,-0.58,-0.58), (0.58,0.58,-0.58),
                          (0.58,-0.58,0.58), (-0.58,0.58,0.58)]
                          (interpolation = "uniform")

    float2[] primvars:st = [(0.75,0.43),(0.5,0.87),(0.25,0.43),
                            (1,0.87),(0,0.87),(0.5,0)]
                            (interpolation = "faceVarying")

    int[] primvars:st:indices = [0,1,2, 3,1,0, 4,2,1, 5,0,2]

    uniform token subdivisionScheme = "none"
}
def Mesh "Tetrahedron"
{
    point3f[] points = [(-1,-1,1), (-1,1,-1), (1,-1,-1), (1,1,1)]
    int[] faceVertexCounts = [3, 3, 3, 3]
    int[] faceVertexIndices = [1,2,0, 3,2,1, 3,0,2, 3,1,0]

    normal3f[] normals = [(-0.58,-0.58,-0.58), (0.58,0.58,-0.58),
                          (0.58,-0.58,0.58), (-0.58,0.58,0.58)]
                          (interpolation = "uniform"

    float2[2] primvars:st = [(0.75,0.43),(0.5,0.87),(0.25,0.43),
                              (1,0.87),(0,0.87),(0.5,0)]
                              (interpolation = "faceVarying"

    int[] primvars:st:indices = [0,1,2, 3,1,0, 4,2,1, 5,0,2]

    uniform token subdivisionScheme = "none"
}

def Mesh "Tetrahedron"
{
    point3f[] points = [(-1,-1,1), (-1,1,-1), (1,-1,-1), (1,1,1)]
    int[] faceVertexCounts = [3, 3, 3, 3]
    int[] faceVertexIndices = [1,2,0, 3,2,1, 3,0,2, 3,1,0]

    normal3f[] normals = [(-0.58,-0.58,-0.58), (0.58,0.58,-0.58),
                          (0.58,-0.58,0.58), (-0.58,0.58,0.58)]
                          (interpolation = "uniform"
                     )

    float2[] primvars:st = [(0.75,0.43),(0.5,0.87),(0.25,0.43),
                             (1,0.87),(0,0.87),(0.5,0)]
                             (interpolation = "faceVarying" per-face per-vertex attribute
                     )
    int[] primvars:st:indices = [0,1,2, 3,1,0, 4,2,1, 5,0,2]

    uniform token subdivisionScheme = "none"
}
def Mesh "Tetrahedron"
{
    point3f[] points = [(-1,-1,1), (-1,1,-1), (1,-1,-1), (1,1,1)]
    int[] faceVertexCounts = [3, 3, 3, 3]
    int[] faceVertexIndices = [1,2,0, 3,2,1, 3,0,2, 3,1,0]

    normal3f[] normals = [(-0.58,-0.58,-0.58), (0.58,0.58,-0.58),
        (0.58,-0.58,0.58), (-0.58,0.58,0.58)]
        interpolation = "uniform"
    }

    float2[] primvars:st = [(0.75,0.43),(0.5,0.87),(0.25,0.43),
        (1,0.87),(0,0.87),(0.5,0)]
        interpolation = "faceVarying"
    }
    int[] primvars:st:indices = [0,1,2, 3,1,0, 4,2,1, 5,0,2]

    uniform token subdivisionScheme = "none"
}
USD Essentials

Basic File Structure

Scene Graph

Mesh Data

Materials
Materials

UsdPreviewSurface
  • Schema for realistic real-time rendering

Physically based material description

Supported workflows
  • Metallic-roughness
  • Specular-roughness
Material Properties
UsdPreviewSurface

Example

diffuseColor = (1,1,1)
Material Properties

UsdPreviewSurface

Example

diffuseColor = (1,1,1)
normal = normals.png
occlusion = occlusion.png
Material Properties

UsdPreviewSurface

Example

diffuseColor = (1,1,1)

normal = normals.png

occlusion = occlusion.png

metallic = 1
Material Properties
UsdPreviewSurface

Example

diffuseColor = (1,1,1)
normal = normals.png
occlusion = occlusion.png
metallic = 1
roughness = roughness.png
Material Properties

UsdPreviewSurface

Example

diffuseColor = diffuse.png
normal = normals.png
occlusion = occlusion.png
metallic = metallic.png
roughness = roughness.png
Shader Node Graph

Material

Shader Node → Shader Node → Shader Node
Shader Node Graph
UsdPreviewSurface

Simple shader node graph structure
Four node types

- Main Node (defines PBR attributes) - "UsdPreviewSurface"
- Texture Sampler - "UsdUVTexture"
- Mesh Attribute Reader (e.g. UV) - "UsdPrimvarReader"
- UV Transform - "UsdTransform2d"
Example

Constant material property

Material

UsdPreviewSurface

diffuseColor = (0.84, 0.65, 0.65)
metallic = 1
roughness = 0

surface

"pbr"

"Material1"
Example
Constant material property

def Material "Material1"
{
    def Shader "pbr"
    {
        uniform token info:id = "UsdPreviewSurface"
        color3f inputs:diffuseColor = (0.84, 0.65, 0.65)
        float inputs:metallic = 1
        float inputs:roughness = 0
        token outputs:surface
    }
    token outputs:surface.connect = <.../Material1/pbr.outputs:surface>
}
Example
Constant material property

```python
def Material "Material1"
{
    def Shader "pbr"
    {
        uniform token info:id = "UsdPreviewSurface"
        color3f inputs:diffuseColor = (0.84, 0.65, 0.65)
        float inputs:metallic = 1
        float inputs:roughness = 0
        token outputs:surface
    }
    token outputs:surface.connect = <.../Material1/pbr.outputs:surface>
}
```
Example
Constant material property

def Material "Material1"
{
    def Shader "pbr"
    {
        uniform token info:id = "UsdPreviewSurface"
        color3f inputs:diffuseColor = (0.84, 0.65, 0.65)
        float inputs:metallic = 1
        float inputs:roughness = 0
        token outputs:surface = "../Material1/pbr.outputs:surface"
    }
    token outputs:surface.connect = "../Material1/pbr.outputs:surface"
}
Example
Texture material property

Material

UsdPrimvarReader_float2
varname = "st"

UsdUVTexture
file = @baseColor.png@

UsdPreviewSurface
diffuseColor

"Material1"
def Material "Material1"
{
    def Shader "stAttr"
    {
        uniform token info:id = "UsdPrimvarReader_float2"
        token inputs:varname = "st"
        float2 outputs:result
    }

def Shader "diffuse"
{
    uniform token info:id = "UsdUVTexture"
    asset inputs:file = @baseColor.png@
    float2 inputs:st.connect = <.../Material1/stAttr.outputs:result>
    float3 outputs:rgb
}

def Shader "pbr"
{
    uniform token info:id = "UsdPreviewSurface"
    color3f inputs:diffuseColor.connect = <.../Material1/diffuse.outputs:rgb>
    token outputs:surface
}

token outputs:surface.connect = <.../Material1/pbr.outputs:surface>
}
def Material "Material1"
{
    def Shader "stAttr"
    {
        uniform token info:id = "UsdPrimvarReader_float2"
        token inputs:varname = "st"
        float2 outputs:result
    }
    def Shader "diffuse"
    {
        uniform token info:id = "UsdUVTexture"
        asset inputs:file = @baseColor.png@
        float2 inputs:st.connect = <.../Material1/stAttr.outputs:result>
        float3 outputs:rgb
    }
    def Shader "pbr"
    {
        uniform token info:id = "UsdPreviewSurface"
        color3f inputs:diffuseColor.connect = <.../Material1/diffuse.outputs:rgb>
        token outputs:surface
    }
    token outputs:surface.connect = <.../Material1/pbr.outputs:surface>
}
def Material "Material1"
{
    def Shader "stAttr"
    {
        uniform token info:id = "UsdPrimvarReader_float2"
        token inputs:varname = "st"
        float2 outputs:result
    }

    def Shader "diffuse"
    {
        uniform token info:id = "UsdUVTexture"
        asset inputs:file = @baseColor.png@
        float2 inputs:st.connect = <.../Material1/stAttr.outputs:result>
        float3 outputs:rgb
    }

    def Shader "pbr"
    {
        uniform token info:id = "UsdPreviewSurface"
        color3f inputs:diffuseColor.connect = <.../Material1/diffuse.outputs:rgb>
        token outputs:surface
    }

    token outputs:surface.connect = <.../Material1/pbr.outputs:surface>
}
def Material "Material1"
{
    def Shader "stAttr"
    {
        uniform token info:id = "UsdPrimvarReader_float2"
        token inputs:varname = "st"
        float2 outputs:result
    }

    def Shader "diffuse"
    {
        uniform token info:id = "UsdUVTexture"
        asset inputs:file = @baseColor.png@
        float2 inputs:st.connect = <.../Material1/stAttr.outputs:result>
        float3 outputs:rgb
    }

    def Shader "pbr"
    {
        uniform token info:id = "UsdPreviewSurface"
        color3f inputs:diffuseColor.connect = <.../Material1/diffuse.outputs:rgb>
        token outputs:surface
    }

    token outputs:surface.connect = <.../Material1/pbr.outputs:surface>
}
def Material "Material1"
{
  def Shader "stAttr"
  {
    uniform token info:id = "UsdPrimvarReader_float2"
    token inputs:varname = "st"
    float2 outputs:result
  }

  def Shader "diffuse"
  {
    uniform token info:id = "UsdUVTexture"
    asset inputs:file = @baseColor.png@
    float2 inputs:st.connect = <..../Material1/stAttr.outputs:result>
    float3 outputs:rgb
  }

  def Shader "pbr"
  {
    uniform token info:id = "UsdPreviewSurface"
    color3f inputs:diffuseColor.connect = <..../Material1/diffuse.outputs:rgb>
    token outputs:surface
  }
}

token outputs:surface.connect = <..../Material1/pbr.outputs:surface>
Python Samples
USD essentials

“Samples” folder
• Scene graph
• Mesh
• Material and mesh groups
• Transform animations
• Skinning and skeletal animations
What is USD?

Workflows

USD essentials

USD highlights
Subdivision Surfaces

Composition Engine
Subdivision Surfaces

Composition Engine
Subdivision Surfaces

Efficient representation of curved surfaces

Dynamic tessellation  
(e.g. based on distance to camera)

Great for animated surfaces
OpenSubdiv

Industry standard for subdivision surfaces
Developed by Pixar
Metal shaders for efficient GPU evaluation
Basis for subdivision surfaces in SceneKit
USD and OpenSubdiv

USD has unified description for
• Polygonal mesh data
• Subdivision surface data

Subdivision-specific properties
Composition Engine

Powerful authoring tool

Enables efficient collaboration

Example: references
Scene Graph

Parent

Cube1

Sphere1

Cube1

Sphere1
Duplicate Subgraphs

Parent

Child1
- Cube
- Sphere

Child2
- Cube
- Sphere

Child1
- Cube
- Sphere

Child2
- Cube
- Sphere
References

Parent

Child1

Child2

Group

Cube

Sphere

Parent

Child1

Child2

Cube

Sphere

Cube

Sphere
Composition Engine

References

- Child1
  - Cube
  - Sphere
- Child2
  - Cube
  - Sphere

Parent

Group
- Cube
- Sphere

Child1
- Cube
- Sphere

Child2
- Cube
- Sphere
Composition Engine
Overrides

Parent

Child1
- Cube
- Sphere

Child2
- Cube
- Sphere

Group
- Cube
- Sphere

Parent

Child1
- Cube
- Sphere

Child2
- Cube
- Sphere
Composition Engine

Overrides

Parent

Child1
CUBE
SPHERE

Child2
CUBE
SPHERE
Composition Engine
Overrides

Parent
- Child1
  - Cube
  - Sphere
- Child2
  - Cube
  - Sphere

Group
- Cube
- Sphere

Parent
- Child1
  - Sphere
- Child2
  - Cube
  - Sphere
Subdivision Surfaces

Composition Engine
Python Samples
USD highlights

“Samples” folder
- Subdivision surface with creases
- References with overrides
Summary

Universal Scene Description
• Powerful library for authoring and deploying 3D content

Workflows to create and convert assets to usdz
• usdzconvert converts .obj, .gltf, .fbx, .abc...

Download Python usdz tools
• Includes usdzconvert, pre-compiled USD library and sample scripts
### More Information

developer.apple.com/wwdc19/602

<table>
<thead>
<tr>
<th>Event</th>
<th>Date/Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>USD and usdz Lab</td>
<td>Wednesday, 3:00</td>
</tr>
<tr>
<td>Advances in AR Quick Look</td>
<td>Friday, 9:00</td>
</tr>
</tbody>
</table>