Santa Monica Studio™

Maya as Editor

The game development approach of Santa Monica Studio

Stephen McAuley, Technical Director
Matt Pettineo, Lead Rendering Programmer

Santa Monica Studio

The Speakers

Stephen McAuley
Technical Director

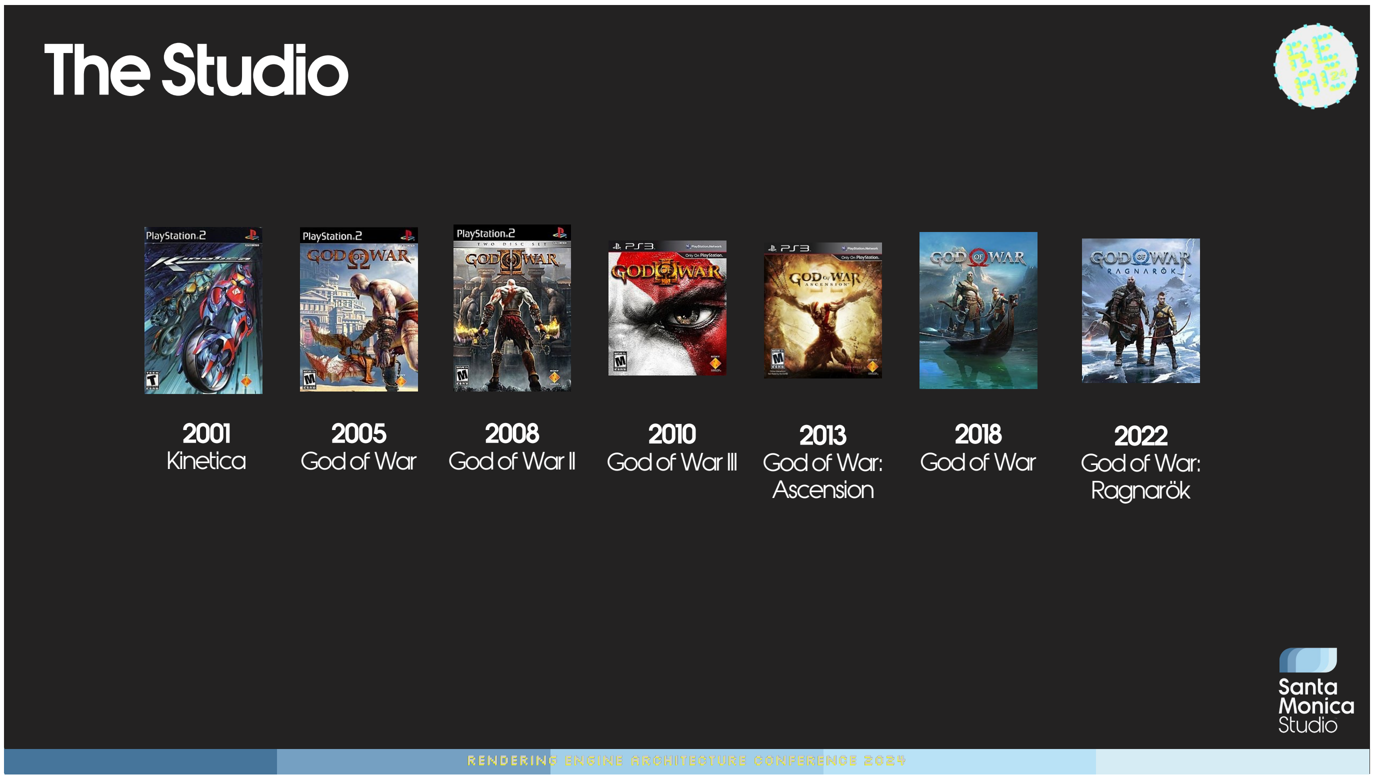
Joined in August 2020

Matt Pettineo
Lead Rendering Programmer

Joined in April 2023

Santa Monica Studio

Both Matt and I are fairly recent to the studio, which started back in 1999. So in talking about the engine and the choices made, we weren't there at the time. However, we both have worked at studios that use Maya as an editor before, and we can observe how our use of Maya at SMS affects the studio culture. So although some of this talk is guess work and assumptions about where we came from, there hopefully is still a lot of good knowledge here



The Studio

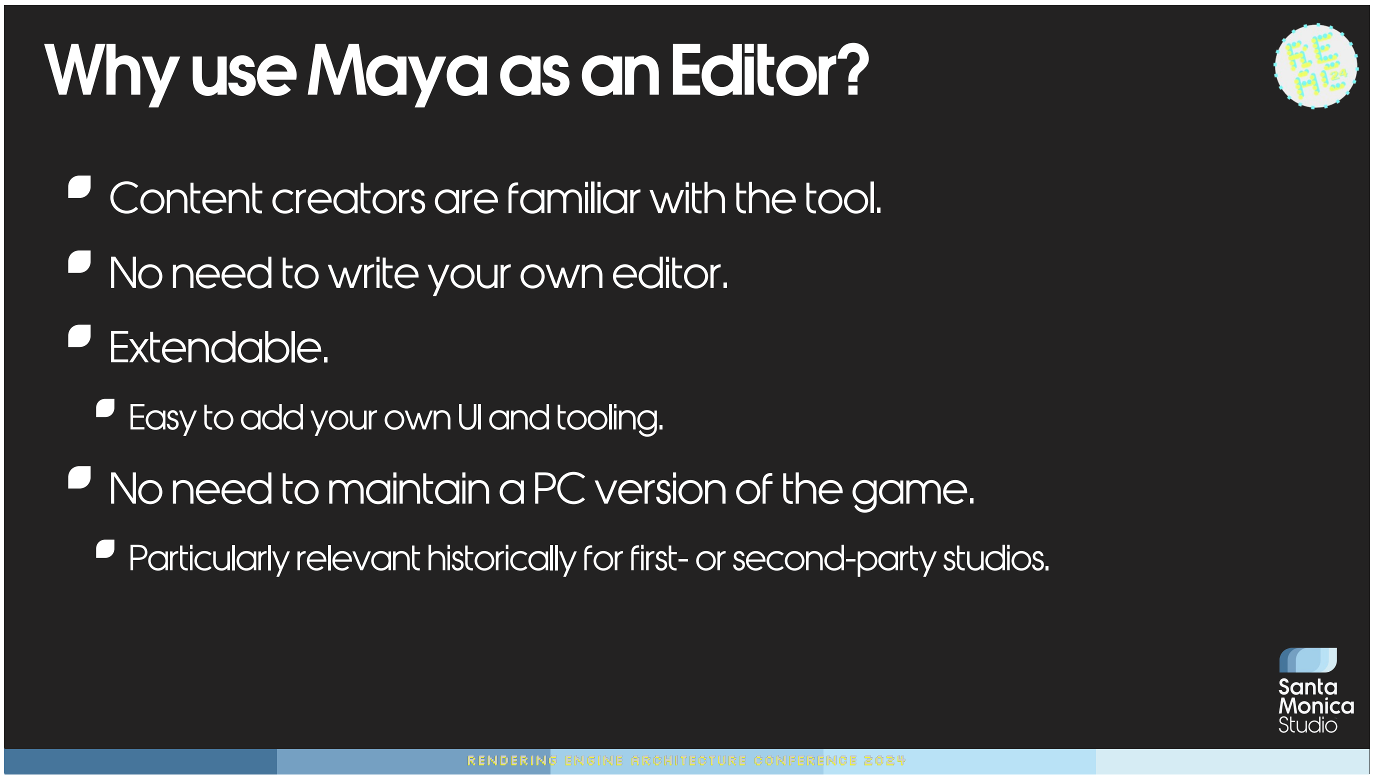
2001	2005	2008	2010	2013	2018	2022
Kinetica	God of War	God of War II	God of War III	God of War: Ascension	God of War	God of War: Ragnarök

Santa Monica Studio

RENDERING ENGINE ARCHITECTURE CONFERENCE 2024

Founded in 1999 and primarily known for the God of War franchise, although we started off with a game called Kinetica, which I didn't even know until I started writing this talk. We're most recently known for the God of War reboot in 2018, and its follow up, our most recent release, God of War: Ragnarök in 2022.

We use Maya as an Editor...

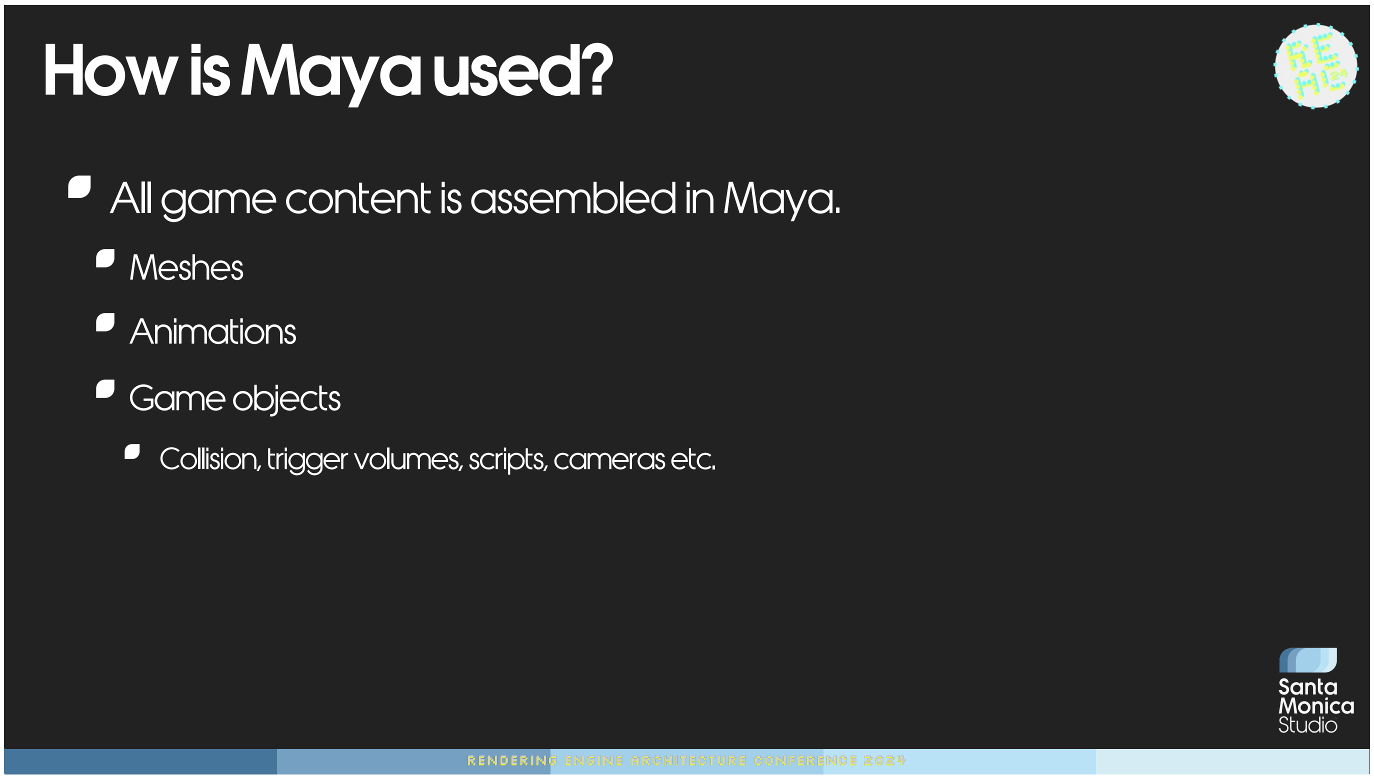


Why use Maya as an Editor?

- Content creators are familiar with the tool.
- No need to write your own editor.
- Extendable.
  - Easy to add your own UI and tooling.
- No need to maintain a PC version of the game.
  - Particularly relevant historically for first- or second-party studios.

Santa Monica Studio

So how did this come about?



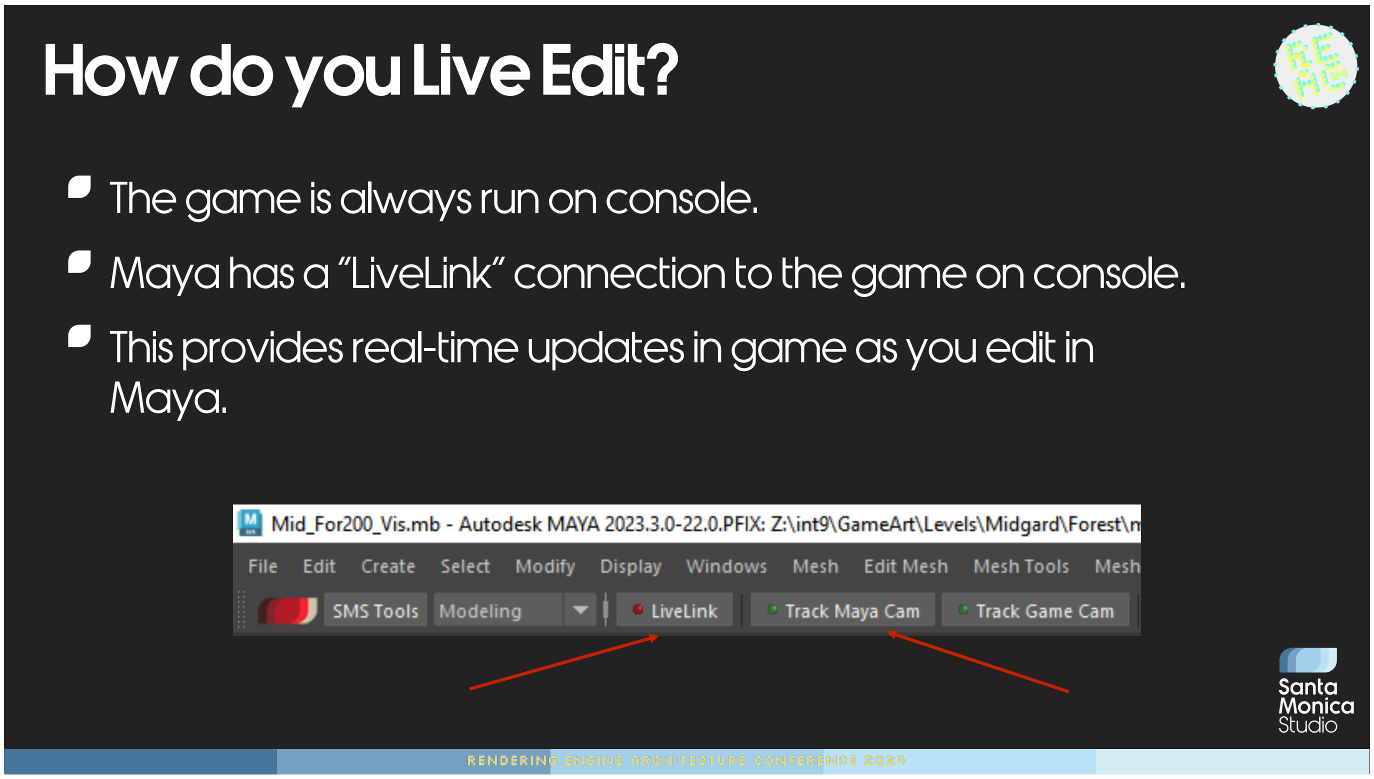
How is Maya used?

- All game content is assembled in Maya.
  - Meshes
  - Animations
  - Game objects
    - Collision, trigger volumes, scripts, cameras etc.

Santa Monica Studio

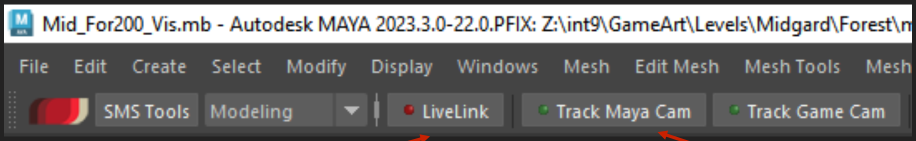
RENDERING ENGINE ARCHITECTURE CONFERENCE 2024

Other editors can of course be used. Either DCCs like Photoshop or Houdini; or in-house tools like our GI baking tool or visual script editor. But fundamentally all data they produce goes through Maya.



How do you Live Edit?

- The game is always run on console.
- Maya has a "LiveLink" connection to the game on console.
- This provides real-time updates in game as you edit in Maya.



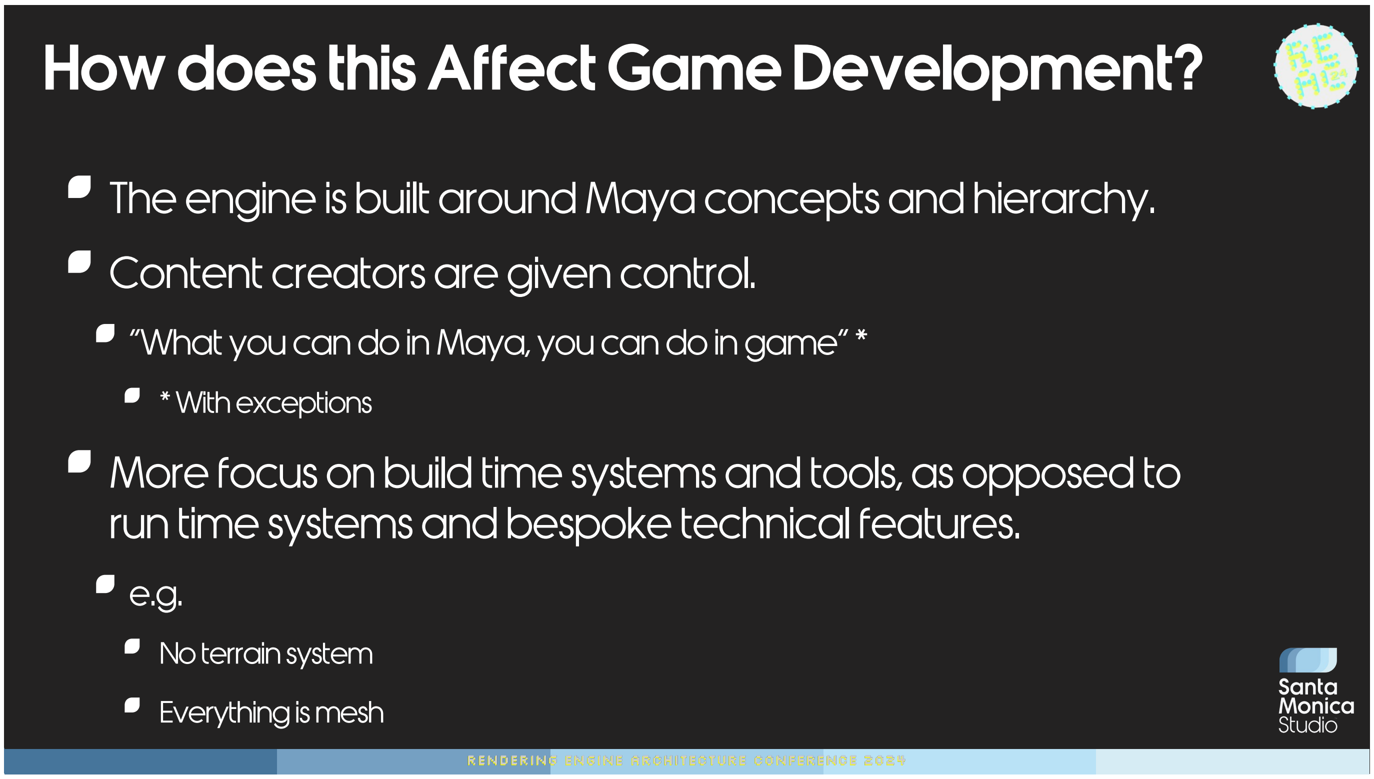
Mid_For200_Vis.mb - Autodesk MAYA 2023.3.0-22.0.PFIX: Z:\int9\GameArt\Levels\Midgard\Forest\n

File Edit Create Select Modify Display Windows Mesh Edit Mesh Mesh Tools Mesh
SMS Tools
Modeling
LiveLink
Track Maya Cam
Track Game Cam

Santa Monica Studio

RENDERING ENGINE ARCHITECTURE CONFERENCE 2025

We will cover how this works later in the presentation.



How does this Affect Game Development?

- The engine is built around Maya concepts and hierarchy.
- Content creators are given control.
  - "What you can do in Maya, you can do in game" *
    - * With exceptions
- More focus on build time systems and tools, as opposed to run time systems and bespoke technical features.
  - e.g.
    - No terrain system
    - Everything is mesh

Santa Monica Studio

This use of Maya has heavily influenced the studio – both technically, and culturally.

One example is how we don't have a terrain system – this of course doesn't map into Maya. Instead everything is just mesh, because that's what Maya supports and expects.

We're going to dig into more influences on the studio later in the presentation.

Base Concepts
How the game is built in Maya

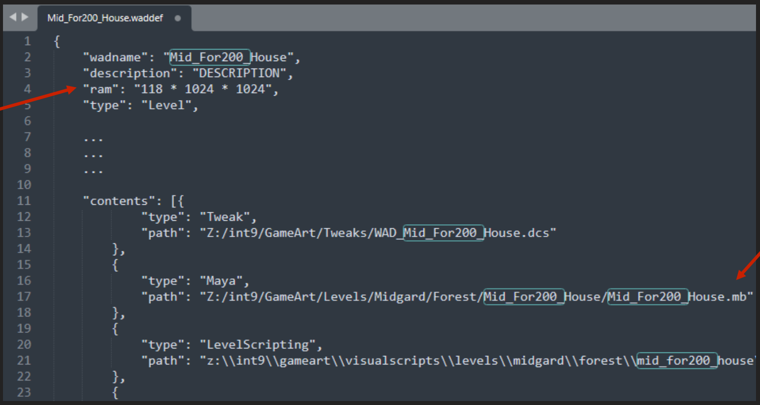
Let’s begin by talking about some base engine concepts, that will help ground us for the rest of the presentation.

Game Architecture

- Our game is split into **wads.**
  - These are the fundamental building blocks of the game that are built offline and loaded/unloaded/streamed by the game.
    - Each wad has its own specified memory budget.
  - Examples:
    - A section of a level is a wad.
    - Kratos is a wad.
    - Atreus is a wad.

Game Architecture

- Each **wad** has a definition file that states which Maya binary files to load/build.



Mid_For200_House.waddef

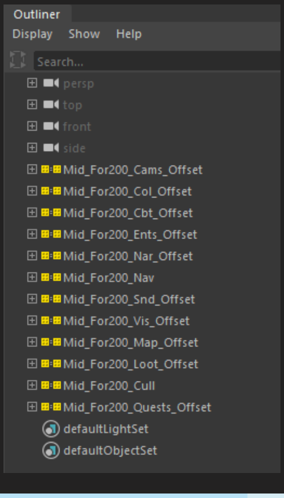
```
{
    "wadname": "Mid_For200_House",
    "description": "DESCRIPTION",
    "ram": "118 * 1024 * 1024",
    "type": "Level",

    ...
    ...
    ...

    "contents": [{
            "type": "Tweak",
            "path": "Z:/int9/GameArt/Tweaks/WAD_Mid_For200_House.dcs"
        },
        {
            "type": "Maya",
            "path": "Z:/int9/GameArt/Levels/Midgard/Forest/Mid_For200_House/Mid_For200_House.mb"
        },
        {
            "type": "LevelScripting",
            "path": "z:\\int9\\gameart\\visualscripts\\levels\\midgard\\forest\\mid_for200_house
        },
        {
```

Game Architecture

- These are typically "parent" .mb files which reference:
  - A "_Vis.mb" file, for visual/art content.
  - A "_Col.mb" file, for collision data.
  - A "_Nav.mb" file, for navmesh data.
  - And more...

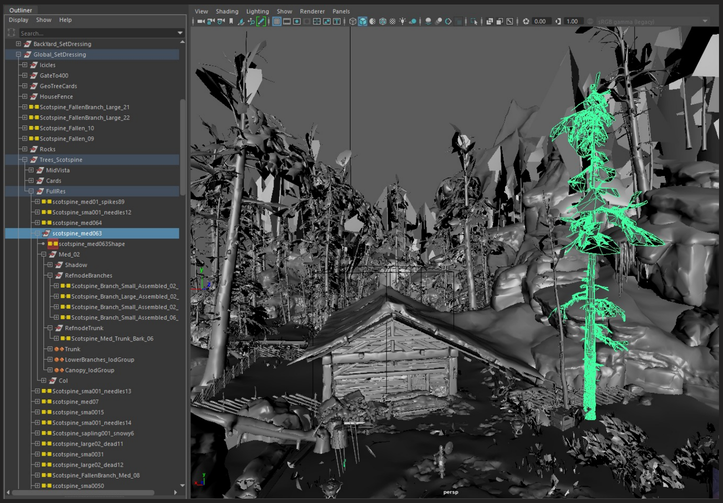


persp
top
front
side
Mid_For200_Cams_Offset
Mid_For200_Col_Offset
Mid_For200_Cbt_Offset
Mid_For200_Ents_Offset
Mid_For200_Nar_Offset
Mid_For200_Nav
Mid_For200_Snd_Offset
Mid_For200_Vis_Offset
Mid_For200_Map_Offset
Mid_For200_Loot_Offset
Mid_For200_Cull
Mid_For200_Quests_Offset
defaultLightSet
defaultObjectSet

Santa Monica Studio

Optimizing Game Objects

- Maya scenes are then long hierarchies of ref nodes:



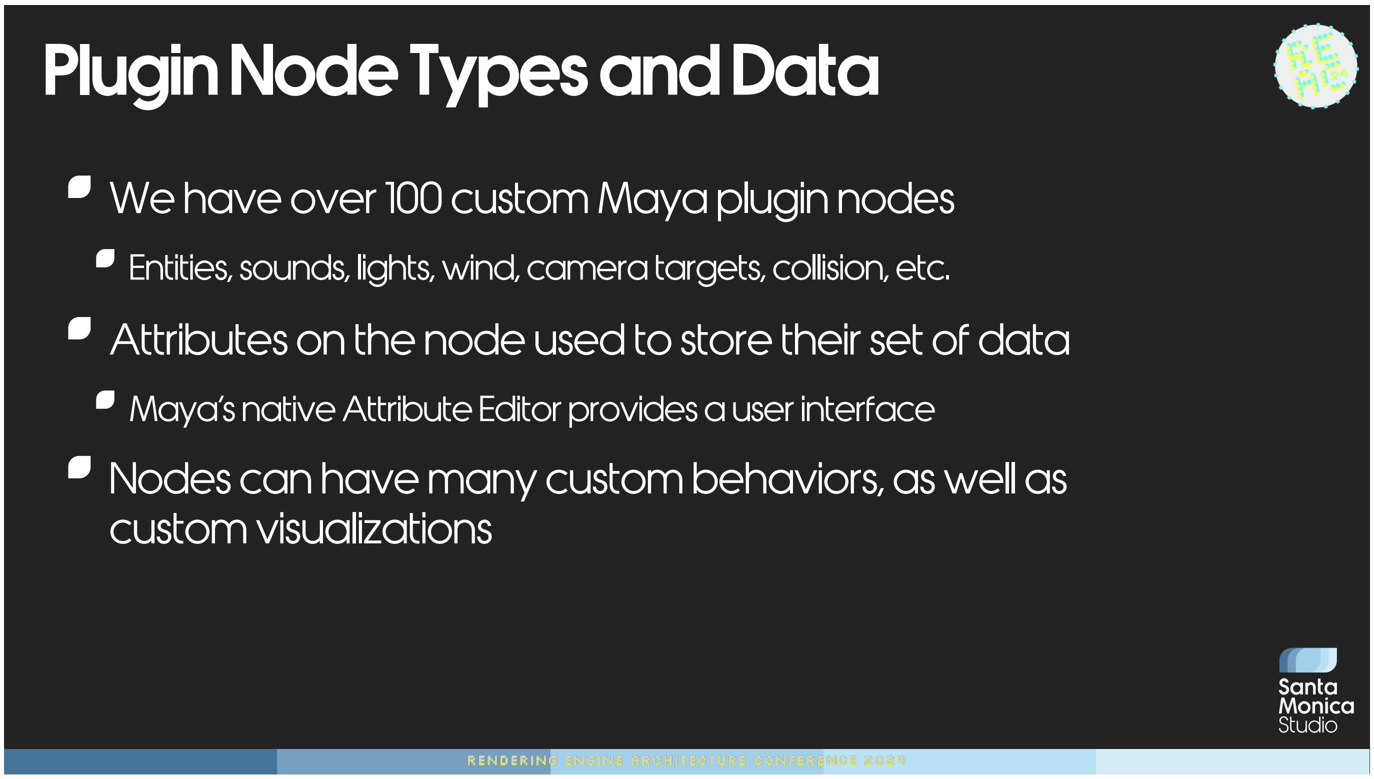
```
BackYard_SetDressing
Global_SetDressing
  Icicles
  GateTo400
  GeoTreeCards
  HouseFence
  Scotspine_FallenBranch_Large_21
  Scotspine_FallenBranch_Large_22
  Scotspine_Fallen_10
  Scotspine_Fallen_09
  Rocks
  Trees_Scotspine
    MidVista
    Cards
    FullRes
      scotspine_med01_spikes89
      Scotspine_sma001_needles12
      scotspine_med064
      scotspine_med063
        scotspine_med063Shape
        Med_02
          Shadow
          RefnodeBranches
            Scotspine_Branch_Small_Assembled_02_
            Scotspine_Branch_Large_Assembled_02_
            Scotspine_Branch_Small_Assembled_02_
            Scotspine_Branch_Small_Assembled_06_
          RefnodeTrunk
            Scotspine_Med_Trunk_Bark_06
          Trunk
          LowerBranches_lodGroup
          Canopy_lodGroup
        Col
      Scotspine_sma001_needles13
      scotspine_med07
      scotspine_sma0015
      Scotspine_sma001_needles14
      Scotspine_sapling001_snowy6
      scotspine_large02_dead11
      scotspine_sma0031
      scotspine_large02_dead12
      Scotspine_FallenBranch_Med_08
      scotspine_sma0050
```

For example, see this tree outside of Kratos' house in Midgard. We have ref nodes for each tree in the level, then each tree has multiple ref nodes for trunk and branches.

Ref Nodes

- A ref node is where a Maya file is referenced into another.
- Our game is fundamentally built from long hierarchies of ref nodes.
- Each ref node is built separately and stitched together using a tool called LinkWad.
  - LinkWad collects all built Maya files that make up a **wad** and links them together.

Santa Monica Studio

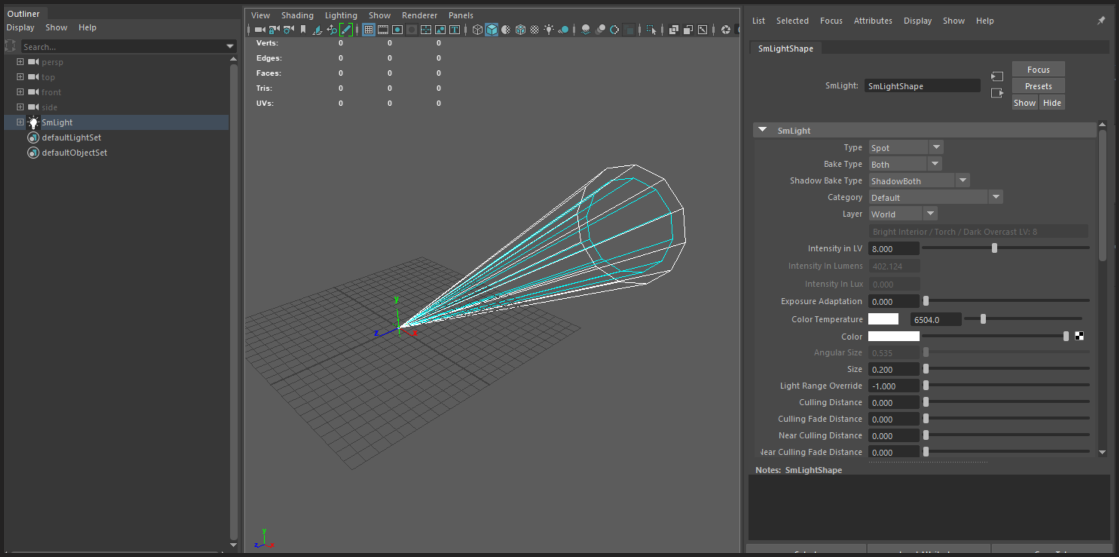


Plugin Node Types and Data

- We have over 100 custom Maya plugin nodes
  - Entities, sounds, lights, wind, camera targets, collision, etc.
- Attributes on the node used to store their set of data
  - Maya's native Attribute Editor provides a user interface
- Nodes can have many custom behaviors, as well as custom visualizations

As we said at the start, Maya is very customizable. Our Maya scenes aren't just full of art.

Example Plugin Node: SMLight



```
Verts:   0   0   0
Edges:   0   0   0
Faces:   0   0   0
Tris:    0   0   0
UVs:     0   0   0
```

SmLightShape

SmLight: SmLightShape

SmLight

Type	Spot
Bake Type	Both
Shadow Bake Type	ShadowBoth
Category	Default
Layer	World
	Bright Interior / Torch / Dark Overcast LV: 8
Intensity in LV	8.000
Intensity In Lumens	402.124
Intensity In Lux	0.000
Exposure Adaptation	0.000
Color Temperature	6504.0
Color	
Angular Size	0.535
Size	0.200
Light Range Override	-1.000
Culling Distance	0.000
Culling Fade Distance	0.000
Near Culling Distance	0.000
Near Culling Fade Distance	0.000

Notes: SmLightShape

DataCompiler Nodes

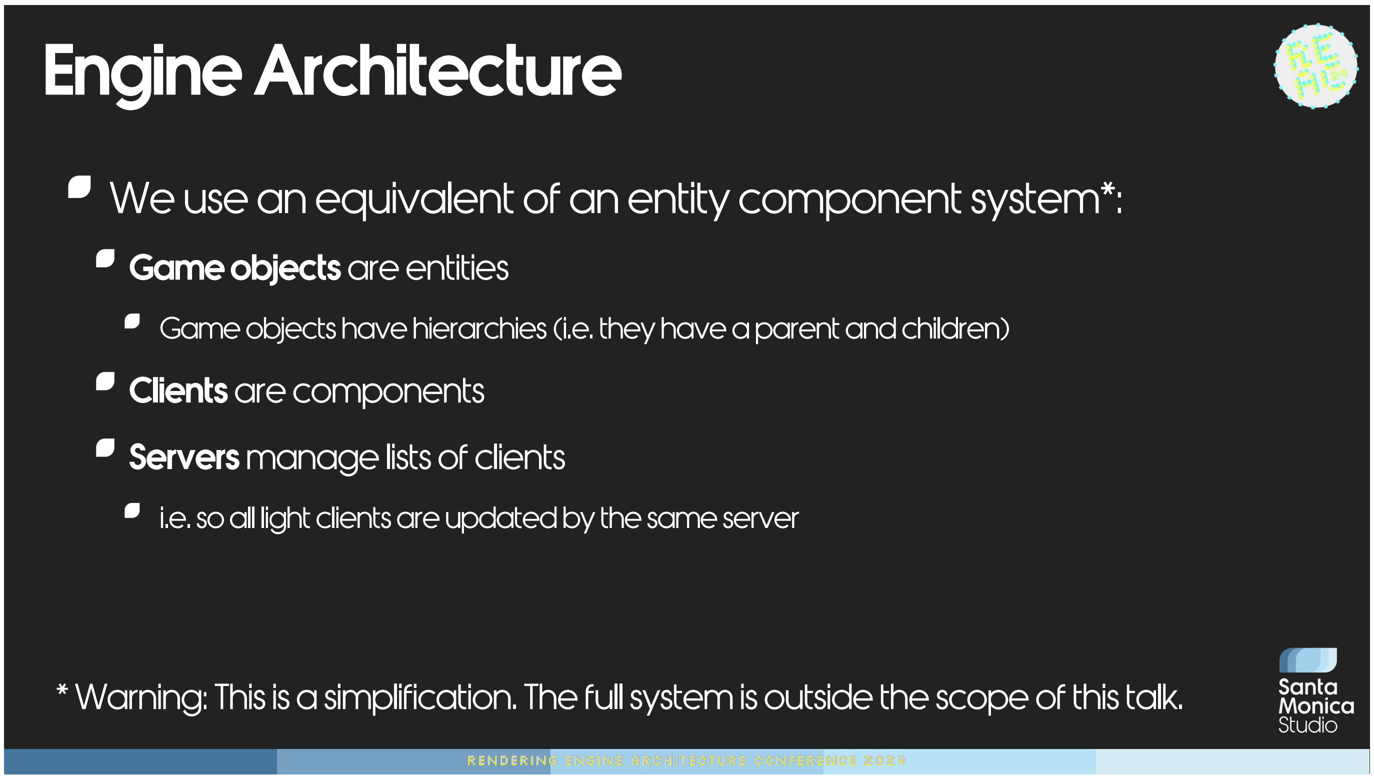
- Many plugin nodes use a generic "DataCompiler" path
- Custom DDL used to describe node properties
- DataCompiler generates C++ code for:
  - Setting up attributes on the Maya plugin node
  - Runtime data class used by the game
- Simplified setup for nodes that just need to store data

Santa Monica Studio

Engine Architecture

Maya scene hierarchy and our engine

Now that we have our grounding about how the game is built in Maya, let's delve deeper into our engine architecture, and how it's influenced by Maya scene hierarchy.



Engine Architecture

- We use an equivalent of an entity component system*:
  - **Game objects** are entities
    - Game objects have hierarchies (i.e. they have a parent and children)
  - **Clients** are components
  - **Servers** manage lists of clients
    - i.e. so all light clients are updated by the same server

* Warning: This is a simplification. The full system is outside the scope of this talk.

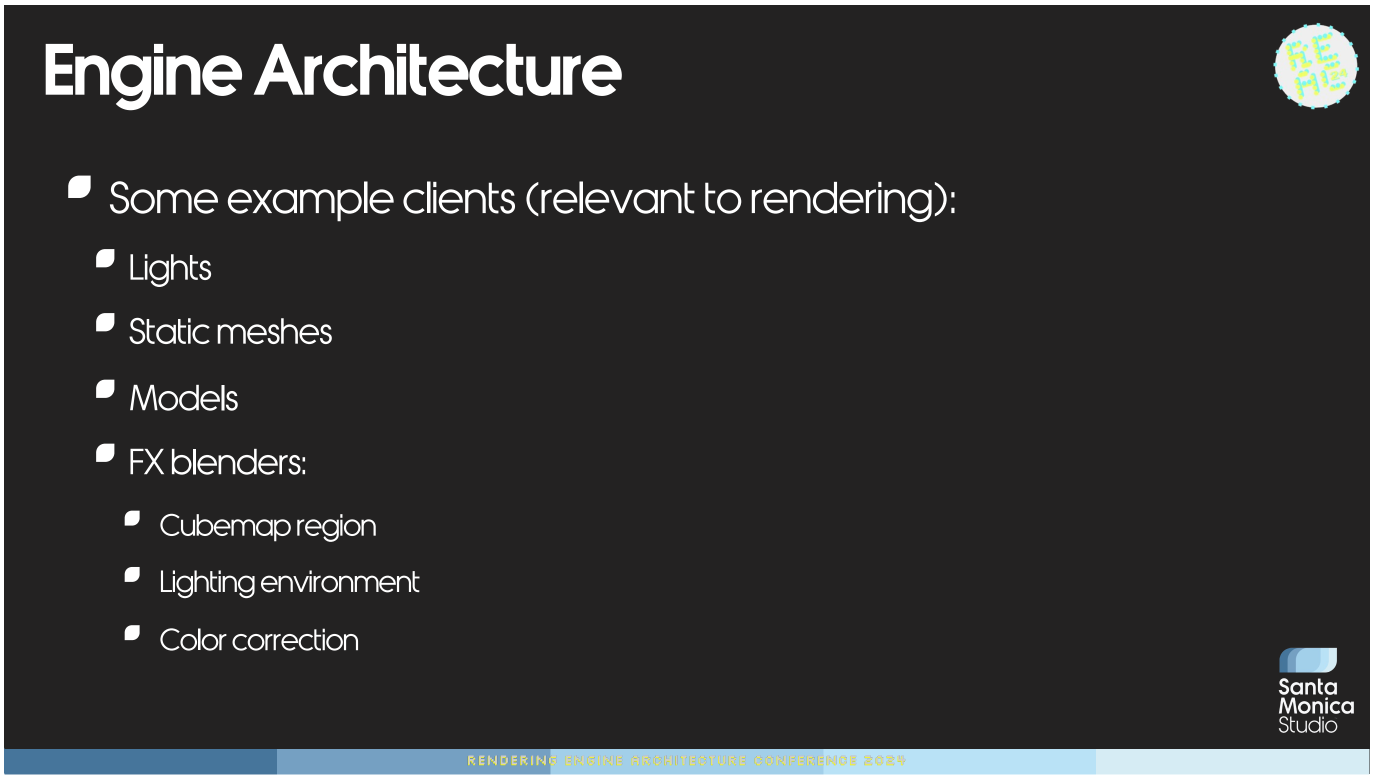
Santa Monica Studio

First, let's explain a little about our architecture.

Having all components of the same type, such as lights, updated by the same server is a good data-oriented design optimization.

This architecture actually goes all the way back to 1999 and the studio's founding!

Some programmers on our team are probably screaming at me right now, saying it's much more complicated than this. It's true. There are cases where servers are entities and clients are also entities, for example. But this whole system is outside the scope of the talk.



Engine Architecture

- Some example clients (relevant to rendering):
  - Lights
  - Static meshes
  - Models
  - FX blenders:
    - Cubemap region
    - Lighting environment
    - Color correction

Santa Monica Studio

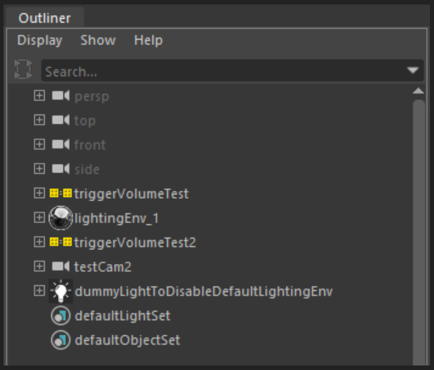
Having all components of the same type, such as lights, updated by the same server is a good data-oriented design optimization.

This architecture actually goes all the way back to 1999 and the studio's founding!

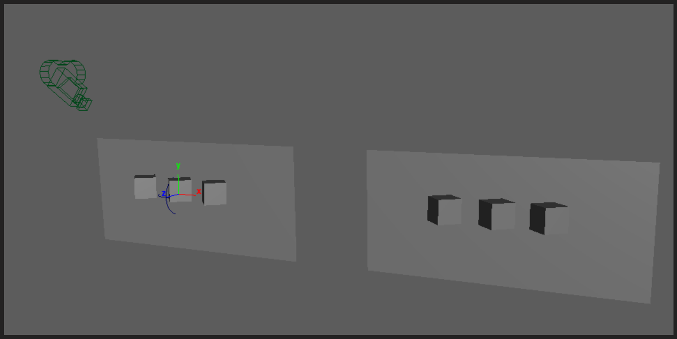
Some programmers on our team are probably screaming at me right now, saying it's much more complicated than this. It's true. There are cases where servers are entities and clients are also entities, for example. But this whole system is outside the scope of the talk.

The Layout of a Maya Scene

- Let's use triggerVolMultiRefnode.mb as an example scene:



Outliner
Display Show Help
Search...
persp
top
front
side
triggerVolumeTest
lightingEnv_1
triggerVolumeTest2
testCam2
dummyLightToDisableDefaultLightingEnv
defaultLightSet
defaultObjectSet



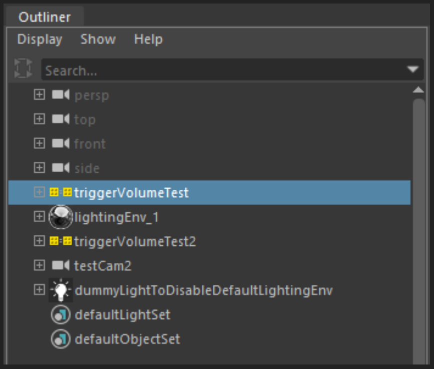
Santa Monica Studio

Let's now move onto Maya.

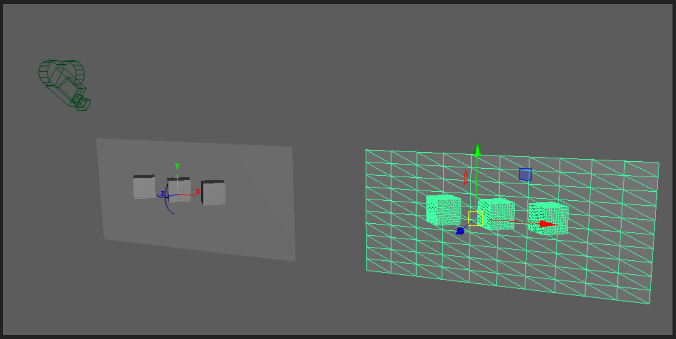
This is one of our visual tests, and it's a good simple example of a scene hierarchy.

The Layout of a Maya Scene

- We have two ref nodes (triggerVolumeTest_source.mb):

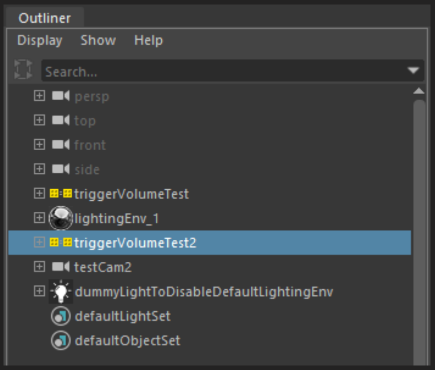


persp
top
front
side
triggerVolumeTest
lightingEnv_1
triggerVolumeTest2
testCam2
dummyLightToDisableDefaultLightingEnv
defaultLightSet
defaultObjectSet

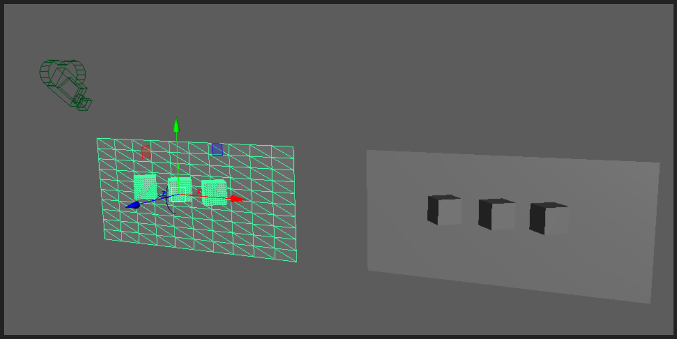


The Layout of a Maya Scene

- We have two ref nodes (triggerVolumeTest_source.mb):



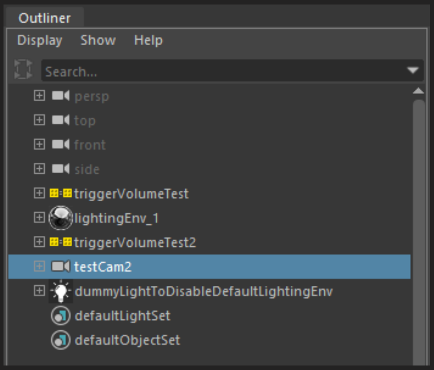
persp
top
front
side
triggerVolumeTest
lightingEnv_1
triggerVolumeTest2
testCam2
dummyLightToDisableDefaultLightingEnv
defaultLightSet
defaultObjectSet



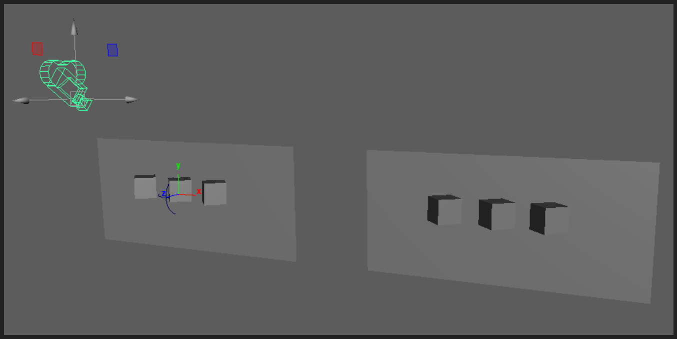
They actually happen to reference the same Maya file

The Layout of a Maya Scene

- A camera:



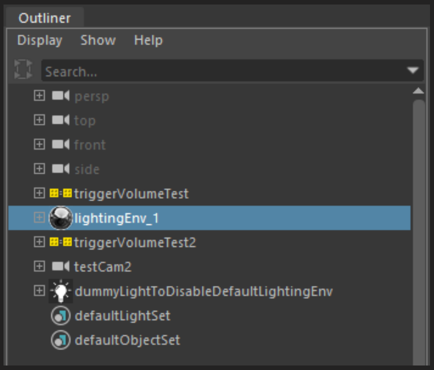
Outliner
Display Show Help
Search...
persp
top
front
side
triggerVolumeTest
lightingEnv_1
triggerVolumeTest2
testCam2
dummyLightToDisableDefaultLightingEnv
defaultLightSet
defaultObjectSet



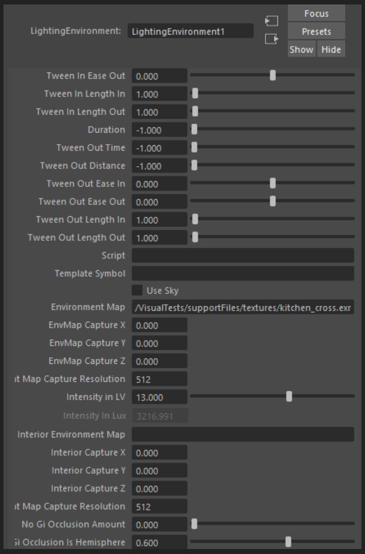
Santa Monica Studio

The Layout of a Maya Scene

- Lighting settings:



persp
top
front
side
triggerVolumeTest
lightingEnv_1
triggerVolumeTest2
testCam2
dummyLightToDisableDefaultLightingEnv
defaultLightSet
defaultObjectSet

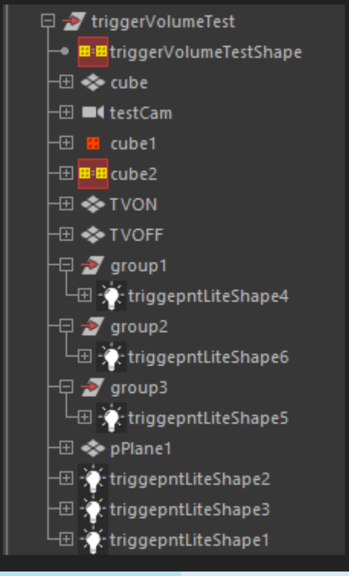


LightingEnvironment:	LightingEnvironment1
Tween In Ease Out	0.000
Tween In Length In	1.000
Tween In Length Out	1.000
Duration	-1.000
Tween Out Time	-1.000
Tween Out Distance	-1.000
Tween Out Ease In	0.000
Tween Out Ease Out	0.000
Tween Out Length In	1.000
Tween Out Length Out	1.000
Script	
Template Symbol	
	Use Sky
Environment Map	/VisualTests/supportFiles/textures/kitchen_cross.exr
EnvMap Capture X	0.000
EnvMap Capture Y	0.000
EnvMap Capture Z	0.000
ıt Map Capture Resolution	512
Intensity in LV	13.000
Intensity In Lux	3216.991
Interior Environment Map	
Interior Capture X	0.000
Interior Capture Y	0.000
Interior Capture Z	0.000
ıt Map Capture Resolution	512
No Gi Occlusion Amount	0.000
;i Occlusion Is Hemisphere	0.600

Santa Monica Studio

The Layout of a Maya Scene

- Let's open the triggerVolumeTest ref node.



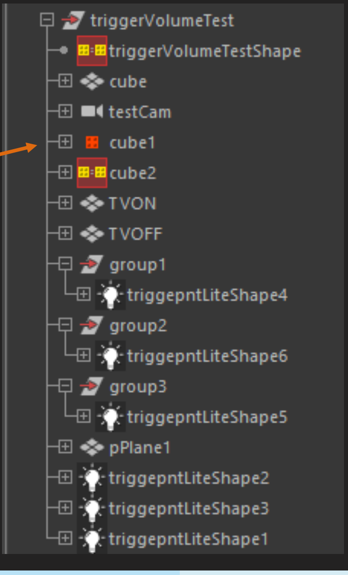
```
triggerVolumeTest
  triggerVolumeTestShape
  cube
  testCam
  cube1
  cube2
  TVON
  TVOFF
  group1
    triggepntLiteShape4
  group2
    triggepntLiteShape6
  group3
    triggepntLiteShape5
  pPlane1
  triggepntLiteShape2
  triggepntLiteShape3
  triggepntLiteShape1
```

Santa Monica Studio

The Layout of a Maya Scene

- Let's open the triggerVolumeTest ref node.

Embedded ref node



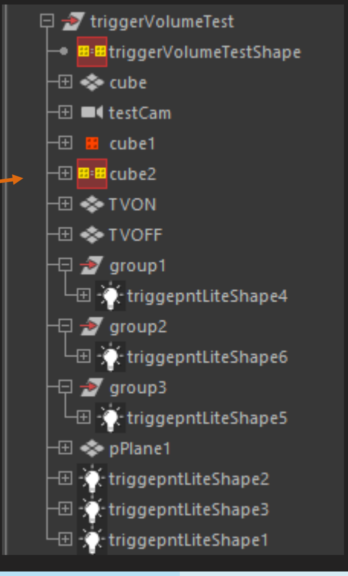
```
⊟ triggerVolumeTest
    triggerVolumeTestShape
  ⊞ cube
  ⊞ testCam
  ⊞ cube1
  ⊞ cube2
  ⊞ TVON
  ⊞ TVOFF
  ⊟ group1
      ⊞ triggepntLiteShape4
  ⊟ group2
      ⊞ triggepntLiteShape6
  ⊟ group3
      ⊞ triggepntLiteShape5
  ⊞ pPlane1
  ⊞ triggepntLiteShape2
  ⊞ triggepntLiteShape3
  ⊞ triggepntLiteShape1
```

Santa Monica Studio

The Layout of a Maya Scene

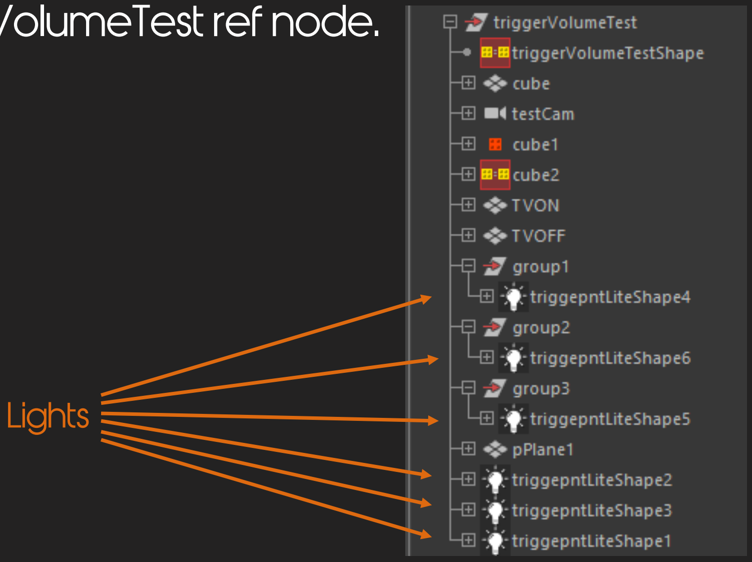
- Let's open the triggerVolumeTest ref node.

Instanced ref node


triggerVolumeTest
triggerVolumeTestShape
cube
testCam
cube1
cube2
TVON
TVOFF
group1
triggepntLiteShape4
group2
triggepntLiteShape6
group3
triggepntLiteShape5
pPlane1
triggepntLiteShape2
triggepntLiteShape3
triggepntLiteShape1

The Layout of a Maya Scene

- Let's open the triggerVolumeTest ref node.

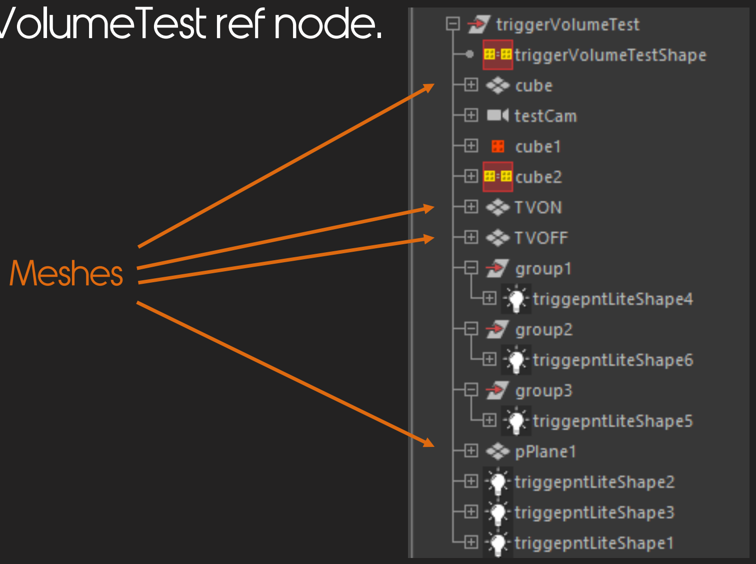


```
⊟ triggerVolumeTest
   triggerVolumeTestShape
 ⊞ cube
 ⊞ testCam
 ⊞ cube1
 ⊞ cube2
 ⊞ TVON
 ⊞ TVOFF
 ⊟ group1
    ⊞ triggepntLiteShape4
 ⊟ group2
    ⊞ triggepntLiteShape6
 ⊟ group3
    ⊞ triggepntLiteShape5
 ⊞ pPlane1
 ⊞ triggepntLiteShape2
 ⊞ triggepntLiteShape3
 ⊞ triggepntLiteShape1
```

Santa Monica Studio

The Layout of a Maya Scene

- Let's open the triggerVolumeTest ref node.

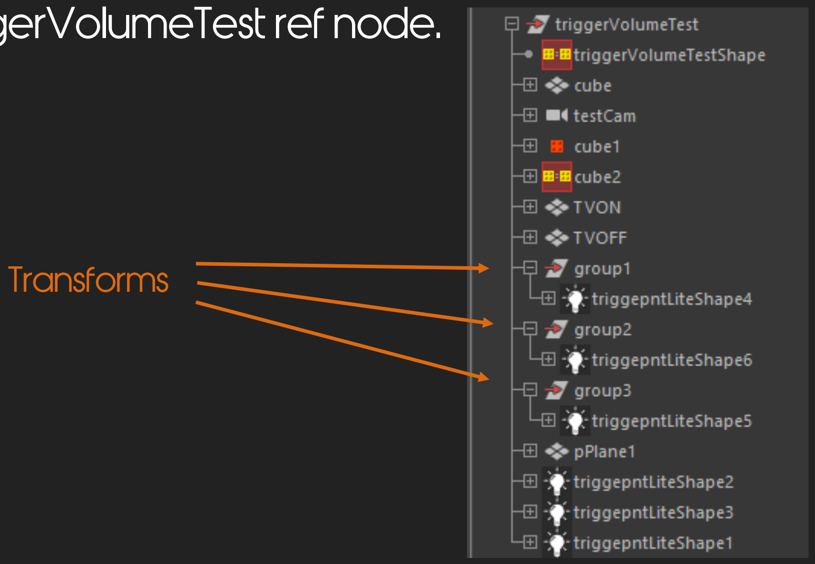


```
triggerVolumeTest
    triggerVolumeTestShape
    cube
    testCam
    cube1
    cube2
    TVON
    TVOFF
    group1
        triggepntLiteShape4
    group2
        triggepntLiteShape6
    group3
        triggepntLiteShape5
    pPlane1
    triggepntLiteShape2
    triggepntLiteShape3
    triggepntLiteShape1
```

Santa Monica Studio

The Layout of a Maya Scene

- Let's open the triggerVolumeTest ref node.

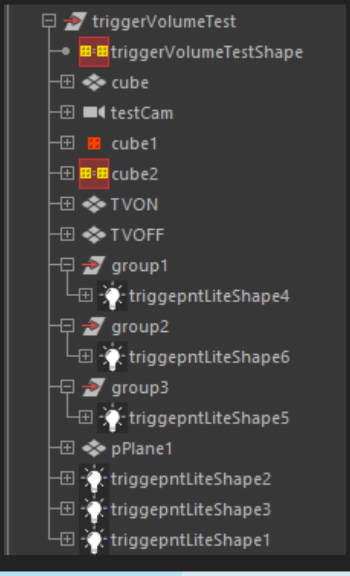


- triggerVolumeTest
  - triggerVolumeTestShape
  - cube
  - testCam
  - cube1
  - cube2
  - TVON
  - TVOFF
  - group1
    - triggepntLiteShape4
  - group2
    - triggepntLiteShape6
  - group3
    - triggepntLiteShape5
  - pPlane1
  - triggepntLiteShape2
  - triggepntLiteShape3
  - triggepntLiteShape1

Santa Monica Studio

The Layout of a Game Scene

- What creates a new game object?
- What becomes a client?

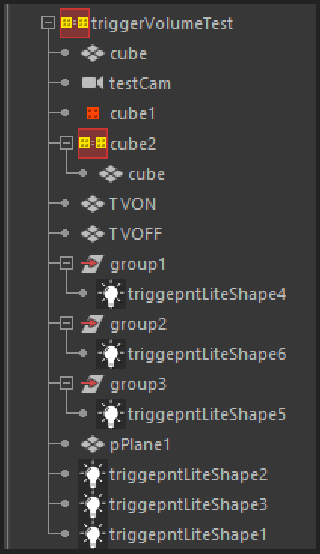


```
triggerVolumeTest
  triggerVolumeTestShape
  cube
  testCam
  cube1
  cube2
  TVON
  TVOFF
  group1
    triggepntLiteShape4
  group2
    triggepntLiteShape6
  group3
    triggepntLiteShape5
  pPlane1
  triggepntLiteShape2
  triggepntLiteShape3
  triggepntLiteShape1
```

Santa Monica Studio

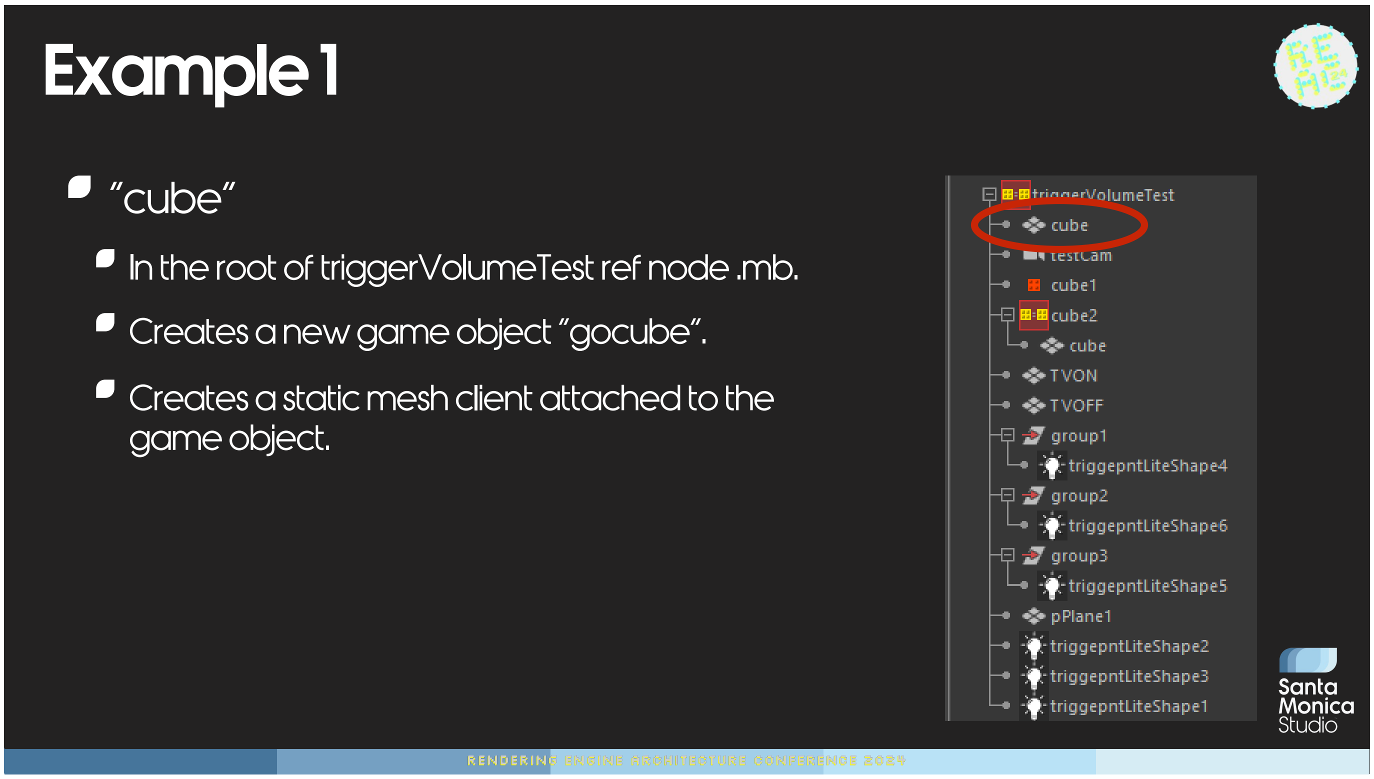
What Becomes a New Game Object?

1. Root transforms in the Maya scene
2. Instance ref node transforms
   - i.e. the parent transform of the actual ref node
3. Transforms with the "new game object" Maya attribute set



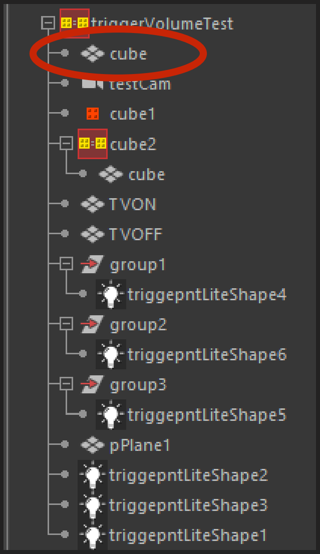
```
triggerVolumeTest
  cube
  testCam
  cube1
  cube2
    cube
  TVON
  TVOFF
  group1
    triggepntLiteShape4
  group2
    triggepntLiteShape6
  group3
    triggepntLiteShape5
  pPlane1
  triggepntLiteShape2
  triggepntLiteShape3
  triggepntLiteShape1
```

Santa Monica Studio



Example 1

- "cube"
  - In the root of triggerVolumeTest ref node .mb.
  - Creates a new game object "gocube".
  - Creates a static mesh client attached to the game object.



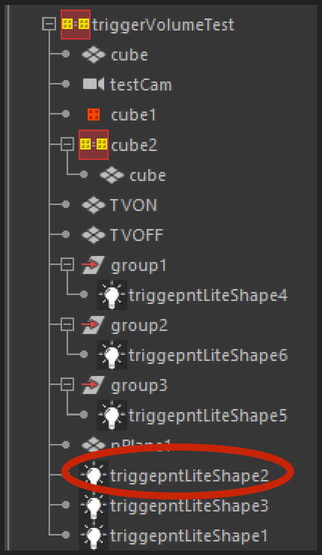
```
triggerVolumeTest
  cube
  testCam
  cube1
  cube2
    cube
  TVON
  TVOFF
  group1
    triggepntLiteShape4
  group2
    triggepntLiteShape6
  group3
    triggepntLiteShape5
  pPlane1
  triggepntLiteShape2
  triggepntLiteShape3
  triggepntLiteShape1
```

Santa Monica Studio

"cube" has a transform at the root of the triggerVolumeTest ref node.

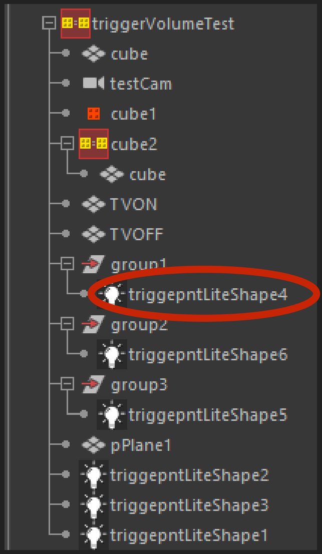
Example 2

- "triggepntLiteShape2"
  - In the root of triggerVolumeTest ref node .mb.
  - Creates a new game object "gotriggepntLiteShape2".
  - Creates a light client attached to the game object.



Example 3

- "triggepntLiteShape4"
  - Under the group1 transform.
  - Does not create a new game object.
  - Creates a light client attached to the game object "gogroup1":
    - group1 *is* a transform at the root of the Maya scene.



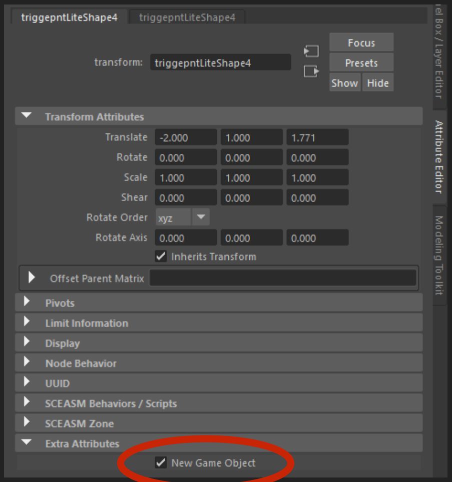
```
triggerVolumeTest
  cube
  testCam
  cube1
  cube2
    cube
  TVON
  TVOFF
  group1
    triggepntLiteShape4
  group2
    triggepntLiteShape6
  group3
    triggepntLiteShape5
  pPlane1
  triggepntLiteShape2
  triggepntLiteShape3
  triggepntLiteShape1
```

Santa Monica Studio

Explicit Game Objects

- You can also flag objects to be explicit game objects.
  - e.g. if you need to refer to them in script.

- "Keep joint" is another flag:
  - The transform for this part of the object is not baked.
  - Adds possibility to animate.



transform: triggepntLiteShape4

Transform Attributes

Translate	-2.000	1.000	1.771
Rotate	0.000	0.000	0.000
Scale	1.000	1.000	1.000
Shear	0.000	0.000	0.000
Rotate Order	xyz		
Rotate Axis	0.000	0.000	0.000

✓ Inherits Transform

Offset Parent Matrix

Pivots

Limit Information

Display

Node Behavior

UUID

SCEASM Behaviors / Scripts

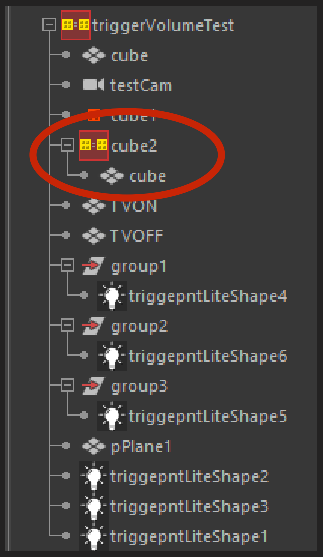
SCEASM Zone

Extra Attributes

✓ New Game Object

Example 4

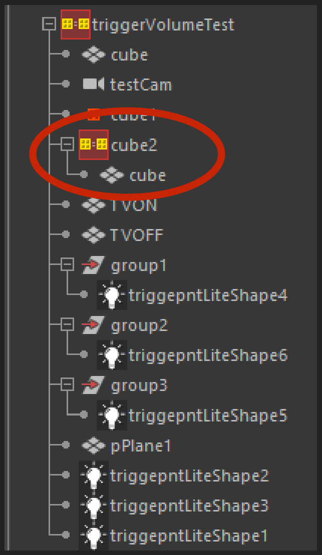
- "cube2"
  - Instance ref node transform.
  - Creates a new game object "gocube2".
- "cube"
  - Root transform in the cube2 ref node Maya scene.
  - Creates a new game object "gocube".



As instanced ref nodes are built separately, we don't know about the surroundings when building. So "cube" is in the root of the cube2 ref node scene, and when cube2 is built, we create a new game object for "cube". When cube2 is instanced into triggerVolumeTest, we also create a new game object for cube2, so we have a ref node hierarchy.

Example 4

- Game object hierarchy:
  - goroot
    - gotriggervolumetest
      - gocube2
        - gocube

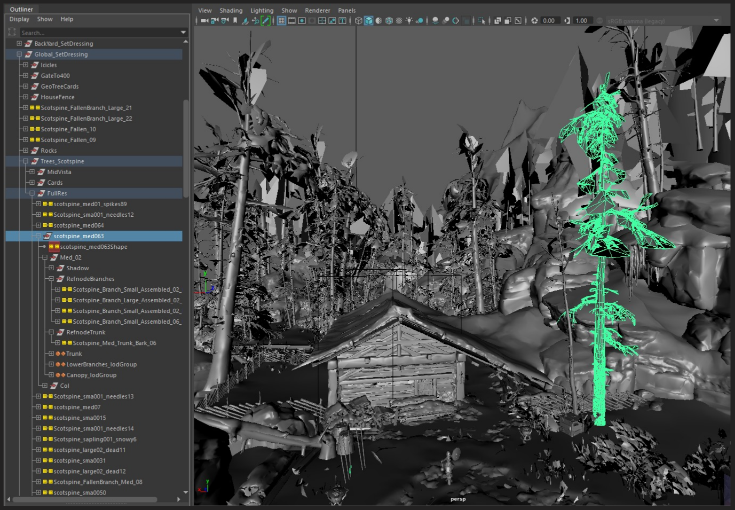


```
triggerVolumeTest
    cube
    testCam
    cube1
    cube2
        cube
    TVON
    TVOFF
    group1
        triggepntLiteShape4
    group2
        triggepntLiteShape6
    group3
        triggepntLiteShape5
    pPlane1
    triggepntLiteShape2
    triggepntLiteShape3
    triggepntLiteShape1
```

As instanced ref nodes are built separately, we don't know about the surroundings when building. So "cube" is in the root of the cube2 ref node scene, and when cube2 is built, we create a new game object for "cube". When cube2 is instanced into triggerVolumeTest, we also create a new game object for cube2, so we have a ref node hierarchy.

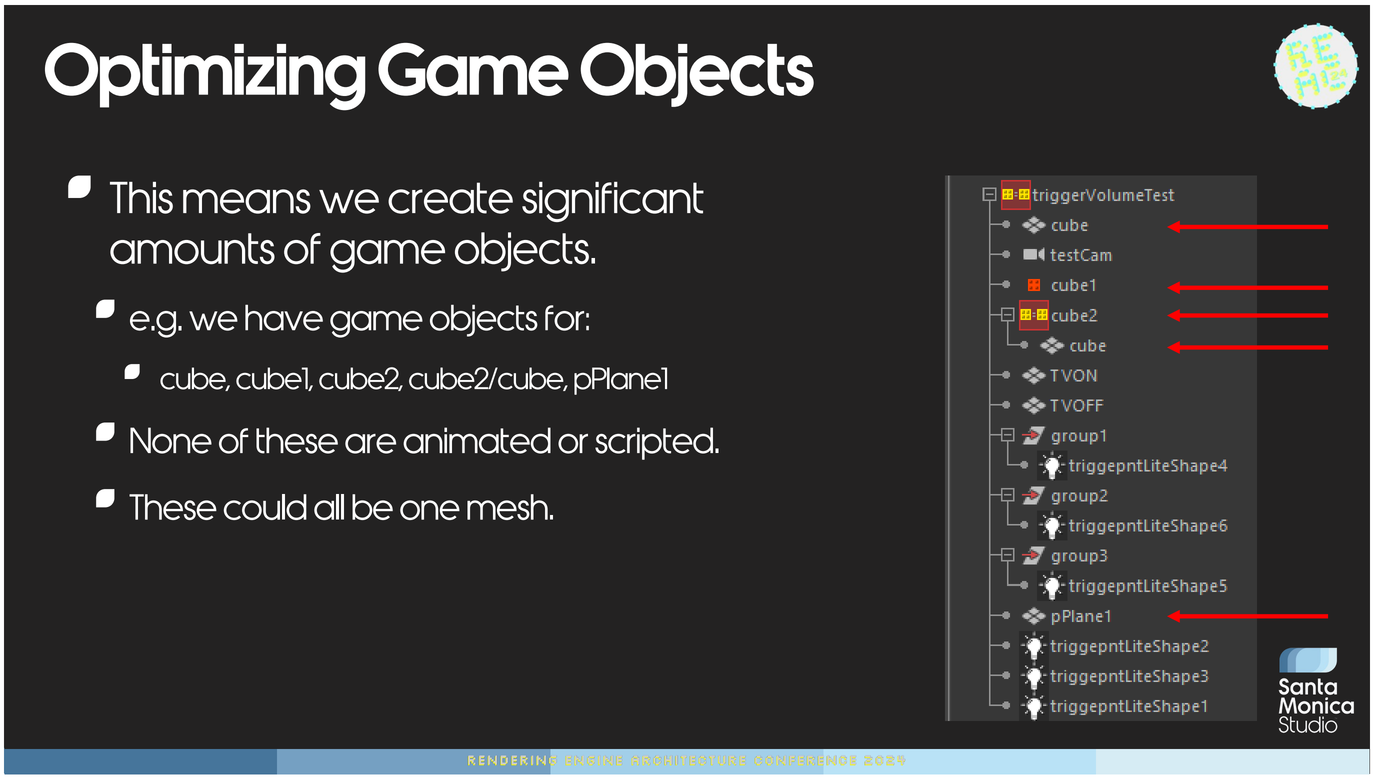
Optimizing Game Objects

- Recall that artists build long hierarchies of ref nodes:



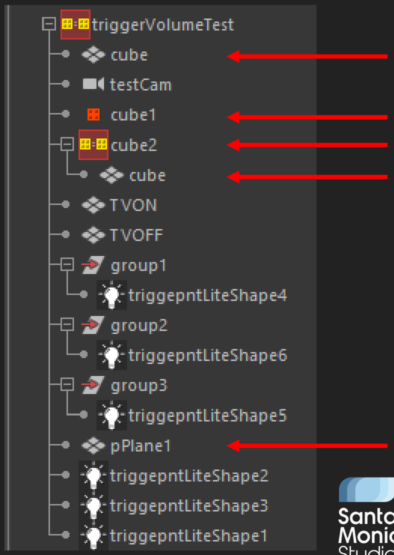
```
BackYard_SetDressing
Global_SetDressing
  Icicles
  GateTo400
  GeoTreeCards
  HouseFence
  Scotspine_FallenBranch_Large_21
  Scotspine_FallenBranch_Large_22
  Scotspine_Fallen_10
  Scotspine_Fallen_09
  Rocks
  Trees_Scotspine
    MidVista
    Cards
    FullRes
      scotspine_med01_spikes89
      Scotspine_sma001_needles12
      scotspine_med064
      scotspine_med063
        scotspine_med063Shape
        Med_02
          Shadow
          RefnodeBranches
            Scotspine_Branch_Small_Assembled_02_
            Scotspine_Branch_Large_Assembled_02_
            Scotspine_Branch_Small_Assembled_02_
            Scotspine_Branch_Small_Assembled_06_
          RefnodeTrunk
            Scotspine_Med_Trunk_Bark_06
          Trunk
          LowerBranches_lodGroup
          Canopy_lodGroup
        Col
      Scotspine_sma001_needles13
      scotspine_med07
      scotspine_sma0015
      Scotspine_sma001_needles14
      Scotspine_sapling001_snowy6
      scotspine_large02_dead11
      scotspine_sma0031
      scotspine_large02_dead12
      Scotspine_FallenBranch_Med_08
      scotspine_sma0050
```

Santa Monica Studio



Optimizing Game Objects

- This means we create significant amounts of game objects.
  - e.g. we have game objects for:
    - cube, cube1, cube2, cube2/cube, pPlane1
  - None of these are animated or scripted.
  - These could all be one mesh.



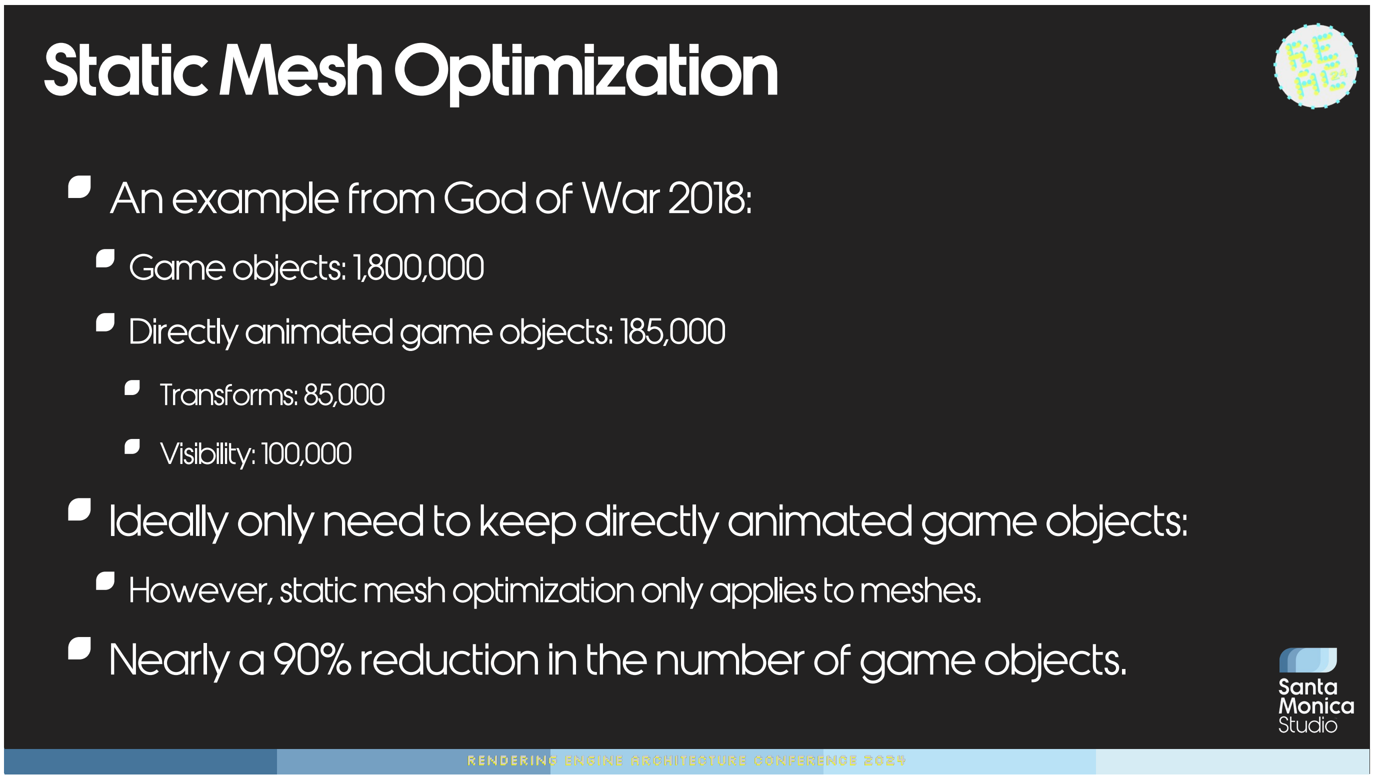
So if we go back to our simple example…

Static Mesh Optimization

- Static mesh optimization introduced in God of War 2018.
- Performed in "LinkWad".
  - Where all the Maya files that constitute a **wad** are stitched together.

- Pulls all suitable mesh instances out of the hierarchy.
- Gathers them into new root-level objects.
  - Transforms and other inherited properties baked in.
- Original game objects are stripped out of the wads.

Santa Monica Studio



Static Mesh Optimization

- An example from God of War 2018:
  - Game objects: 1,800,000
  - Directly animated game objects: 185,000
    - Transforms: 85,000
    - Visibility: 100,000
- Ideally only need to keep directly animated game objects:
  - However, static mesh optimization only applies to meshes.
- Nearly a 90% reduction in the number of game objects.

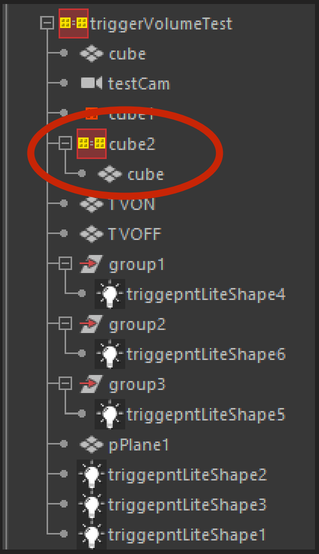
Santa Monica Studio

We had 1.8 million game objects, but only 185,000 were directly animated (just under half had their transform animated, just over half had their visibility animated). We only need to keep directly animated game objects in principle, but as the static mesh optimization applies only to meshes, some non-directly animated game objects for particles, collision etc. would remain.

However, in total, that meant we kept just 185,000 game objects out of an initial 1.8 million, which is nearly a 90% reduction!

LiveLink

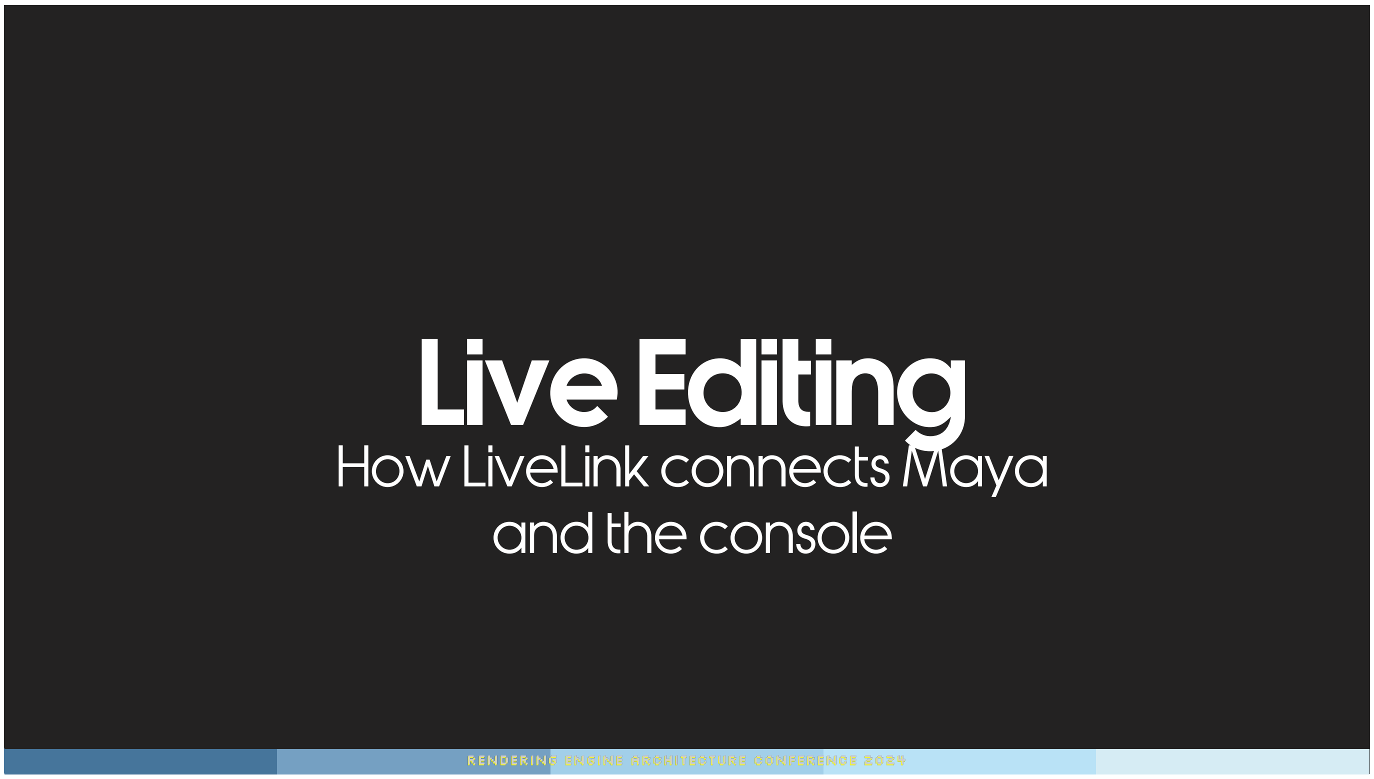
- We've just flattened and merged part of our ref node hierarchy… but what if we need it?
- LiveLink:
  - Our tool that sends live updates from Maya to the game.
- If I move cube2, how do I tell the game to update its transform?
  - We've merged and flattened it.
- We keep the hierarchy as "LiveLink" only.



```
triggerVolumeTest
  cube
  testCam
  cube1
  cube2
    cube
  TVON
  TVOFF
  group1
    triggepntLiteShape4
  group2
    triggepntLiteShape6
  group3
    triggepntLiteShape5
  pPlane1
  triggepntLiteShape2
  triggepntLiteShape3
  triggepntLiteShape1
```

Santa Monica Studio

LiveLink is a really big part of our engine, so we need to talk some more about it…



Live Editing

How LiveLink connects Maya and the console

RENDERING ENGINE ARCHITECTURE CONFERENCE 2024

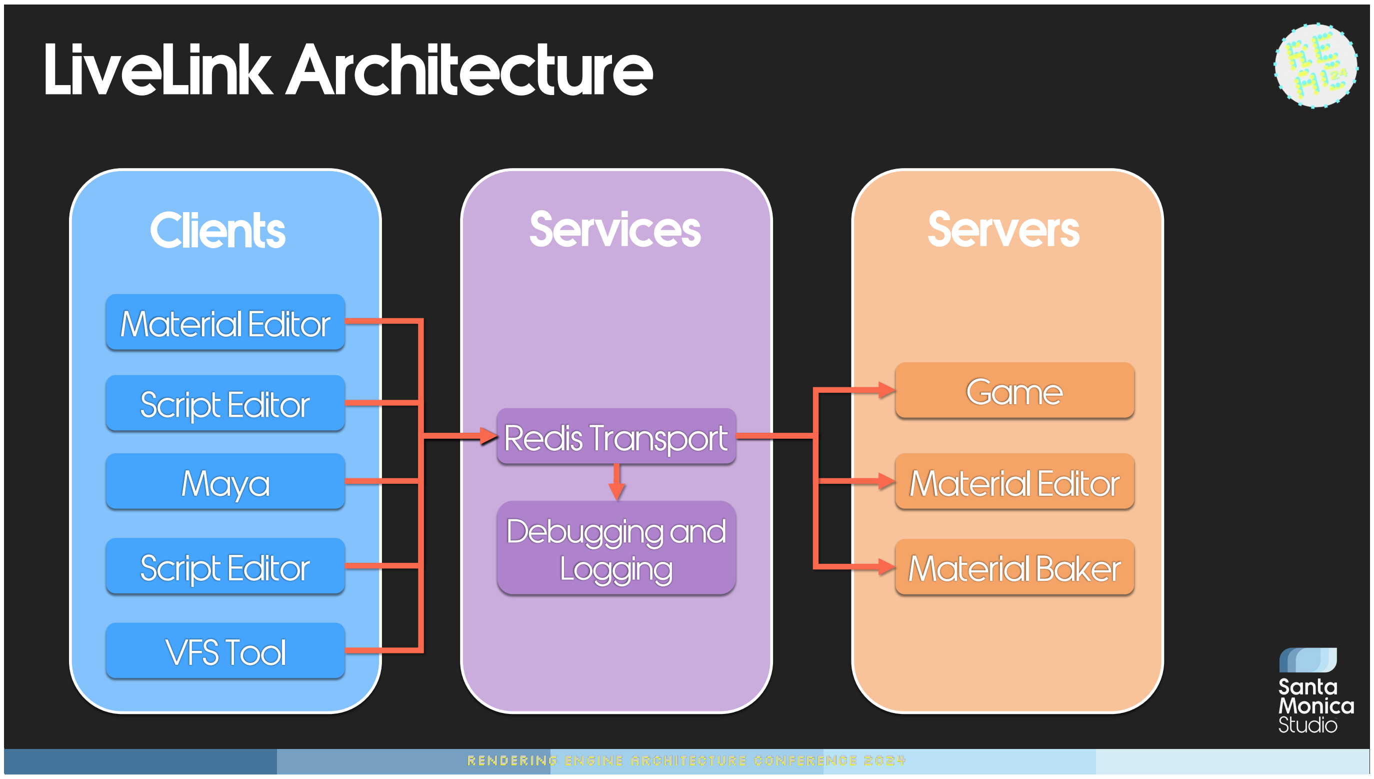
I'm now going to talk a bit how we do live editing at SMS as part of our core workflows, as well as the underlying technology that we use for implementing live editing.

LiveLink

- Generic framework for passing data between tools and the game
- "Edit the game while it's running"
- Allows editing tools to run on the PC while the game runs on a console
- See final visual results without a PC version of the game or renderer

"LiveLink" is the name we use internally at SMS to refer to our framework for passing data between our various PC-based tools as well as the game running on the console. The primary goal of this framework is to essentially allow tools to live edit aspects of the game while it's running on the console, which our tools generally do by sending messages and data across the network. This ability to live edit is extremely important for development as we do not have a version of the game or engine that runs on PC, which means the game must be running on the console in order to actually see the final rendering, animation, or gameplay results. While we do have some limited ability to preview assets directly on the PC (which I'll talk about later), in general we rely very heavily on LiveLink workflows to be get feedback on changes as we're making them.

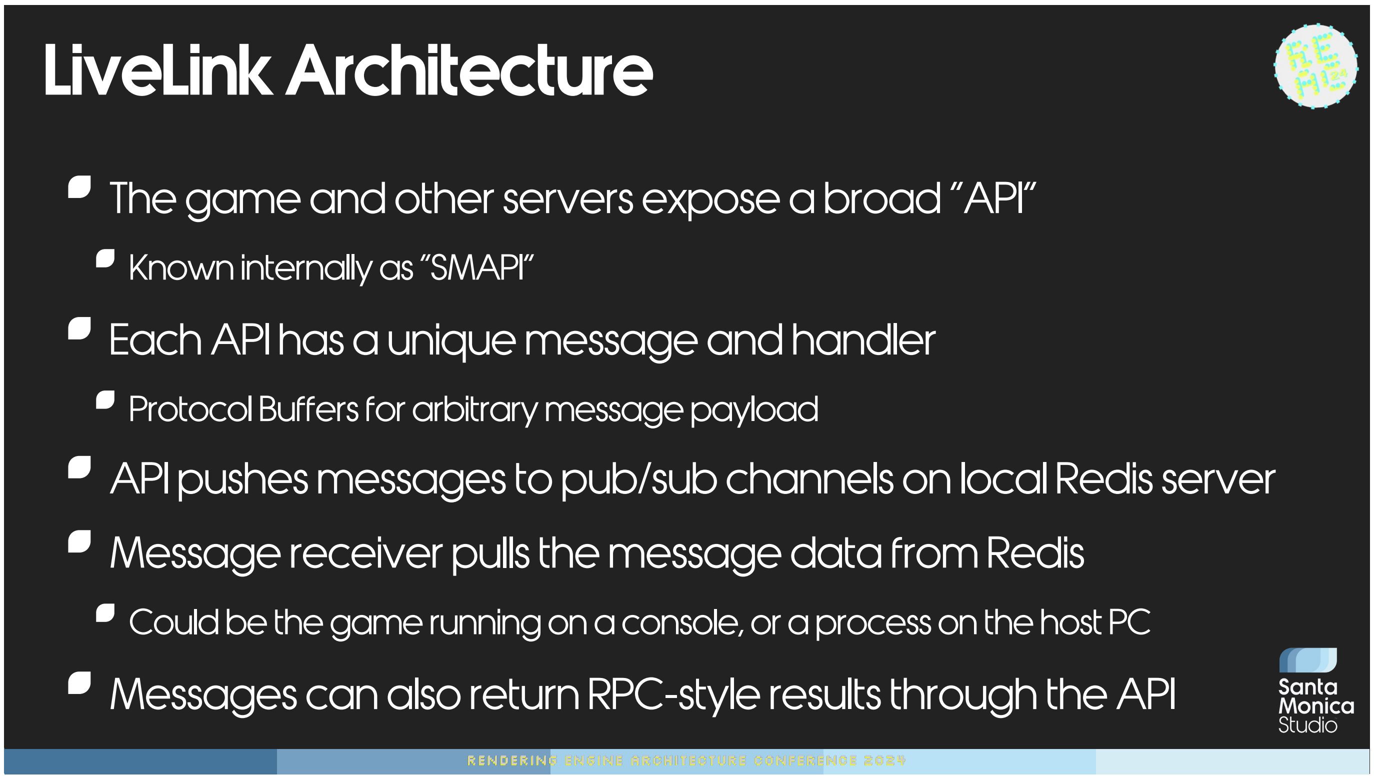
LiveLink Architecture



Before I go over the use cases for LiveLink, I wanted to give a brief overview of how LiveLink works under the hood. To start, we have a diagram here showing the general connection and data flow for LiveLink. On the left we have a group of LiveLink "clients", which are each an application that can make a connection to the central services so that they can ultimately send messages to LiveLink servers. These clients are typically custom in-house tools that run on the PC, but can also include third-party tools such as Maya where we integrate LiveLink via a plugin.

The middle shows our persistent services, which we refer to as "sms-services" internally. These services area always running in the background on the developer's host PC. The Redis Transport service is the important one here, but we also have some other services available for other use cases. We also have a service that provides debugging and logging for all LiveLink traffic, which can be useful for inspecting what a tool is doing or diagnosing a problem.

Finally we have our LiveLink servers on the right, which are apps that receive LiveLink messages and can respond to them. The main server is the game, which is the recipient of the majority of our LiveLink messages. However other tools such as the material editor or material baking system can also act as servers to allow the game to send its own messages to these tools.



LiveLink Architecture

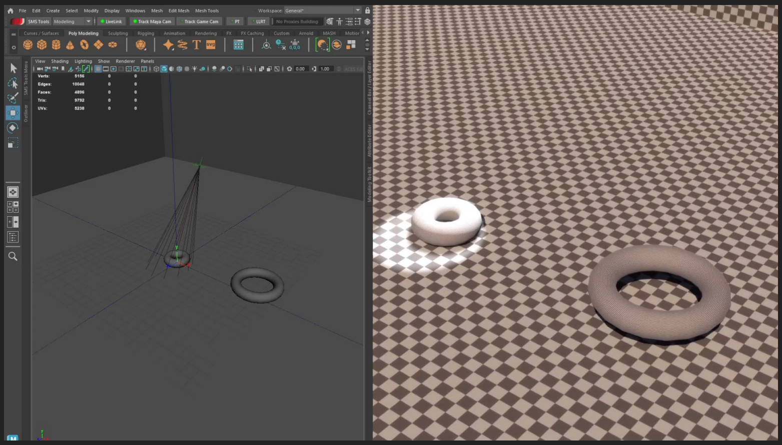
- The game and other servers expose a broad "API"
  - Known internally as "SMAPI"
- Each API has a unique message and handler
  - Protocol Buffers for arbitrary message payload
- API pushes messages to pub/sub channels on local Redis server
- Message receiver pulls the message data from Redis
  - Could be the game running on a console, or a process on the host PC
- Messages can also return RPC-style results through the API

Santa Monica Studio

Under the hood, LiveLink works with a set of wrapper functions that's known internally as "SMAPI". These basically server as a broad API that clients can use to communicate with the game to send it messages and data, with the calling code basically calling regular C++ or C# functions. Each API call has a unique message that gets sent, which can include an arbitrary payload. We use Protocol Buffers to automatically generate the message payload structs from the protobuf DDL, which also handles serialization for us. The way that communication happens is that the LiveLink system opens a network connection to the Redis server that's running locally on the host PC, and uses Redis pub/sub channels to publish the messages to a particular channel. These channels roughly map to a particular LiveLink server, which allows those servers to pull the messages intended for it from Redis. Since it's happening over the network this allows servers to be running on a console, or it could just be a process on the host PC. Messages also allow replying with a result, which essentially allows the API to act as remote procedure calls.

LiveLink Clients

- Maya
  - Update node transforms and properties

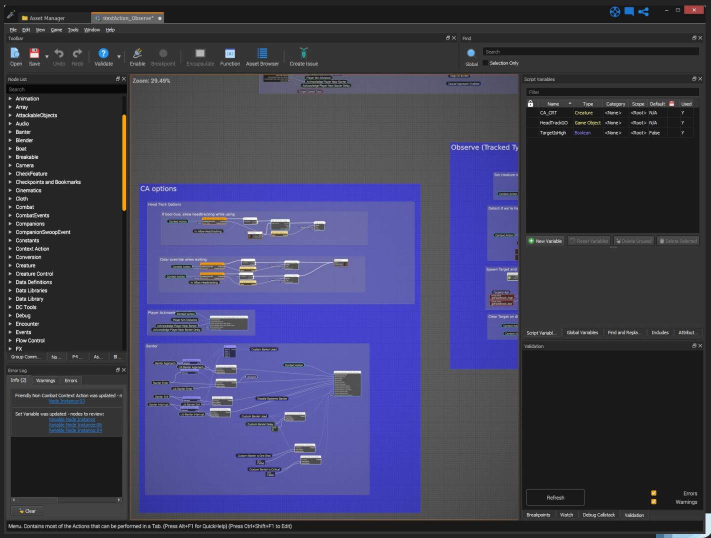


Santa Monica Studio

Let’s now look at some of the clients that make use of LiveLink. To start, we have Maya which connects to the game to automatically propagate edits to the scene in real time. I’ll go into some of the details in a bit, but here in the video you can see Maya on the left and the game running on the right. As I re-position the camera in Maya or move the nodes around, the changes are immediately reflected in the running game. We also support a limited ability to duplicate existing meshes and nodes, which you can see me do towards the end. In practice there are all kinds of things you can do through LinkLink in Maya, but I’ll talk a bit more about this later.

LiveLink Clients

- Visual Script Editor
  - Script breakpoints and live edit
  - See our GDC 2023 presentation



Santa Monica Studio

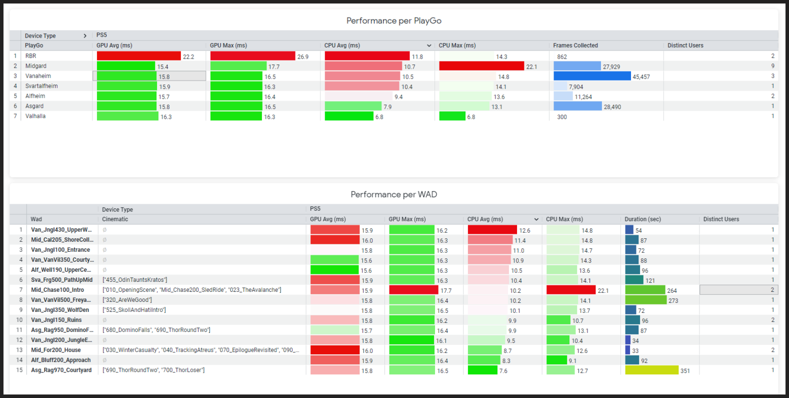
RENDERING ENGINE ARCHITECTURE CONFERENCE 2024

Here we have our visual script editor, which is a newer tool and system that was added for God Of War: Ragnarok to replace the previous lua-based scripting system. The script editor has debugging functionality that works by connecting to the game over LiveLink and communicating state. The editor can set breakpoints to know when a particular node has been executed, and can also propagate changes to the game without having to restart.

For more information about the visual scripting system, you can check out our presentation from GDC 2023: https://www.gdcvault.com/play/1028787/-God-of-War-Ragnark

LiveLink Clients

- Automated performance captures
  - Gather CPU and GPU performance stats from the running game



Performance per PlayGo

	Device Type	PS5					
	PlayGo	GPU Avg (ms)	GPU Max (ms)	CPU Avg (ms)	CPU Max (ms)	Frames Collected	Distinct Users
1	RBR	22.2	26.9	11.8	14.3	862	2
2	Midgard	15.4	17.7	10.7	22.1	27,929	9
3	Vanaheim	15.8	16.5	10.5	14.8	45,457	3
4	Svartalfheim	15.9	16.3	10.4	14.1	7,904	1
5	Alfheim	15.7	16.4	9.4	13.6	11,264	2
6	Asgard	15.8	16.5	7.9	13.1	28,490	1
7	Valhalla	16.3	16.3	6.8	6.8	300	1

Performance per WAD

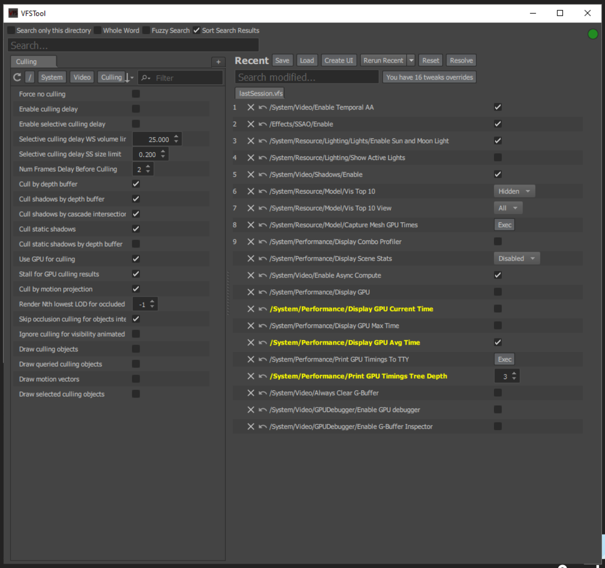
		Device Type	PS5					
	Wad	Cinematic	GPU Avg (ms)	GPU Max (ms)	CPU Avg (ms)	CPU Max (ms)	Duration (sec)	Distinct Users
1	Van_Jngl430_UpperW...	∅	15.9	16.2	12.6	14.8	54	1
2	Mid_Cal205_ShoreColl...	∅	16.0	16.3	11.4	14.8	87	1
3	Van_Jngl100_Entrance	∅	15.8	16.3	11.0	14.7	72	1
4	Van_VanVil350_Courty...	∅	15.6	16.3	10.9	14.3	88	1
5	Alf_Well190_UpperCe...	∅	15.6	16.3	10.5	13.6	96	1
6	Sva_Frg500_PathUpMid	["455_OdinTauntsKratos"]	15.9	16.3	10.4	14.1	121	1
7	Mid_Chase100_Intro	["010_OpeningScene", "Mid_Chase200_SledRide", "023_TheAvalanche"]	15.9	17.7	10.2	22.1	264	2
8	Van_VanVil500_Freya...	["320_AreWeGood"]	15.8	16.4	10.2	14.1	273	1
9	Van_Jngl350_WolfDen	["525_SkollAndHatiIntro"]	15.8	16.5	10.1	13.7	72	1
10	Van_Jngl150_Ruins	∅	15.8	16.2	9.9	10.7	96	2
11	Asg_Rag950_DominoF...	["680_DominoFalls", "690_ThorRoundTwo"]	15.7	16.4	9.9	13.1	87	1
12	Van_Jngl200_JungleE...	∅	15.8	16.1	9.5	10.4	34	1
13	Mid_For200_House	["030_WinterCasualty", "040_TrackingAtreus", "070_EpilogueRevisited", "090_...	16.0	16.2	8.7	12.6	33	2
14	Alf_Bluff200_Approach	∅	15.9	16.4	8.3	9.1	92	1
15	Asg_Rag970_Courtyard	["690_ThorRoundTwo", "700_ThorLoser"]	15.8	16.5	7.6	12.7	351	1

Santa Monica Studio

As an example of a more unusual client, we also our automatic performance capture and telemetry framework. This system utilizes LiveLink to gather performance statistics from the game itself as it's being run by developers, which can then be visualized and queried through a web interface. The data can be used to easily see which areas are over-budget and how they're trending over time. The tools also support generating heat maps which can also be really useful for locating problematic content.

LiveLink Clients

- VFSTool
  - "Virtual File System"
  - Tweak variables/settings from host PC



VFSTool

Search only this directory | Whole Word | Fuzzy Search | ✓ Sort Search Results

Search...

Culling

/ System Video Culling

Force no culling	☐
Enable culling delay	☐
Enable selective culling delay	☐
Selective culling delay WS volume lir	25.000
Selective culling delay SS size limit	0.200
Num Frames Delay Before Culling	2
Cull by depth buffer	✓
Cull shadows by depth buffer	✓
Cull shadows by cascade intersectior	✓
Cull static shadows	✓
Cull static shadows by depth buffer	☐
Use GPU for culling	✓
Stall for GPU culling results	✓
Cull by motion projection	✓
Render Nth lowest LOD for occluded	-1
Skip occlusion culling for objects inte	✓
Ignore culling for visibility animated	☐
Draw culling objects	☐
Draw queried culling objects	☐
Draw motion vectors	☐
Draw selected culling objects	☐

Recent | Save | Load | Create UI | Rerun Recent | Reset | Resolve

Search modified...

You have 16 tweaks overrides

lastSession.vfs

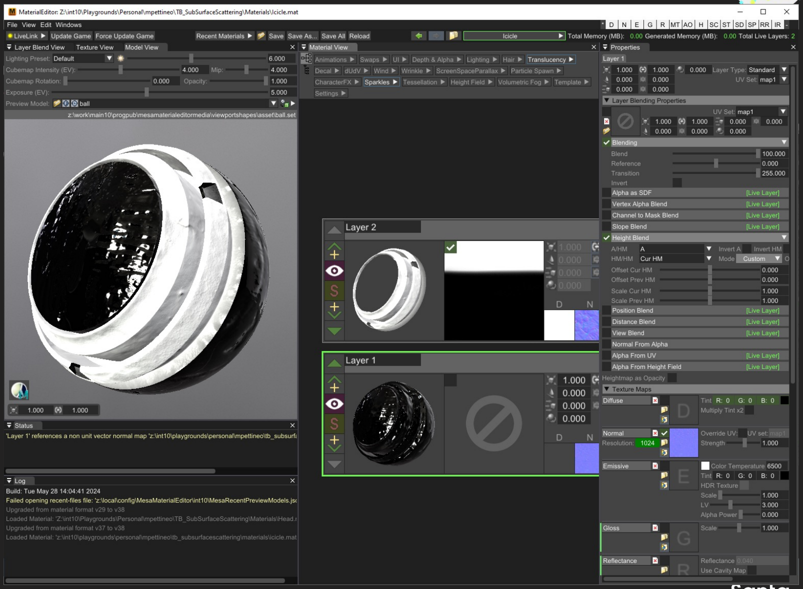
1	/System/Video/Enable Temporal AA	✓
2	/Effects/SSAO/Enable	✓
3	/System/Resource/Lighting/Lights/Enable Sun and Moon Light	✓
4	/System/Resource/Lighting/Show Active Lights	☐
5	/System/Video/Shadows/Enable	✓
6	/System/Resource/Model/Vis Top 10	Hidden
7	/System/Resource/Model/Vis Top 10 View	All
8	/System/Resource/Model/Capture Mesh GPU Times	Exec
9	/System/Performance/Display Combo Profiler	☐
	/System/Performance/Display Scene Stats	Disabled
	/System/Video/Enable Async Compute	✓
	/System/Performance/Display GPU	☐
	/System/Performance/Display GPU Current Time	☐
	/System/Performance/Display GPU Max Time	☐
	/System/Performance/Display GPU Avg Time	✓
	/System/Performance/Print GPU Timings To TTY	Exec
	/System/Performance/Print GPU Timings Tree Depth	3
	/System/Video/Always Clear G-Buffer	☐
	/System/Video/GPUDebugger/Enable GPU debugger	☐
	/System/Video/GPUDebugger/Enable G-Buffer Inspector	☐

Santa Monica Studio

The "Virtual File System" is a core piece of engine technology that has existed since the studio's earliest games. While it used to be used more broadly for core game data, these days it's used for exposing development-only "tweak" variables. VFSTool is an internal tool that runs on the host machine and exposes a simple UI for interacting with these VFS tweaks. It's similar to an ImGui debug menu, except it predates ImGui does not need to run on the console itself. This tool uses LiveLink to query which the VFS directory structure, the type of the tweak, as well as its current value. It also obviously allows changing the value, and communicates when a tweak has been changed from its default value.

LiveLink Clients

- Material Editor
  - Live update material properties and textures
  - More on this later!



```
MaterialEditor: Z:\int10\Playgrounds\Personal\mpettineo\TB_SubSurfaceScattering\Materials\icicle.mat
```

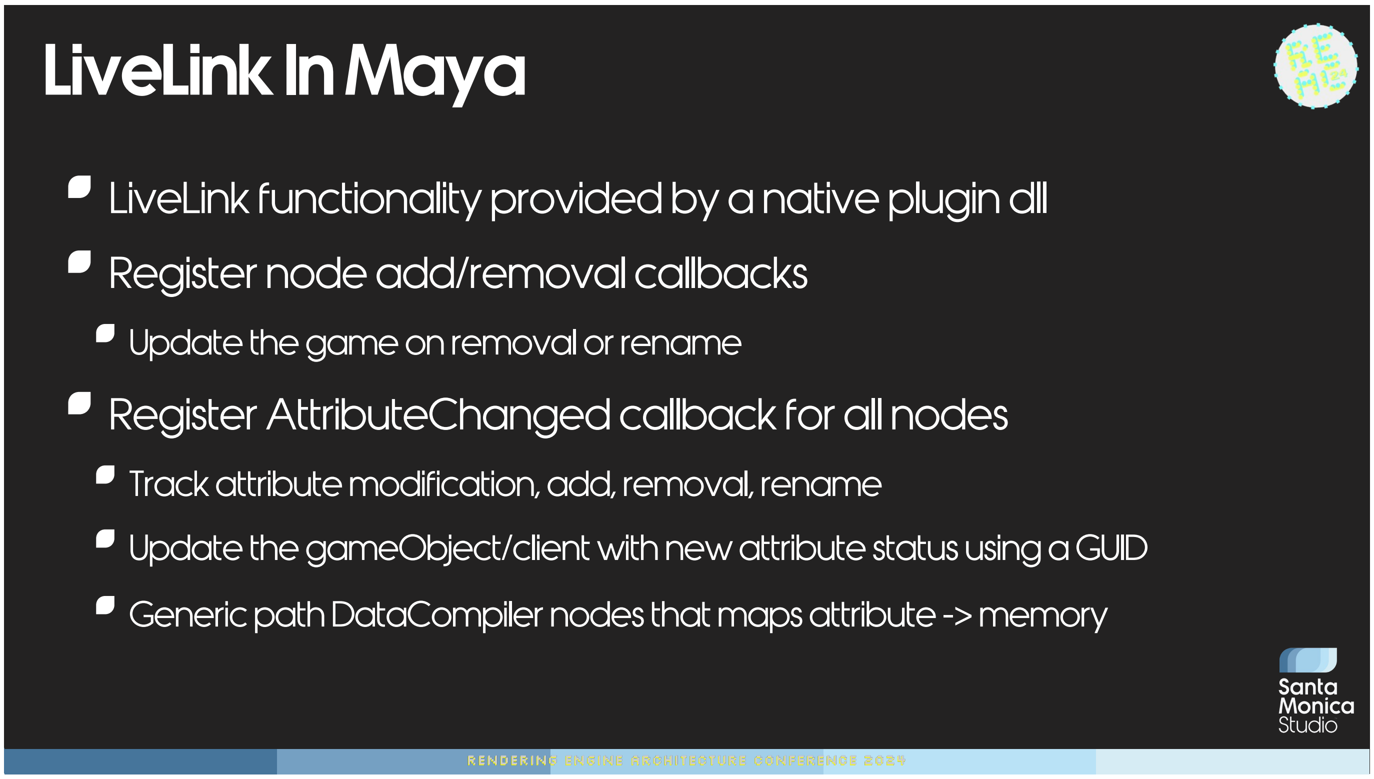
```
Status
'Layer 1' references a non unit vector normal map 'z:\int10\playgrounds\personal\mpettineo\tb_subsurfa
```

```
Log
Build: Tue May 28 14:04:41 2024
Failed opening recent-files file: 'z:\local\config\MesaMaterialEditor\int10\MesaRecentPreviewModels.jsc
Upgraded from material format v29 to v38
Loaded Material: 'Z:\int10\Playgrounds\Personal\mpettineo\TB_SubSurfaceScattering\Materials\Head.r
Upgraded from material format v37 to v38
Loaded Material: 'z:\int10\playgrounds\personal\mpettineo\tb_subsurfacescattering\materials\icicle.mat
```

Santa Monica Studio

RENDERING ENGINE ARCHITECTURE CONFERENCE 2024

Mesa, our internal material editor utilizes LiveLink extensively to allow live editing and preview of materials in the game. This includes modifying bool/float/color properties as well as assigning new textures. I'll be talking more about Mesa in the next section.

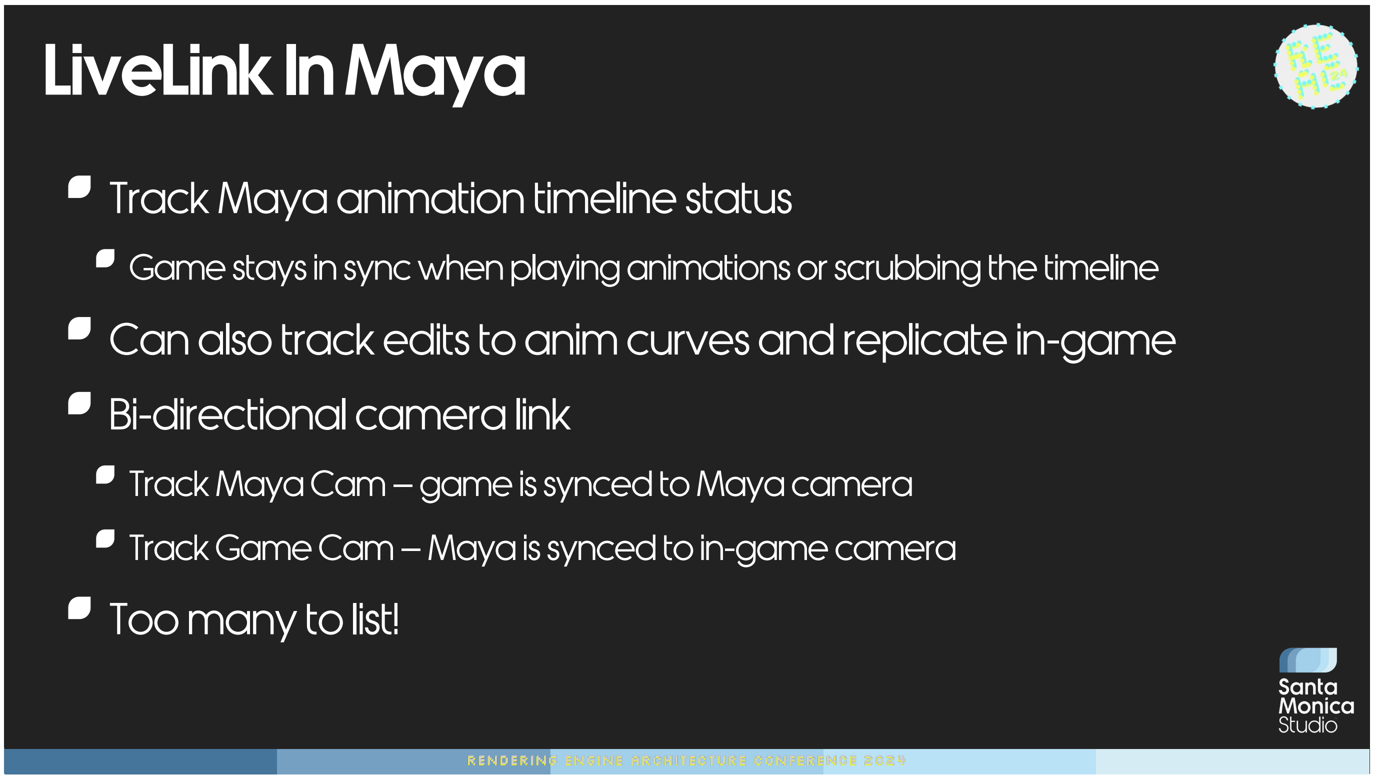


LiveLink In Maya

- LiveLink functionality provided by a native plugin dll
- Register node add/removal callbacks
  - Update the game on removal or rename
- Register AttributeChanged callback for all nodes
  - Track attribute modification, add, removal, rename
  - Update the gameObject/client with new attribute status using a GUID
  - Generic path DataCompiler nodes that maps attribute -> memory

Let's now return to Maya to talk a bit about how live editing is implemented. Maya is perhaps our most important LiveLink client, since it's the main interface for editing the game world. Since Maya is a closed-source third-party tool, we utilize Maya's plugin architecture to extend it with our LiveLink functionality. This plugin works by registering node add and removal callbacks using the Maya API, which allows the plugin to internally keep track of every node in the scene without having to constantly poll and traverse the DAG. Once the plugin is aware of a node, it can then register a callback to have Maya notify it whenever an attribute has been modified, added, removed, or renamed. These attributes encompass built-in maya properties like translation/rotation/scale, as well as custom attributes that our plugins register to represent data that is only meaningful to our engine. Based on the what changed, the plugin can then extract the relevant information from the node + attribute and package it up in a message to send to the game. This typically includes the node GUID as well as the attribute name and value at minimum.

On the game side the gameObject and client ends up getting a virtual method called so it can handle the AttributeChanged message. Some clients have special-case handling, but in many cases we can use a generic path for DataCompiler nodes that simply copies the new attribute value to a location in memory.



LiveLink In Maya

- Track Maya animation timeline status
  - Game stays in sync when playing animations or scrubbing the timeline
- Can also track edits to anim curves and replicate in-game
- Bi-directional camera link
  - Track Maya Cam – game is synced to Maya camera
  - Track Game Cam – Maya is synced to in-game camera
- Too many to list!

Santa Monica Studio

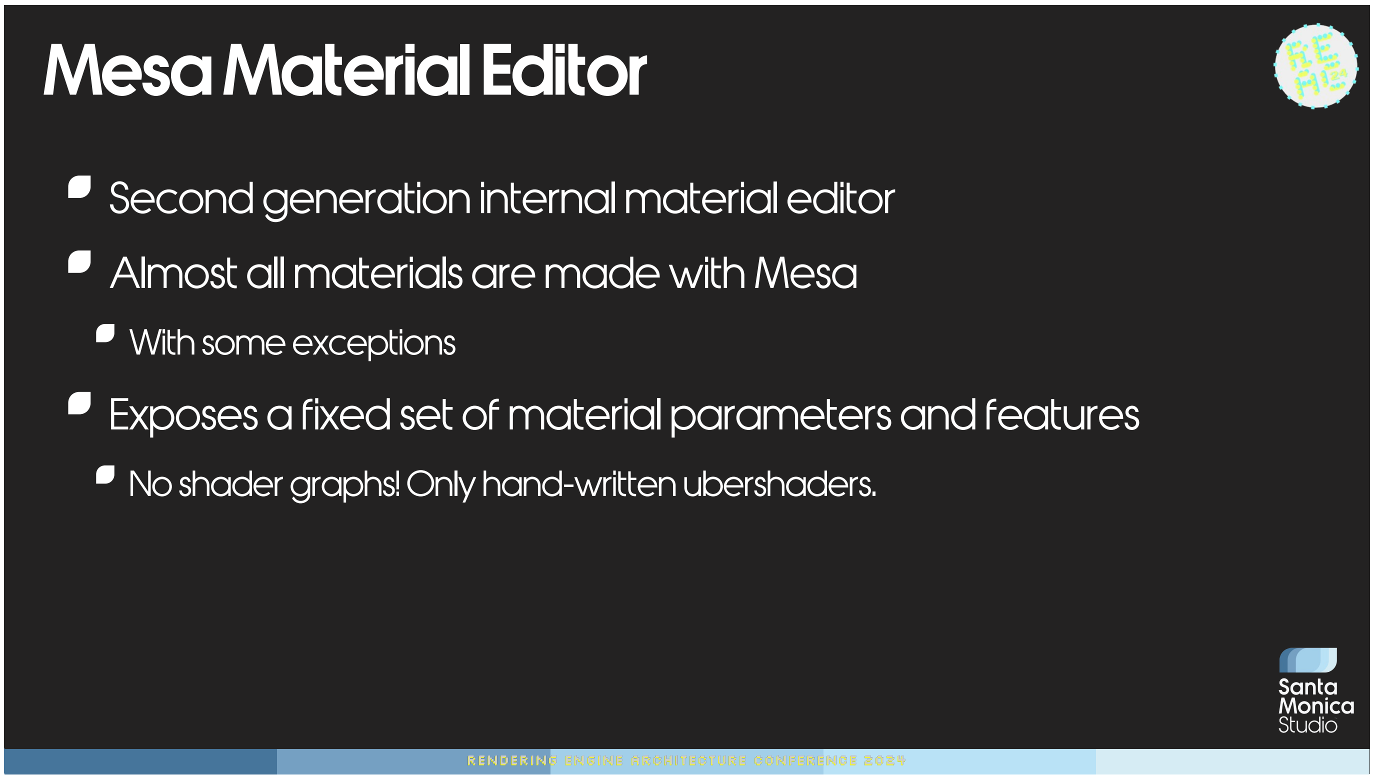
Our LiveLink plugin does more than just track and replicate node attributes. The full list is too large for this presentation, but let me provide a few other examples. Another important LiveLink functionality is syncing the game's animation timeline with Maya's. This allows controlling the animation state directly from Maya, including scrubbing the timeline. In addition to the animation timeline, the plugin can also track edits to animation curves and notify the game so that the changes can be replicated. Finally, we also have a bi-directional camera link between Maya and the game. This allows the game camera to follow Maya's camera, which is useful for live edit scenarios. Or alternatively Maya's camera can be synced to the game's camera, which is convenient to locate where the player is within a larger area.

Mesa Material Editor

How we create and edit materials

RENDERING ENGINE ARCHITECTURE CONFERENCE 2024

Next, I would like to talk about Mesa, which is our in-house material editor.



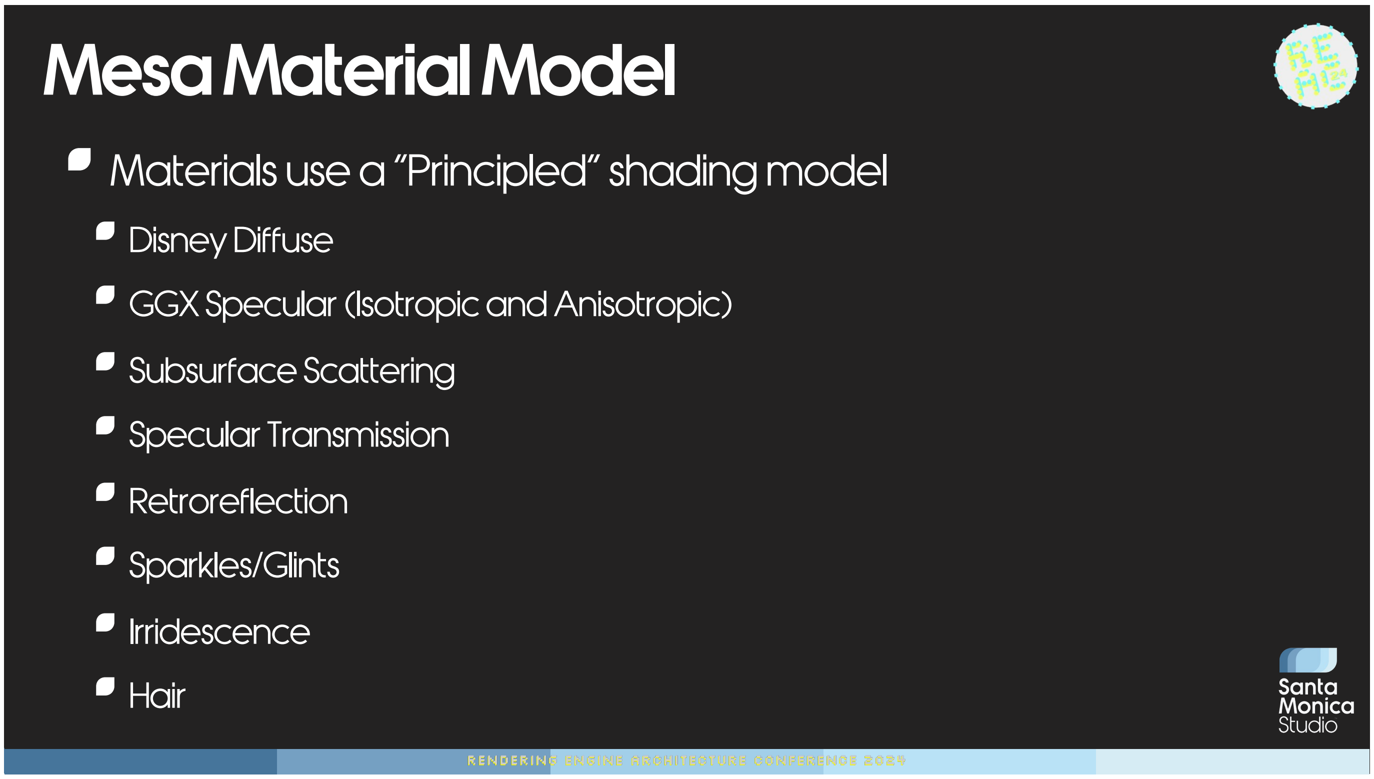
Mesa Material Editor

- Second generation internal material editor
- Almost all materials are made with Mesa
  - With some exceptions
- Exposes a fixed set of material parameters and features
  - No shader graphs! Only hand-written ubershaders.

Santa Monica Studio

Mesa is actually our second-generation material editor, and was written during development God Of War: Ragnarok to replace the material editor that had been used for previous titles. Currently almost all materials are created and edited with Mesa, with the exception of a few legacy use cases that have not been replaced yet.

From a high-level, one thing that's important to note about Mesa is that it exposes a fixed set of controls and parameters for the material that's directly tied to a hand-written ubershaders. In other words, we do not expose a visual shader graph that allows technical artists design their own shader logic that determines the final shading parameters for our core BRDF.



Mesa Material Model

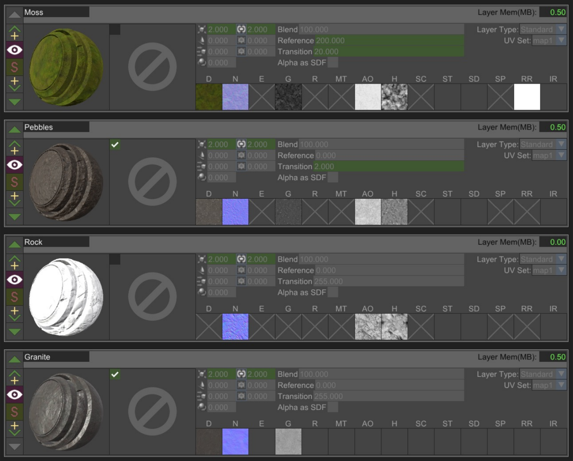
- Materials use a "Principled" shading model
  - Disney Diffuse
  - GGX Specular (Isotropic and Anisotropic)
  - Subsurface Scattering
  - Specular Transmission
  - Retroreflection
  - Sparkles/Glints
  - Irridescence
  - Hair

Santa Monica Studio

The core shading model used by Mesa materials is a principled shading model, similar to what you might see in OpenPBR. What this means is that the shading model has several "features" that correspond to different BxDF terms for diffuse and specular, which can each be enabled and controlled independently with well-defined interactions. These features include Disney Diffuse and isotropic GGX specular as a baseline. But optionally we also support anisotropic GGX, subsurface scattering, specular transmission, retroreflection, sparkles/glints, and iridescence. We also have a hair BxDF based on the Marschner model, which is mostly mutually-exclusive with the other shading features.

Mesa Material Layers

- Organized as a "stack"
  - Similar to Photoshop or Substance Painter
  - Blends parameters together to get the final shading parameters
- Layers can also be "referenced" to allow sharing a layer among several material assets



Moss — Layer Mem(MB): 0.50
2.000 2.000 Blend 100.000 — Layer Type: Standard
0.000 0.000 Reference 200.000 — UV Set: map1
0.000 0.000 Transition 20.000
0.000 Alpha as SDF
D N E G R MT AO H SC ST SD SP RR IR

Pebbles — Layer Mem(MB): 0.50
2.000 2.000 Blend 100.000 — Layer Type: Standard
0.000 0.000 Reference 0.000 — UV Set: map1
0.000 0.000 Transition 2.000
0.000 Alpha as SDF
D N E G R MT AO H SC ST SD SP RR IR

Rock — Layer Mem(MB): 0.00
2.000 2.000 Blend 100.000 — Layer Type: Standard
0.000 0.000 Reference 0.000 — UV Set: map1
0.000 0.000 Transition 255.000
0.000 Alpha as SDF
D N E G R MT AO H SC ST SD SP RR IR

Granite — Layer Mem(MB): 0.50
2.000 2.000 Blend 100.000 — Layer Type: Standard
0.000 0.000 Reference 0.000 — UV Set: map1
0.000 0.000 Transition 255.000
0.000 Alpha as SDF
D N E G R MT AO H SC ST SD SP RR IR

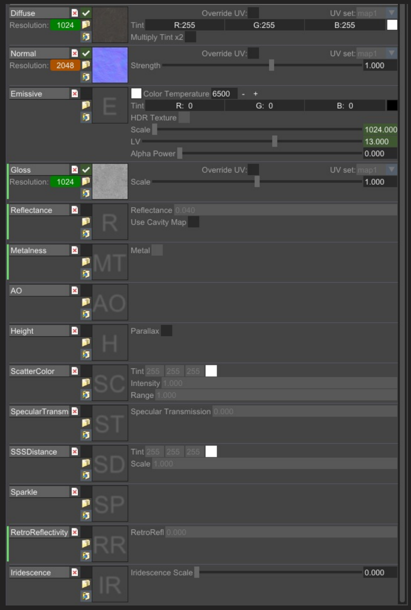
Santa Monica Studio

Internally, Mesa materials are built around a layer stack. This stack is similar to what you would see in Photoshop or Substance Painter, where the stack defines an implied order of how the layers are blended together. The result of blending these layers is ultimately the surface and shading parameters that feed into our principled shading model. This is effectively "horizontal" layering, since we're just blending parameters. We don't support any kind of arbitrary "vertical" layering, which would require evaluating multiple interactions as light scatters into and out of the physical layers.

In many cases a single material is effectively a meta-material formed by combining many individual sub-materials together through layers. For example stone might be one layer, with moss being a secondary layer blended on top. To facilitate this, we support "layer referencing". This allows artists to create a library of base materials, which can then be used as individual layers. This avoids having to replicate material properties for each potential layer combination.

Mesa Material Layers

- A uniform set of channels is exposed per-layer
  - Secondary layers opt-in to channels they want to modify
- A layer UV transform is shared among all channels
  - Some channels have an optional UV override



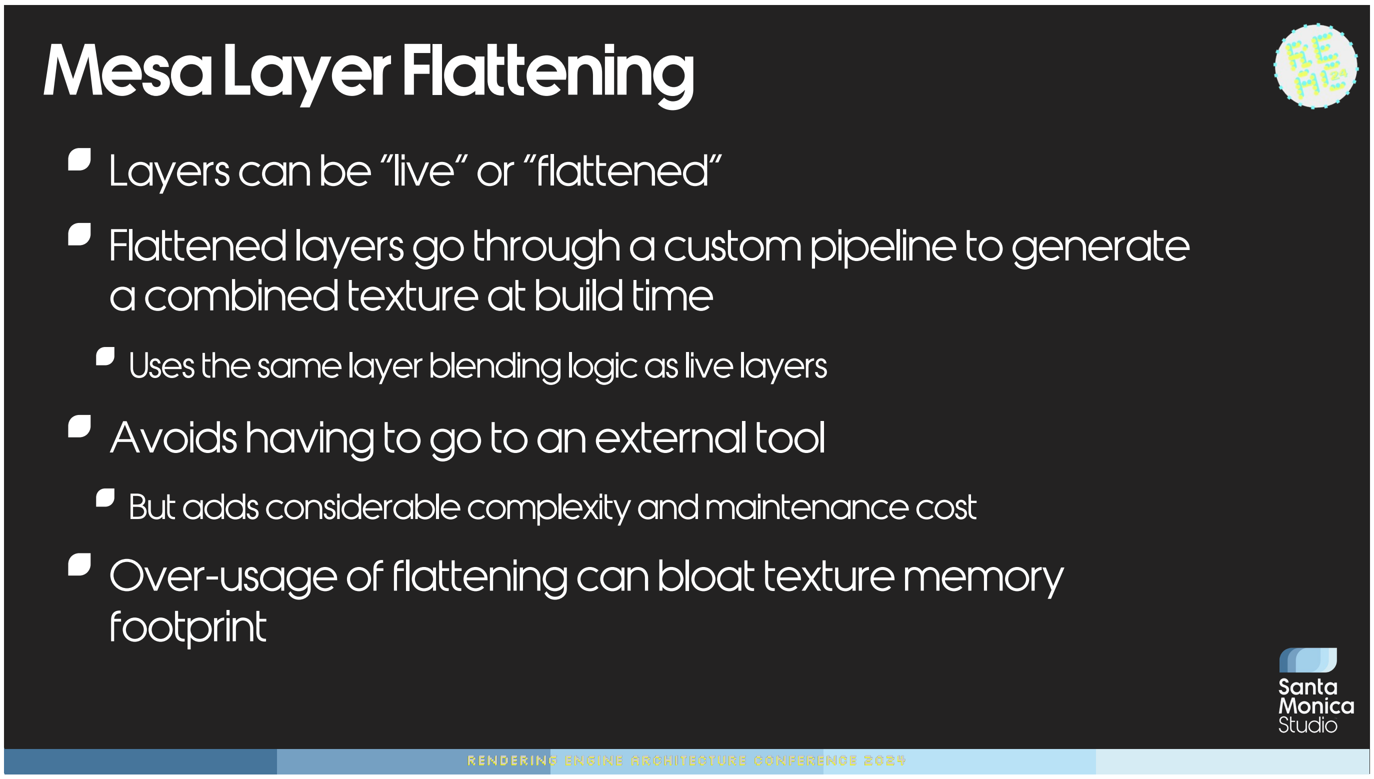
```
Diffuse              Override UV:                         UV set: map1
Resolution: 1024     Tint  R:255   G:255   B:255
                     Multiply Tint x2
Normal               Override UV:                         UV set: map1
Resolution: 2048     Strength                             1.000
Emissive             Color Temperature 6500  -  +
                     Tint  R: 0   G: 0   B: 0
                     HDR Texture
                     Scale                                1024.000
                     LV                                   13.000
                     Alpha Power                          0.000
Gloss                Override UV:                         UV set: map1
Resolution: 1024     Scale                                1.000
Reflectance          Reflectance 0.040
                     Use Cavity Map
Metalness            Metal
AO
Height               Parallax
ScatterColor         Tint 255 255 255
                     Intensity 1.000
                     Range 1.000
SpecularTransmi      Specular Transmission 0.000
SSSDistance          Tint 255 255 255
                     Scale 1.000
Sparkle
RetroReflectivity    RetroRefl 0.000
Iridescence          Iridescence Scale                    0.000
```

Santa Monica Studio

Each layer exposes a uniform set of “channels. Each channel is typically a combination of a texture map with a set of parameters that combine with that texture. The channels then map to some major component of our shading model. For example we have base color, normal, emissive, gloss, AO, scattering distance, and iridescence as separate channels.

For multi-layer materials, the secondary layers are setup so that each channel must “opt in” to participating in the blend. This allows layers to only override a subset of the channels rather than affecting all of them.

Each layer also has a set of UV transform properties that affect all channels in the layer. This includes simple operations like scale/rotation/translation/scroll, as well as more complex transformations such as parallax mapping.



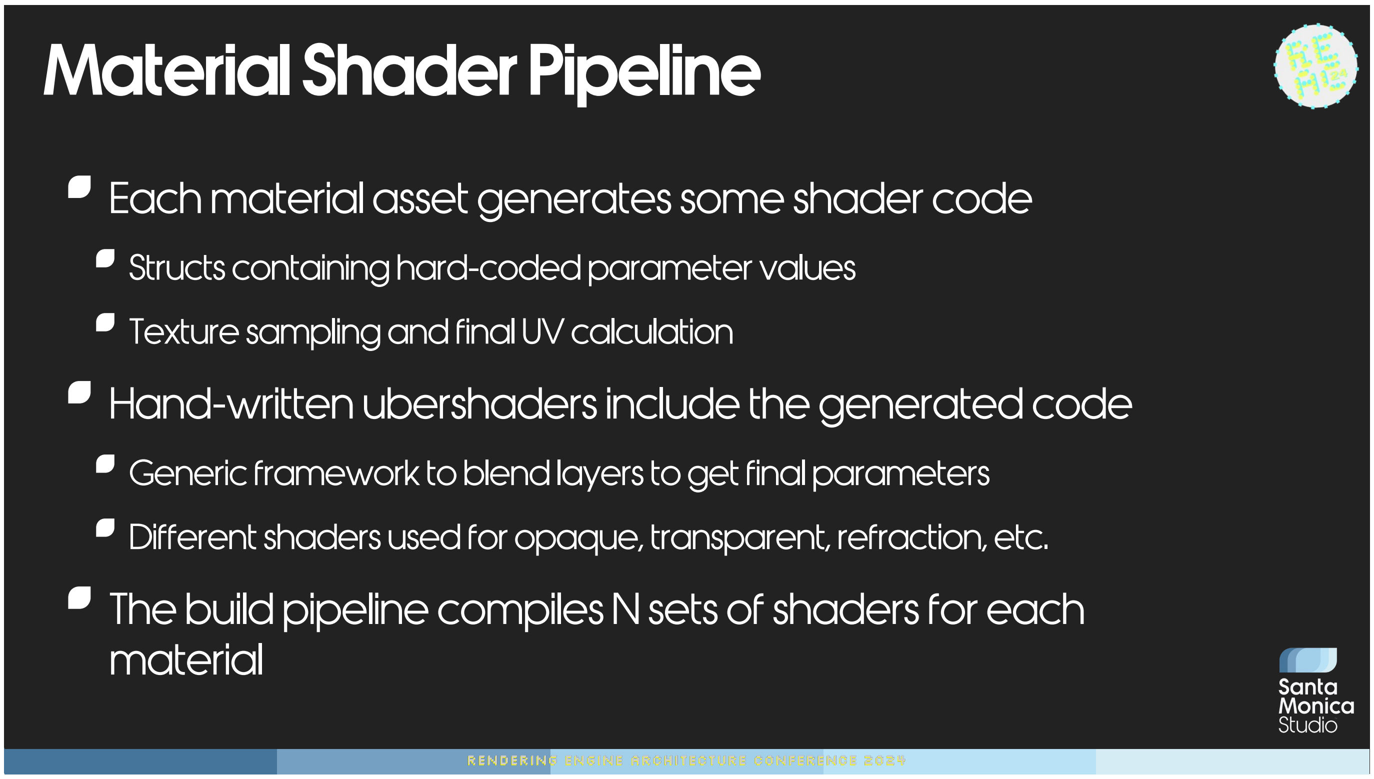
Mesa Layer Flattening

- Layers can be "live" or "flattened"
- Flattened layers go through a custom pipeline to generate a combined texture at build time
  - Uses the same layer blending logic as live layers
- Avoids having to go to an external tool
  - But adds considerable complexity and maintenance cost
- Over-usage of flattening can bloat texture memory footprint

Santa Monica Studio

RENDERING ENGINE ARCHITECTURE CONFERENCE 2024

In Mesa, each layer can be either "live" or "flattened". For live layers we evaluate the layer at runtime in the pixel shader, and then blend the results. For flattened layers we flatten that portion of the stack offline to produce a set of pre-blended textures. This happens during a step on our content build pipeline, and ultimately uses the same logic as our live layer blending so that the expected results are produced. The advantage of this approach is that it's fully built into our material editor and build pipeline, and avoids the need to have to go into an external tool such as Substance Designer to produce textures (although I should point out that we still use Substance Designer quite a bit internally). However it is also a significant trade-off, since flattening adds considerable complexity and maintenance cost for the rendering team. Another thing to point out is that over-usage of flattening can also cause issues for texture memory and disk footprint. Flattening is often viewed as "free" compared to live layers since the results are precomputed, but that can sometimes lead to flattening being used for relatively simple cases. For example, there might be N variations of a material where the base color is just tinted differently.



Material Shader Pipeline

- Each material asset generates some shader code
  - Structs containing hard-coded parameter values
  - Texture sampling and final UV calculation
- Hand-written ubershaders include the generated code
  - Generic framework to blend layers to get final parameters
  - Different shaders used for opaque, transparent, refraction, etc.
- The build pipeline compiles N sets of shaders for each material

Santa Monica Studio

RENDERING ENGINE ARCHITECTURE CONFERENCE 2024

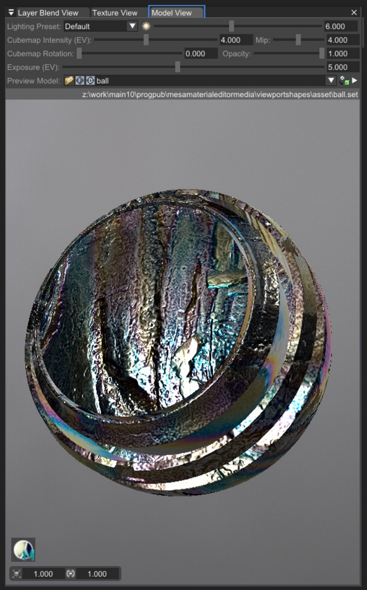
In terms of how materials map to actual compiled shaders, each material generates a block of shader code that corresponds to the exact options and parameters configured in the material. The bulk of this generated code takes the form of structs that contain hard-coded parameter values for things like tint colors and flags. The remainder is mostly comprised of code to sample the assigned textures for a particular channel. This part is also responsible for applying any requested UV transformations before sampling the texture.

This generated code then ends up being included by our hand-written ubershaders that form the overall structure of the shaders. We have a generic set of functions and utilities that can use the generated code to calculate the layer alpha, evaluate the layer channels and then blend the channels based on the alpha value. In practice we have a set of these ubershaders for different purposes, for example for rendering opaques to a G-Buffer or rendering transparents with forward lighting.

Ultimately our build pipeline ends up generating and compiling N sets of shaders for each material, where each set has a vertex shader + pixel shader at minimum. Most of these sets are tied to a particular variant that might be requested by the mesh using the material, for example skinning vs. non-skinned. However in some cases we have special rendering paths for different material types that result in multiple passes, which can result in additional shader sets being compiled.

Mesa Material Preview

- Mesa runs bespoke DX11 shaders to display a real-time preview
- Includes some of the actual in-game shader code for lighting/BRDF evaluation
- Uses a simplified lighting environment
  - Single environment probe + directional light
- Currently requires shared shader code to compile in DX11/FXC

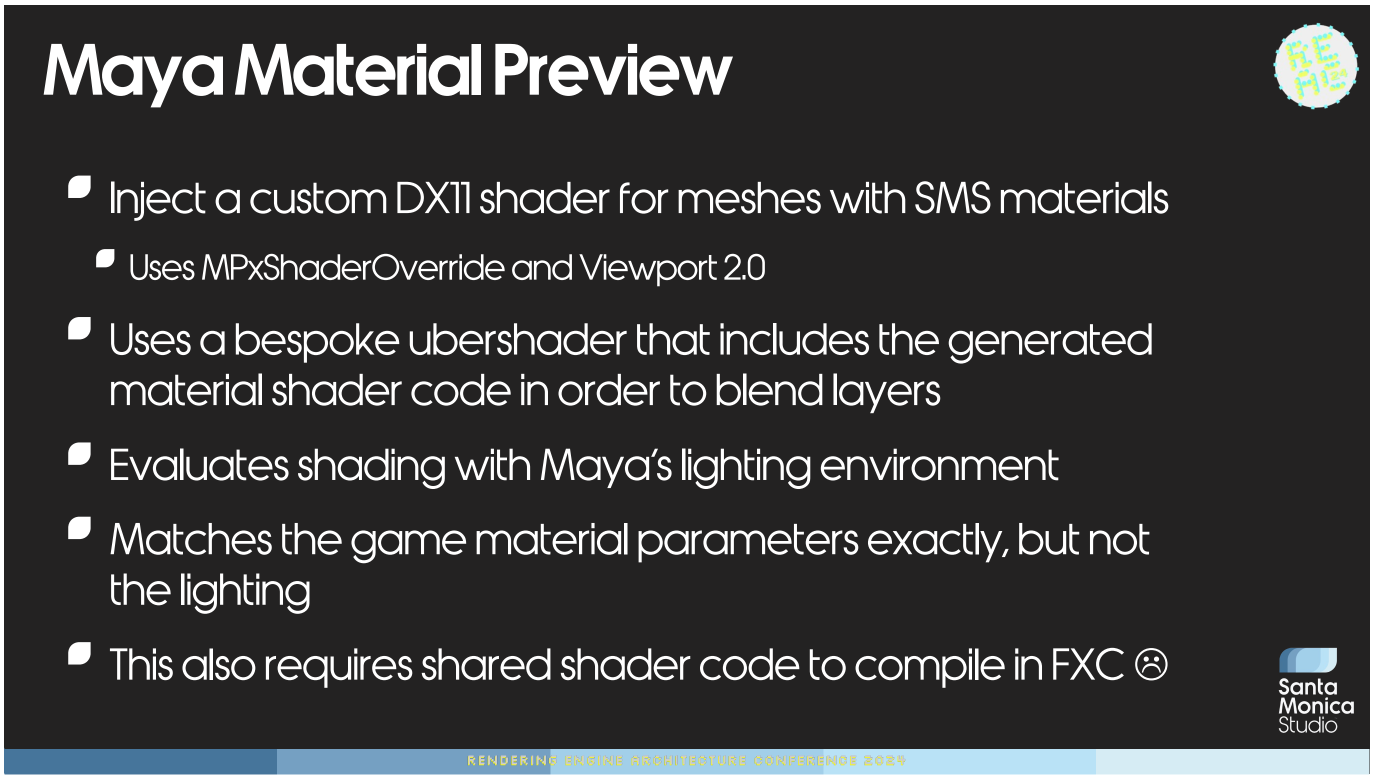


Lighting Preset: Default 6.000
Cubemap Intensity (EV): 4.000 Mip: 4.000
Cubemap Rotation: 0.000 Opacity: 1.000
Exposure (EV): 5.000
Preview Model: ball
z:\work\main10\progpub\mesamaterialeditormedia\viewportshapes\asset\ball.set
1.000 1.000

Santa Monica Studio

Mesa does also feature a real-time live preview that's rendered on the host PC. This preview image is generated using a set of bespoke DirectX11 shaders, which include some of the game's actual actual shader code to perform the same BRDF evaluation. However this is ultimately done in a very simplified forward lighting environment, since the Mesa codebase does not have nearly any of the same rendering features that the game has. Instead Mesa just uses a simple lighting environment with a pre-convolved HDR lighting probe and an analytical directional light. For many shading features this is sufficient to get a decent approximation of what the material will look like in-game. But some of the more advanced ones, like subsurface scattering, aren't replicated because they would require additional passes.

Another thing to point out about this approach is that it requires a subset of our shader code to be directly included in the DX11 shaders, which means they must compile under FXC's very limited feature set.



Maya Material Preview

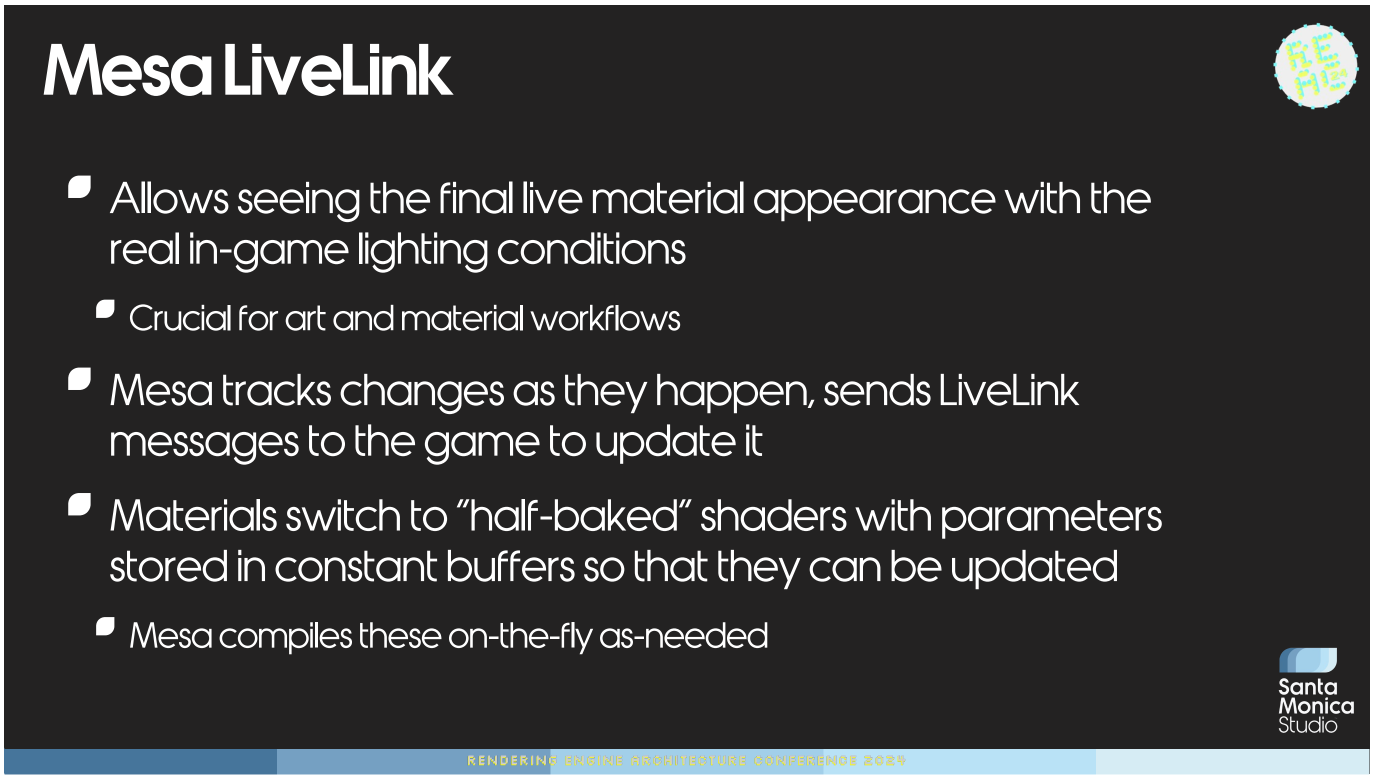
- Inject a custom DX11 shader for meshes with SMS materials
  - Uses MPxShaderOverride and Viewport 2.0
- Uses a bespoke ubershader that includes the generated material shader code in order to blend layers
- Evaluates shading with Maya's lighting environment
- Matches the game material parameters exactly, but not the lighting
- This also requires shared shader code to compile in FXC ☹

Santa Monica Studio

As a quick aside, I also want to briefly cover how our materials are represented in Maya. For Maya we have an additional plugin that interacts with Viewport 2.0 to override the shader and textures used when drawing a mesh. This allows us to provide our own shaders that can be used which can approximate the look of the mesh in the actual game.

The way we do this is by having an additional base ubershader that includes the generated material shader code, just like our in-game shaders. However in this case the shader uses Maya's mesh and lighting inputs to calculate the final surface parameters, and then compute the final lighting from those parameters. What this means is that the shading parameters inside of the shader actually match the game very closely, but the actual lighting environment is vastly different. And just like the Mesa preview, we lack external passes such as subsurface scattering or decals which causes the appearance to diverge even further.

It's also worth pointing out that this is yet another place that requires a subset of shader code to compile under DX11 and FXC, which severely limits the shader language features that we can use.

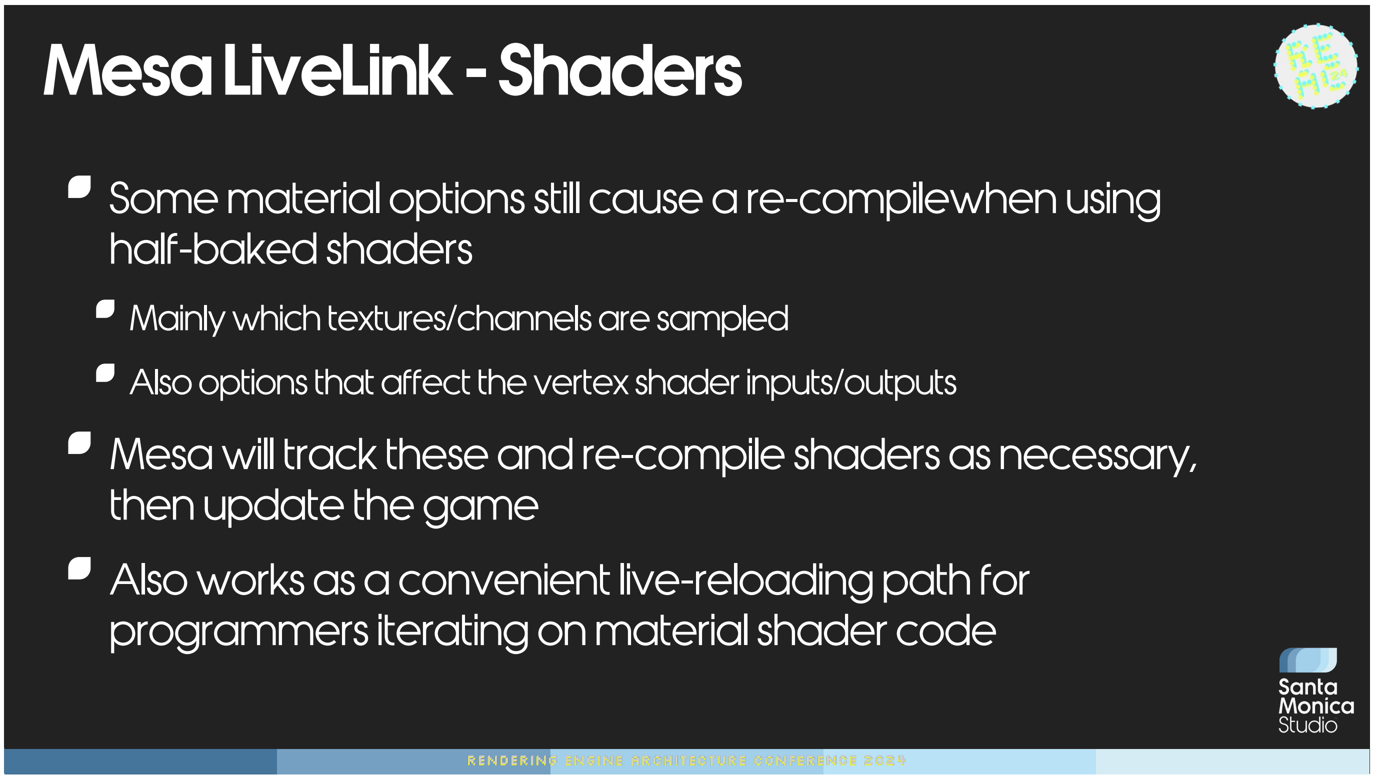


Mesa LiveLink

- Allows seeing the final live material appearance with the real in-game lighting conditions
  - Crucial for art and material workflows
- Mesa tracks changes as they happen, sends LiveLink messages to the game to update it
- Materials switch to "half-baked" shaders with parameters stored in constant buffers so that they can be updated
  - Mesa compiles these on-the-fly as-needed

Santa Monica Studio

Returning back to Mesa, I want to finish by talking about how LiveLink works to implement live editing within the game. This workflow is absolutely critical for authoring materials, since it's the best way to see how a material actually responds with the game's full lighting environment and features. The way it work is that Mesa internally tracks changes to the underlying material document as they happen. As changes occur, they are sent to a thread which performs necessary processing to update the game with the detected changes. When the first change is detected, the first step is usually is to on-the-fly compile a "half-baked" variant of the shaders that the game will switch to. This development-only variant replaces hard-coded parameters with constant buffer values, which allows those values to be live updated by the game. Mesa can then form LiveLink messages when parameters are changed in the UI, and the game responds to them by updating the constant buffer.

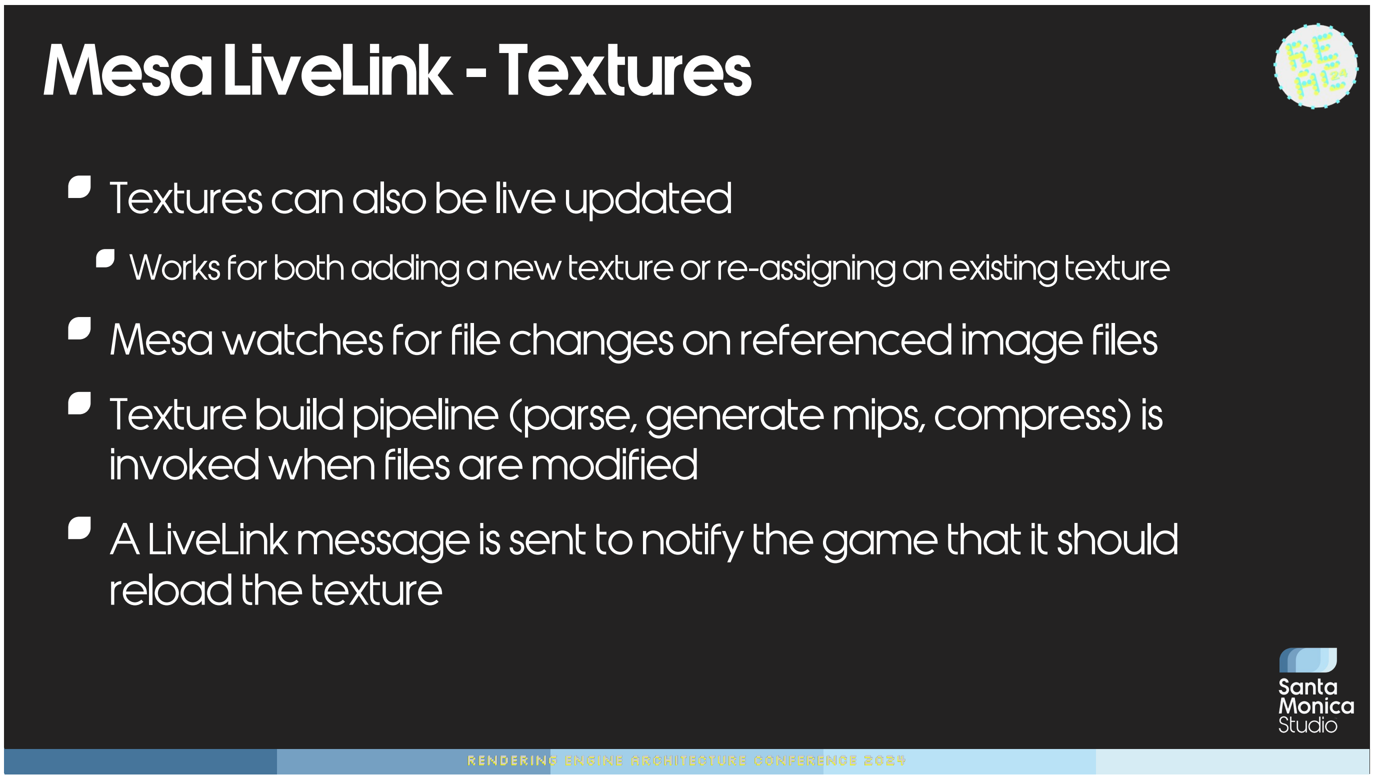


Mesa LiveLink - Shaders

- Some material options still cause a re-compilewhen using half-baked shaders
  - Mainly which textures/channels are sampled
  - Also options that affect the vertex shader inputs/outputs
- Mesa will track these and re-compile shaders as necessary, then update the game
- Also works as a convenient live-reloading path for programmers iterating on material shader code

Some material changes can't be represented with a simple constant buffer update, and will cause Mesa to re-compile the half-baked shader and send it over to the game. For example, this will usually happen when textures are assigned to a channel which causes the channel to now be enabled. It also happens when features require a particular preprocessor macro to change. In all of these cases Mesa will figure out which changes require a shader re-compile and kick off compilation jobs on-demand.

One side benefit of this system is that it can serve has a handy live-reloading path for when programmers iterate on material shader code. Since the ubershader code is shared by many materials, changing the shared code would result in re-compiling many shaders. As an alternative, programmers can instead instruct Mesa to reload a single material being edited which causes the shaders to be re-compiled and re-loaded.

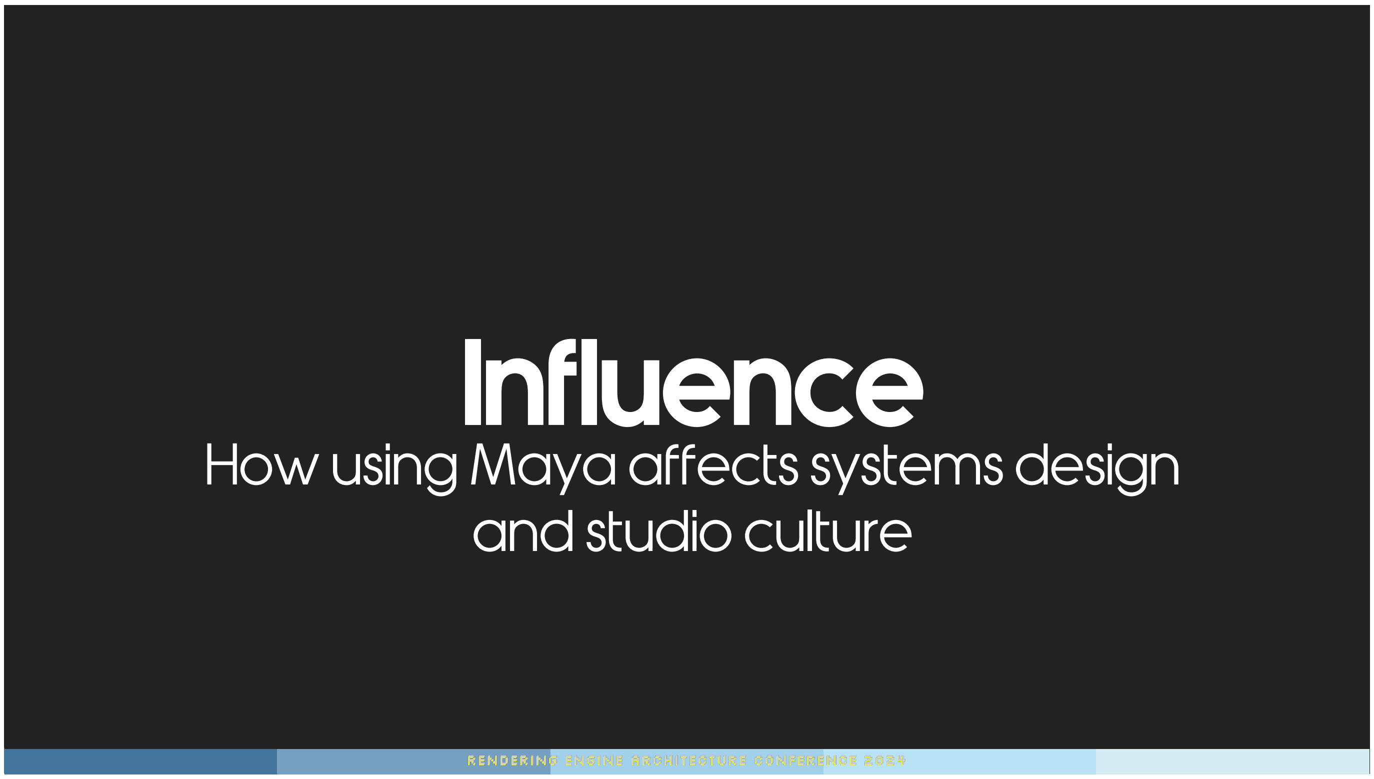


Mesa LiveLink - Textures

- Textures can also be live updated
  - Works for both adding a new texture or re-assigning an existing texture
- Mesa watches for file changes on referenced image files
- Texture build pipeline (parse, generate mips, compress) is invoked when files are modified
- A LiveLink message is sent to notify the game that it should reload the texture

Santa Monica Studio

Finally, Mesa also supports live updating of assigned textures. This includes both adding a new texture to a previously-empty slot, or changing an already-assigned slot to a different texture. Additionally, Mesa will watch for file changes on any referenced textures to determine if they have been modified by an external tool. In all of these cases, Mesa will run the normal texture build pipeline to load the image file, generate mips, and compress it to a BC format. This processed texture file can then be packed up and transmitted to the game in a LiveLink message, so that the game can re-load the texture.



Influence

How using Maya affects systems design and studio culture

Thank you Matt. Hopefully you've seen how Mesa Material Editor has developed and is influenced by our we build our game in Maya. But now let's talk about some other influences on our tech and studio culture.

Examples of Influence

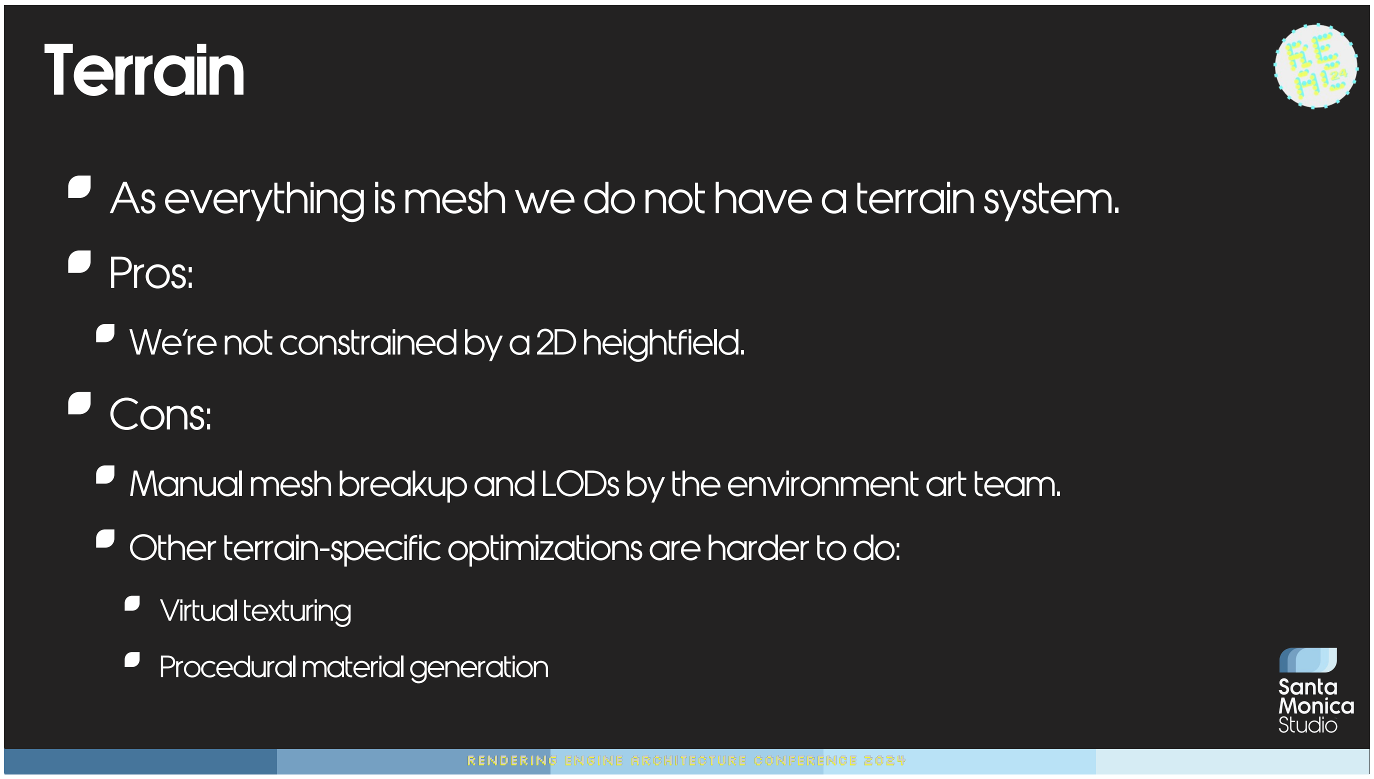
1. Mesh pipeline
2. Animation system
3. Team culture and structure

Santa Monica Studio

Mesh Pipeline

- Everything is hand-authored mesh.
- We have no terrain systems or equivalent.
- This is incredibly powerful:
  - What artists want to build, they can.
  - Extensive uses of skirting.

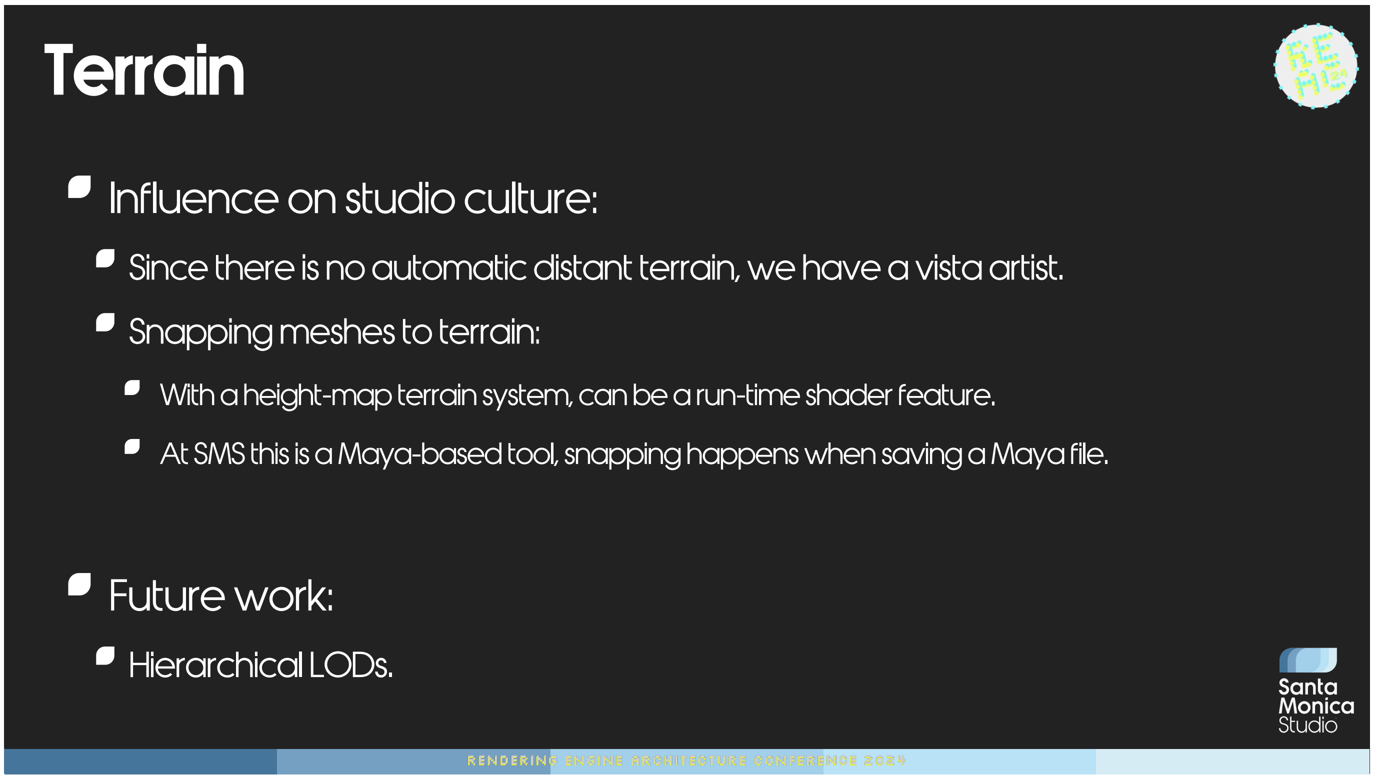
Santa Monica Studio


Terrain

- As everything is mesh we do not have a terrain system.
- Pros:
  - We're not constrained by a 2D heightfield.
- Cons:
  - Manual mesh breakup and LODs by the environment art team.
  - Other terrain-specific optimizations are harder to do:
    - Virtual texturing
    - Procedural material generation

Santa Monica Studio

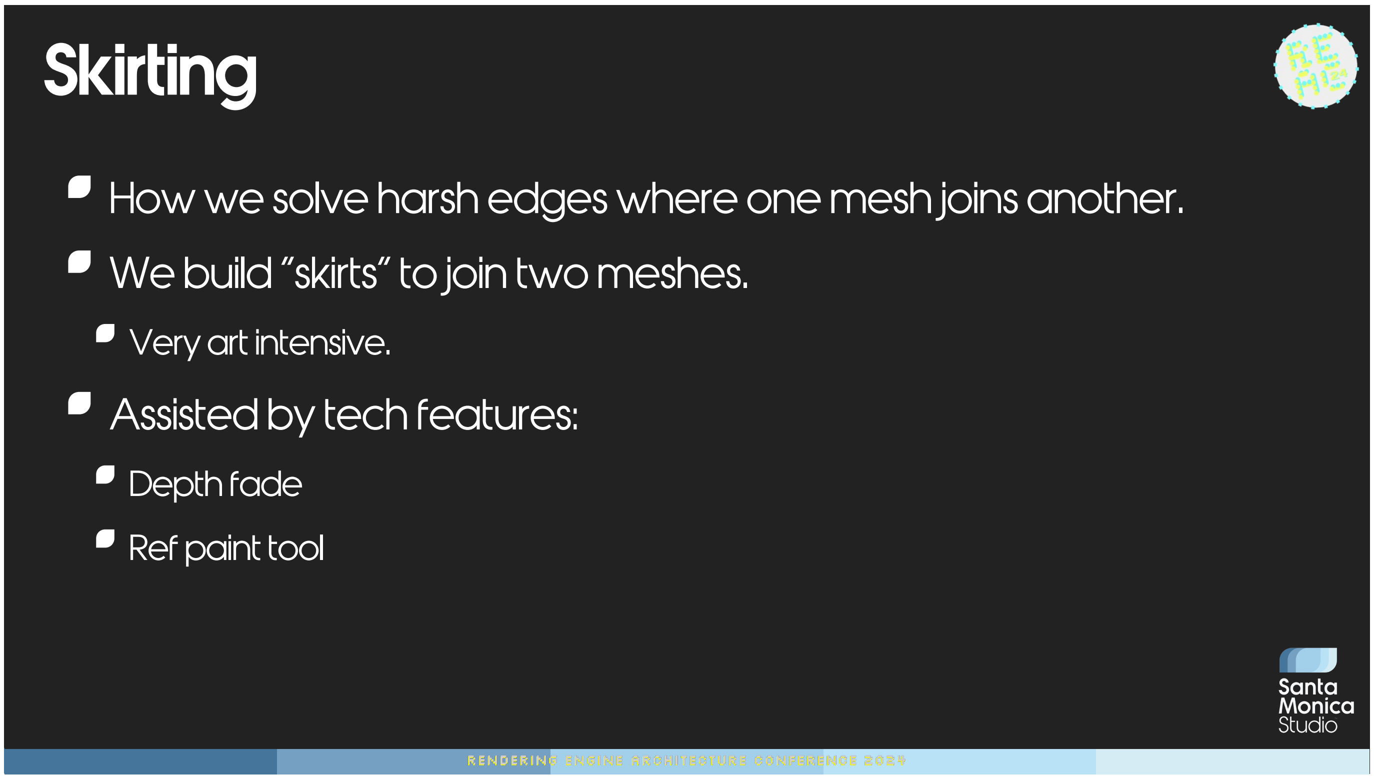
If you don't have any mesh that's obviously terrain, it's harder to write terrain specific optimizations. On previous engines I've worked with, we could do virtual texturing just for terrain. Or implement a custom LOD solution. Or procedurally generate materials. If it's just regular mesh, it's harder to separate that out.



Terrain

- Influence on studio culture:
  - Since there is no automatic distant terrain, we have a vista artist.
  - Snapping meshes to terrain:
    - With a height-map terrain system, can be a run-time shader feature.
    - At SMS this is a Maya-based tool, snapping happens when saving a Maya file.
- Future work:
  - Hierarchical LODs.

Although not having a terrain system has hindered us, I think in the long-term it sets us up really well. For example, when we do implement an automatic distant LOD system, we're going to write a hierarchical LOD system that works for all meshes, rather than just have something for terrain and separate systems for other meshes on top of that.



Skirting

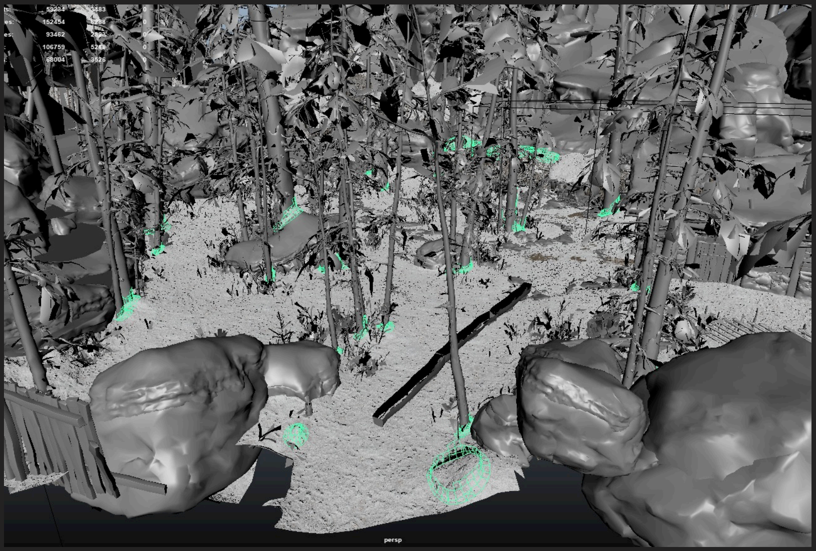
- How we solve harsh edges where one mesh joins another.
- We build "skirts" to join two meshes.
  - Very art intensive.
- Assisted by tech features:
  - Depth fade
  - Ref paint tool

Santa Monica Studio

At other studios I've seen skirting be a huge issue. We've solved it in other ways – if you have a terrain system, we've deformed objects to the terrain height, or blended terrain albedo into objects that sat in the terrain. It's also been a demand by art direction to ramp up SSAO to solve the skirting problem… yes… I know…

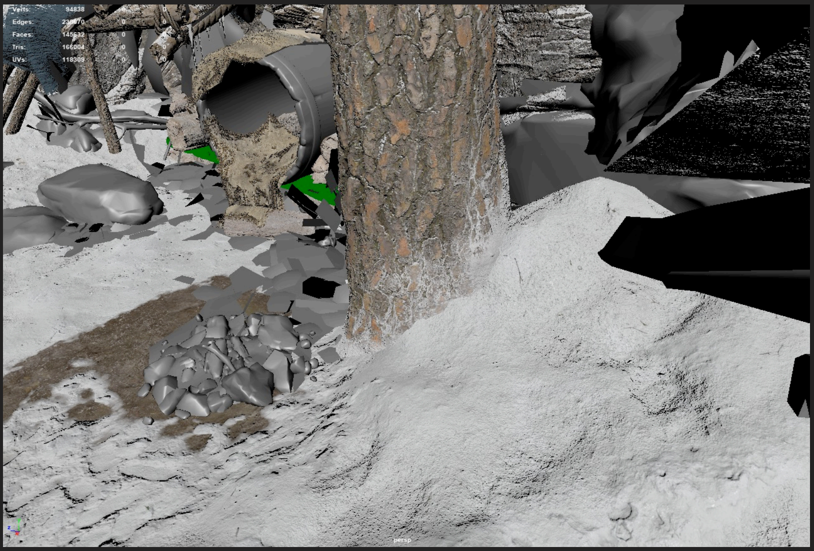
Here, as everything is mesh in Maya, we're able to solve this artistically, by hand placing mesh.

Example (from Midgard)



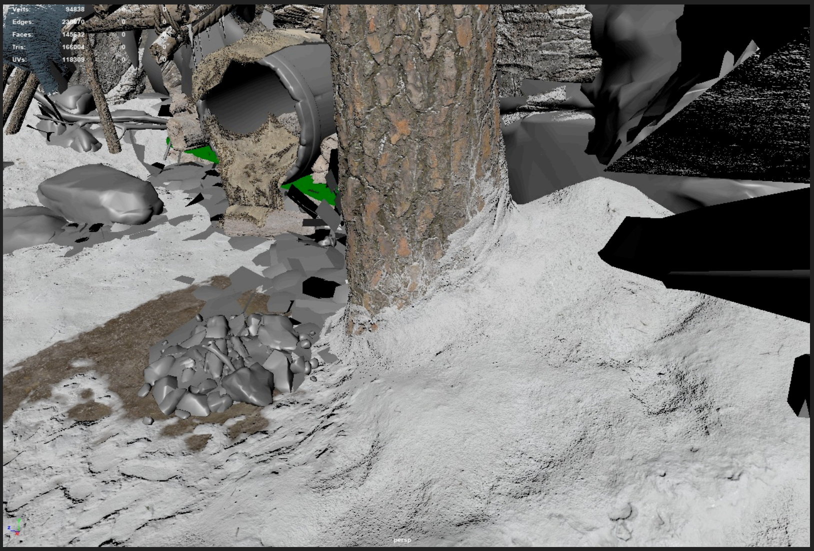
Santa Monica Studio

Tree without Skirt



```
Verts:    94838
Edges:   238670   0
Faces:   145632   0
Tris:    166004   0
UVs:     118309   0
```

Tree with Skirt



```
Verts:     94838    0
Edges:    238670    0
Faces:    145632    0
Tris:     166004    0
UVs:      118309    0
```

Santa Monica Studio

Depth Fade

- Want meshes to fade where they join other meshes.
  - Either skirt meshes…
  - Or meshes in general, avoiding the need for skirts.
- Brute force solution:
  - Alpha blend meshes
  - Read the depth buffer and use "soft alpha" to fade where the mesh is close to the depth buffer / an existing mesh.

Santa Monica Studio

Depth Fade

- Optimized solution (from Bart Wronski on God of War 2018):
  - Modify the depth of the mesh to fade in a dithered pattern.
  - e.g. if we offset the depth by up to 5cm:
    - The mesh starts fading out when within 5cm of a mesh.
    - The mesh is fully faded out at 0cm from a mesh.
  - Conservatively modify depth to preserve early Z.
  - Fade out the modify depth feature at 50m from the camera.

Santa Monica Studio

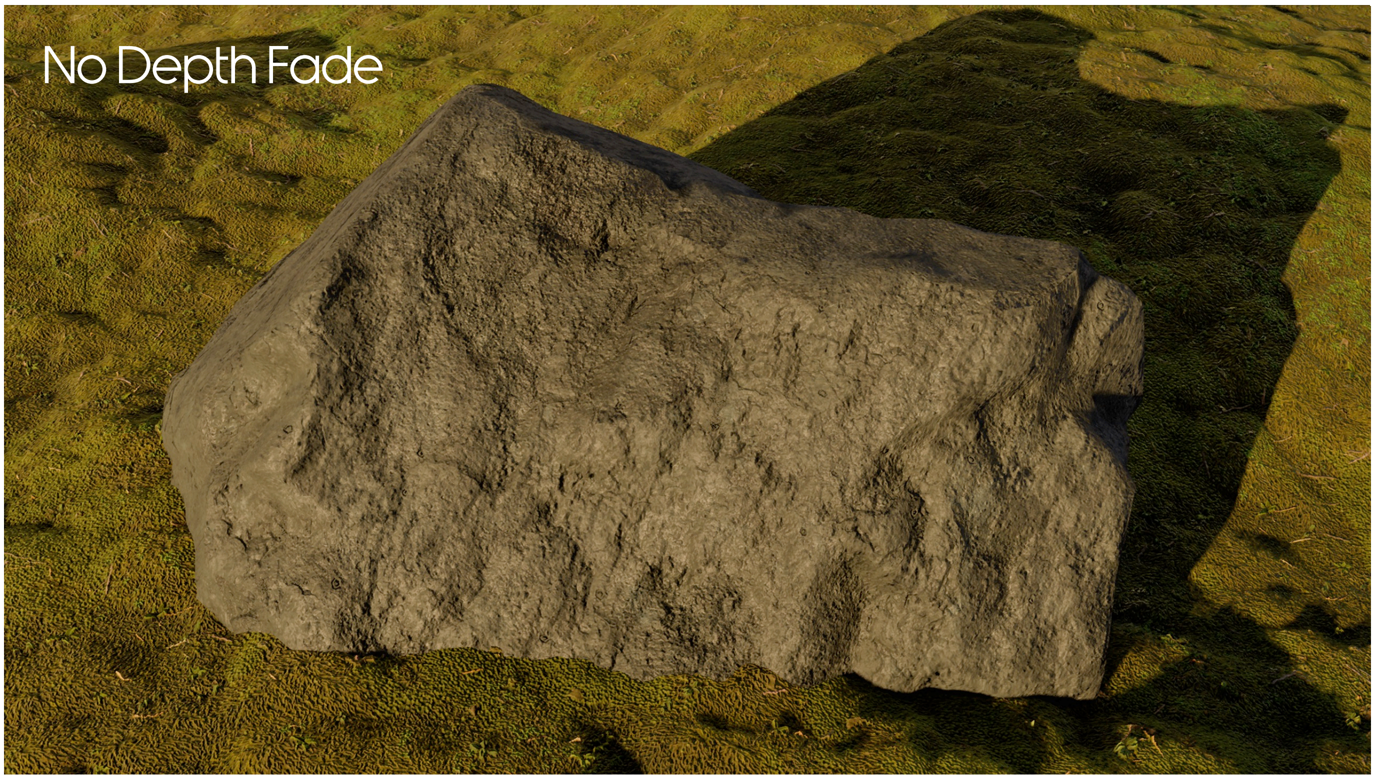
Depth Fade

```
[illegible]
```

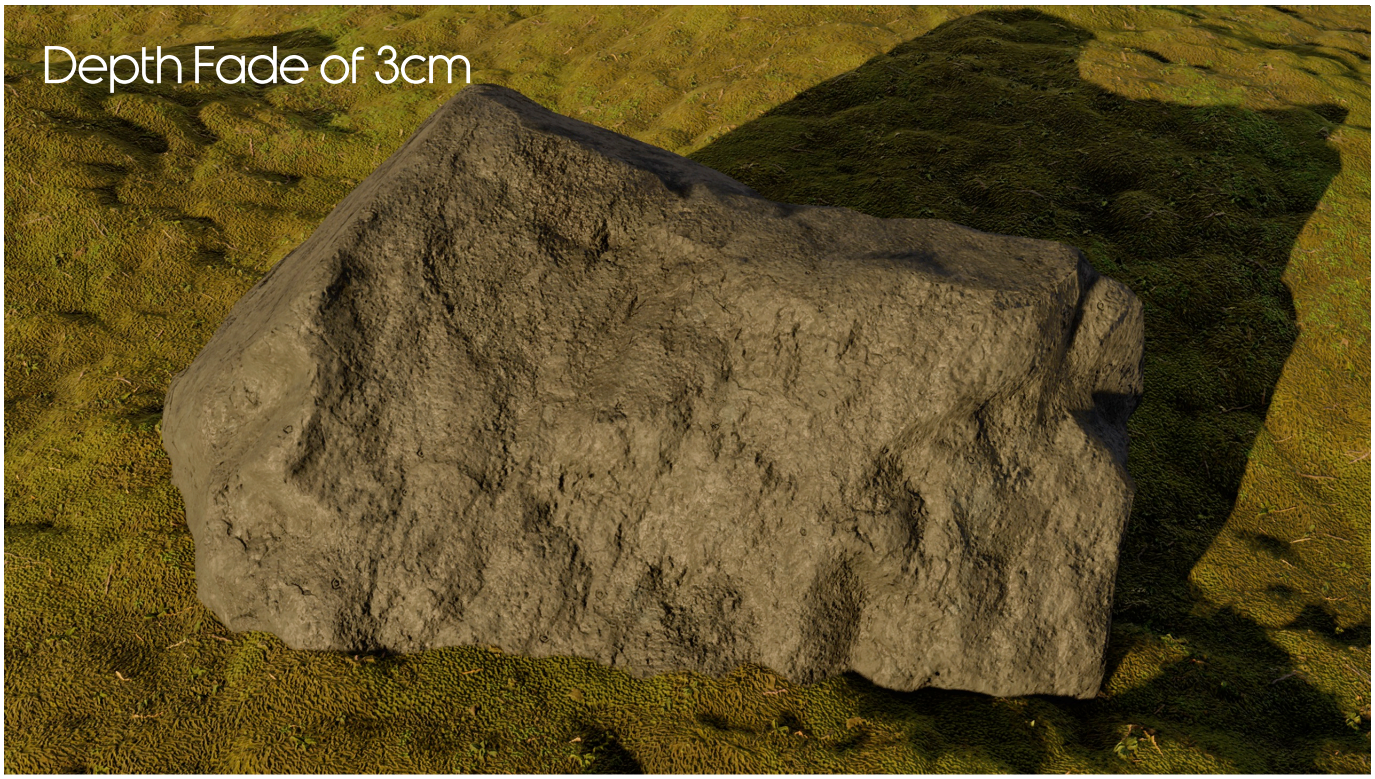
Santa Monica Studio

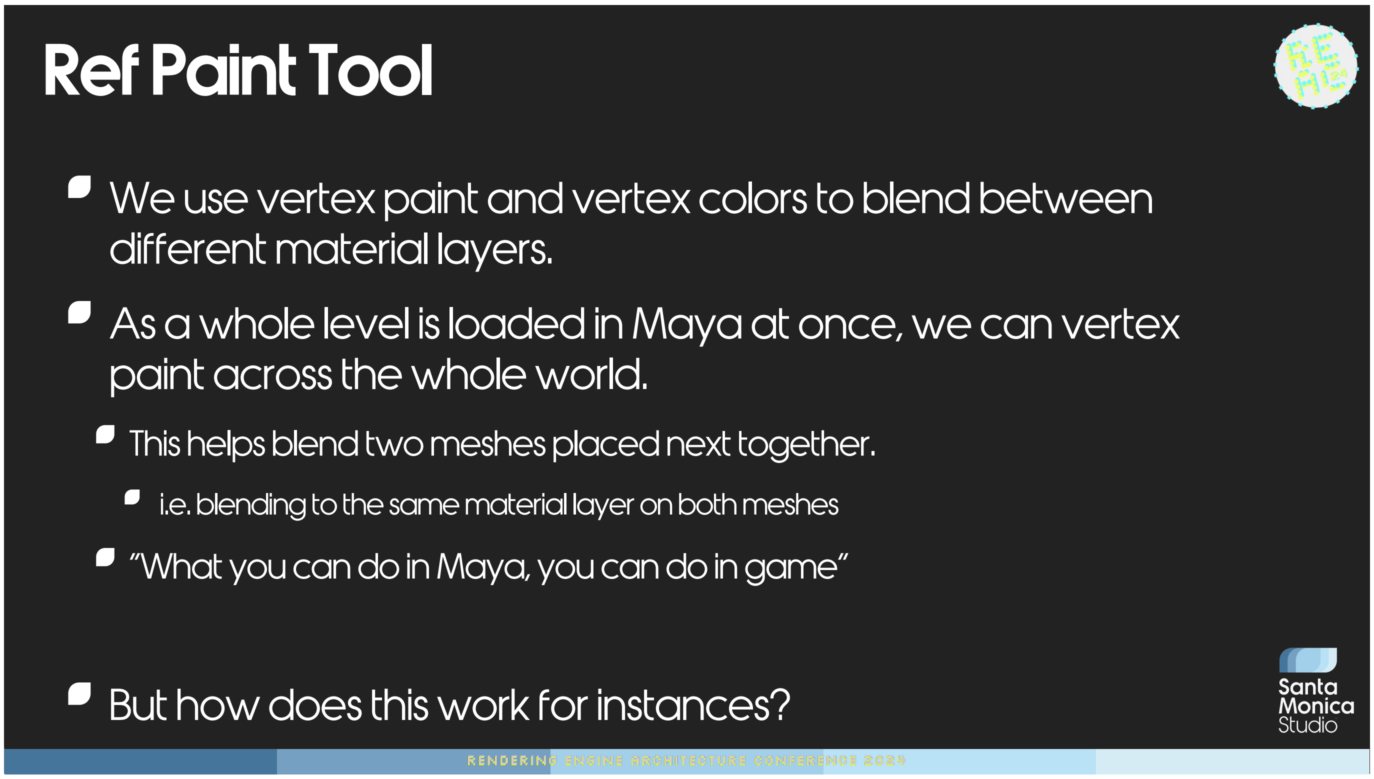
RENDERING ENGINE ARCHITECTURE CONFERENCE 2025

Note: we don’t do this in shadow maps



We often use depth fade to blend rocks against the terrain.


Depth Fade of 3cm



Ref Paint Tool

- We use vertex paint and vertex colors to blend between different material layers.
- As a whole level is loaded in Maya at once, we can vertex paint across the whole world.
  - This helps blend two meshes placed next together.
    - i.e. blending to the same material layer on both meshes
  - "What you can do in Maya, you can do in game"
- But how does this work for instances?

Santa Monica Studio

This is a great example of us supporting a very Maya-based workflow, just painting across the world, but then supporting this in game.

Ref Paint Tool

- Problems with painting across instances:
  - Need a separate data set for each copy of the instance
  - When an instance changes, all vertex paint data is invalidated
    - Everywhere in the world.

Santa Monica Studio

Ref Paint Tool

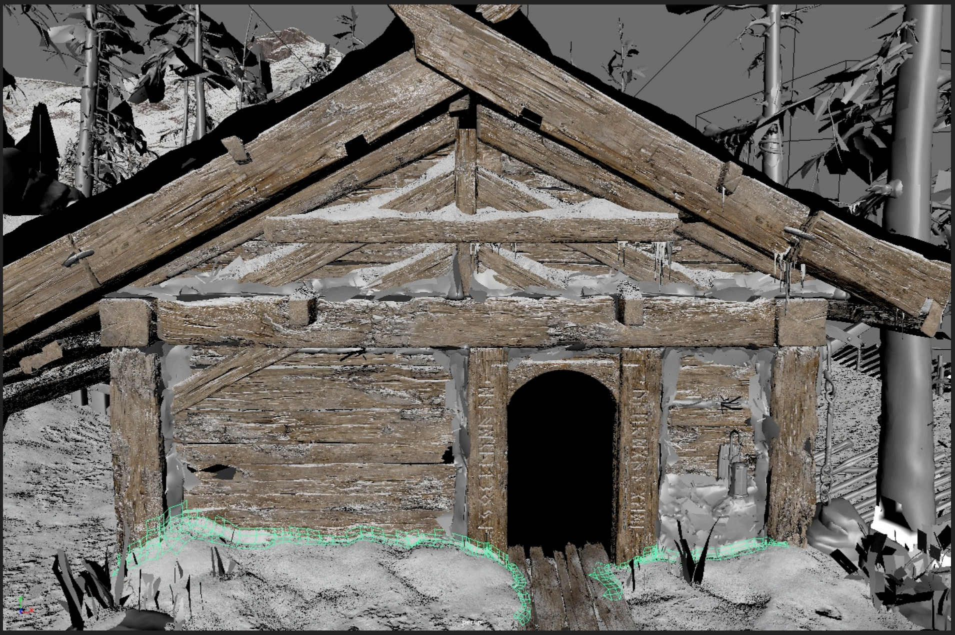
- Vertex colors are stored in a point cloud.
  - If the underlying mesh changes, the painted point cloud remains the same.
- Projected onto each mesh at build time.
  - A copy of the data for each instance.
- Causes build dependency problems:
  - If an instance changes, need to rebuild vertex color data in every place it is used.

Santa Monica Studio

Skirts

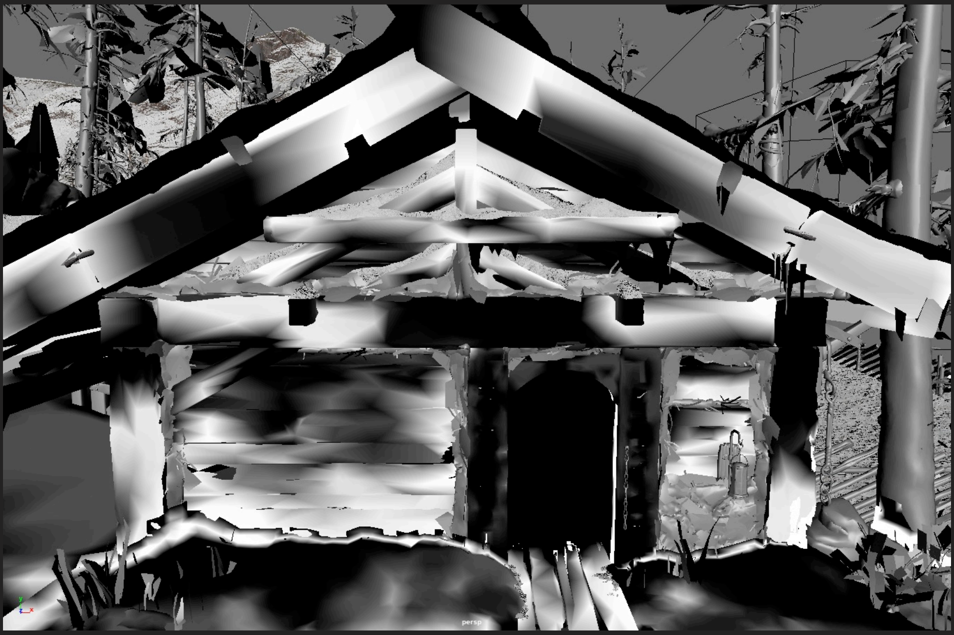
Santa Monica Studio

Geometry

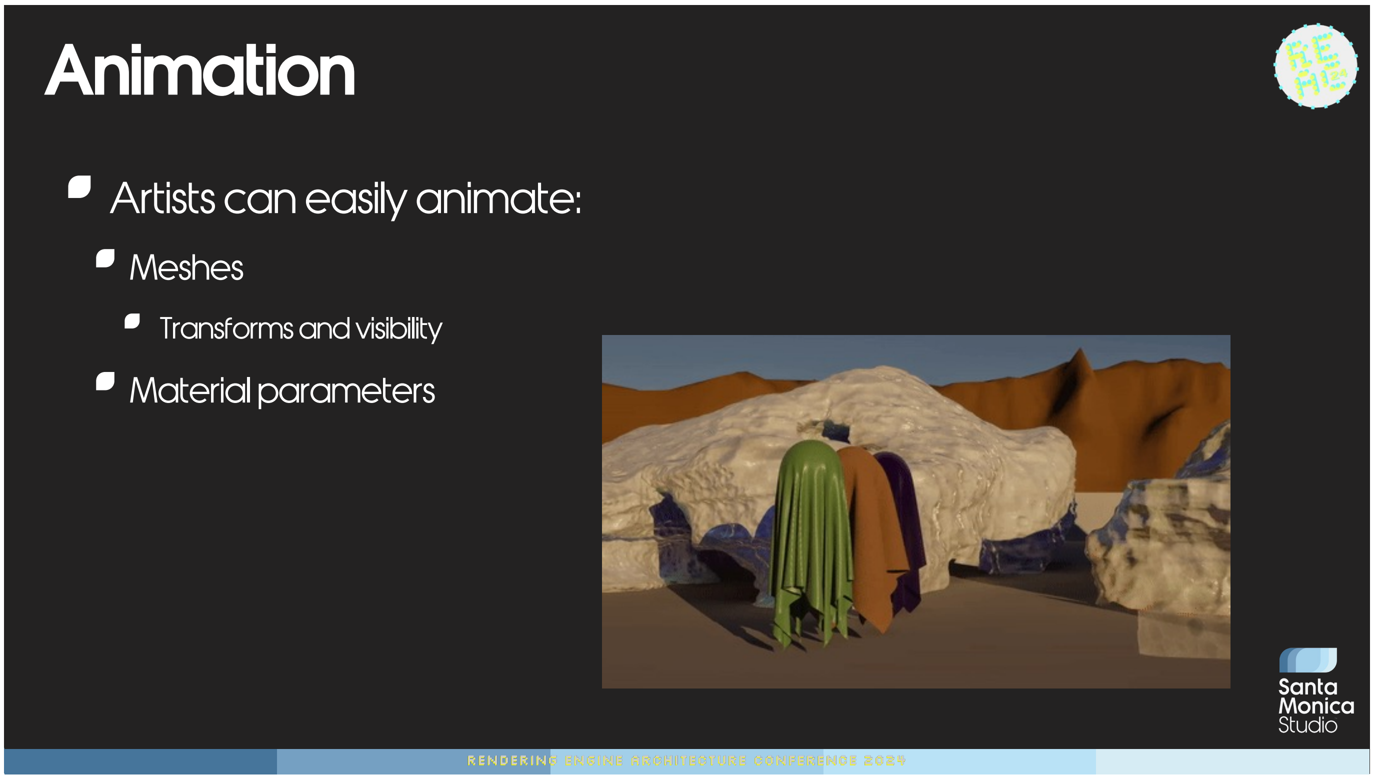


Santa Monica Studio

Vertex Painting



Santa Monica Studio



Animation

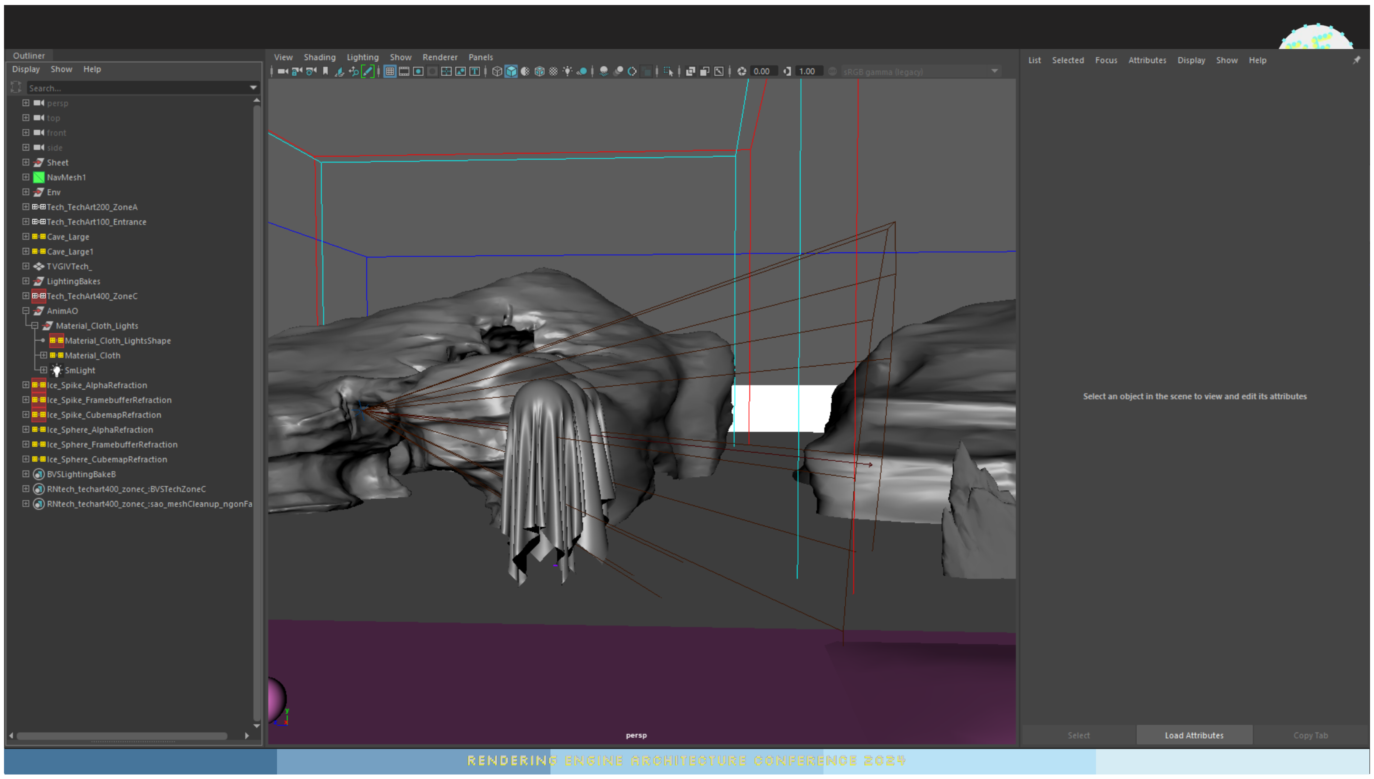
- Artists can easily animate:
  - Meshes
    - Transforms and visibility
  - Material parameters

Santa Monica Studio

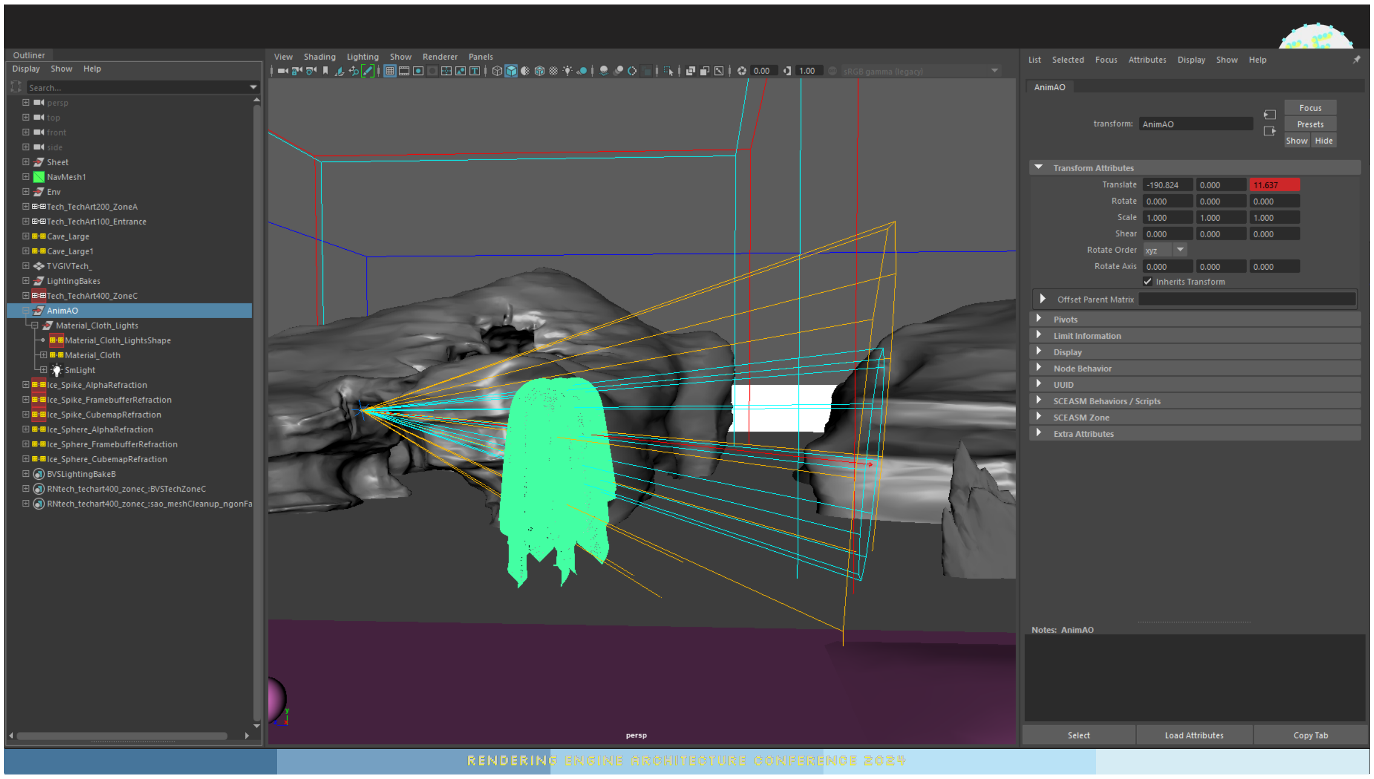
RENDERING ENGINE ARCHITECTURE CONFERENCE 2024

One engine I worked with, it was a feature request the rendering team to have light bulbs swinging. Of course, that's something that should just be done by the content creation team, in engine, but in this particularly engine it was so hard as to be impossible. However, at SMS, adding animations is easy.

This is a simple animated asset we have in a test scene. Let's see how easy it is to add this animation.


Outliner
Display Show Help
Search...
persp
top
front
side
Sheet
NavMesh1
Env
Tech_TechArt200_ZoneA
Tech_TechArt100_Entrance
Cave_Large
Cave_Large1
TVGIVTech_
LightingBakes
Tech_TechArt400_ZoneC
AnimAO
Material_Cloth_Lights
Material_Cloth_LightsShape
Material_Cloth
SmLight
Ice_Spike_AlphaRefraction
Ice_Spike_FramebufferRefraction
Ice_Spike_CubemapRefraction
Ice_Sphere_AlphaRefraction
Ice_Sphere_FramebufferRefraction
Ice_Sphere_CubemapRefraction
BVSLightingBakeB
RNtech_techart400_zonec_:BVSTechZoneC
RNtech_techart400_zonec_:sao_meshCleanup_ngonFa
View Shading Lighting Show Renderer Panels
0.00
1.00
sRGB gamma (legacy)
persp
List Selected Focus Attributes Display Show Help
Select an object in the scene to view and edit its attributes
Select
Load Attributes
Copy Tab
RENDERING ENGINE ARCHITECTURE CONFERENCE 2024

We first need to select the object in Maya…


Outliner
Display Show Help
Search...
persp
top
front
side
Sheet
NavMesh1
Env
Tech_TechArt200_ZoneA
Tech_TechArt100_Entrance
Cave_Large
Cave_Large1
TVGIVTech_
LightingBakes
Tech_TechArt400_ZoneC
AnimAO
Material_Cloth_Lights
Material_Cloth_LightsShape
Material_Cloth
SmLight
Ice_Spike_AlphaRefraction
Ice_Spike_FramebufferRefraction
Ice_Spike_CubemapRefraction
Ice_Sphere_AlphaRefraction
Ice_Sphere_FramebufferRefraction
Ice_Sphere_CubemapRefraction
BVSLightingBakeB
RNtech_techart400_zonec_:BVSTechZoneC
RNtech_techart400_zonec_:sao_meshCleanup_ngonFa
View Shading Lighting Show Renderer Panels
persp
List Selected Focus Attributes Display Show Help
AnimAO
Focus
Presets
Show Hide
Pivots
Limit Information
Display
Node Behavior
UUID
SCEASM Behaviors / Scripts
SCEASM Zone
Extra Attributes
Notes: AnimAO
Select
Load Attributes
Copy Tab
RENDERING ENGINE ARCHITECTURE CONFERENCE 2024

transform: AnimAO

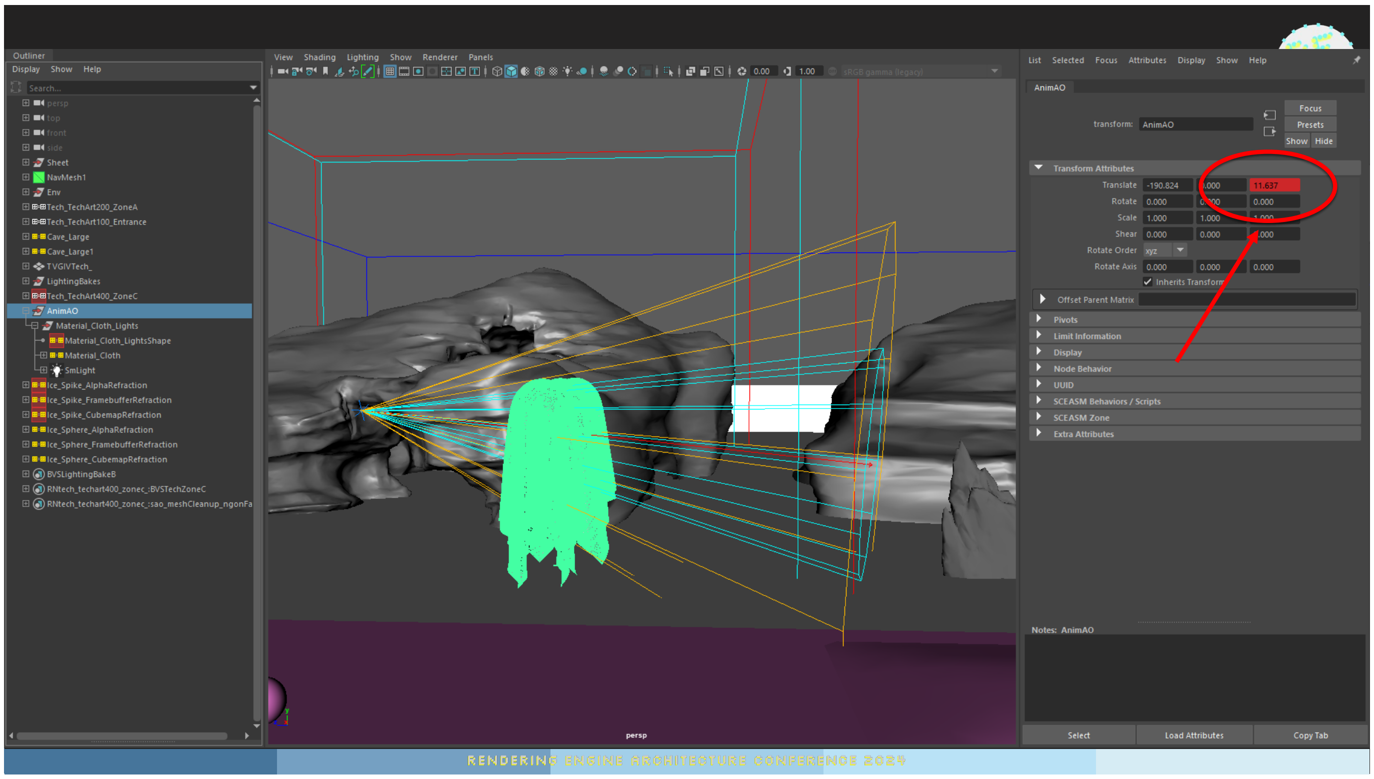
Transform Attributes

Translate	-190.824	0.000	11.637
Rotate	0.000	0.000	0.000
Scale	1.000	1.000	1.000
Shear	0.000	0.000	0.000
Rotate Order	xyz		
Rotate Axis	0.000	0.000	0.000

✓ Inherits Transform

Offset Parent Matrix

Then if you look to the top right…


Outliner
Display Show Help
persp
top
front
side
Sheet
NavMesh1
Env
Tech_TechArt200_ZoneA
Tech_TechArt100_Entrance
Cave_Large
Cave_Large1
TVGIVTech_
LightingBakes
Tech_TechArt400_ZoneC
AnimAO
Material_Cloth_Lights
Material_Cloth_LightsShape
Material_Cloth
SmLight
Ice_Spike_AlphaRefraction
Ice_Spike_FramebufferRefraction
Ice_Spike_CubemapRefraction
Ice_Sphere_AlphaRefraction
Ice_Sphere_FramebufferRefraction
Ice_Sphere_CubemapRefraction
BVSLightingBakeB
RNtech_techart400_zonec_:BVSTechZoneC
RNtech_techart400_zonec_:sao_meshCleanup_ngonFa
View Shading Lighting Show Renderer Panels
persp
List Selected Focus Attributes Display Show Help
Focus
Presets
Show Hide
Pivots
Limit Information
Display
Node Behavior
UUID
SCEASM Behaviors / Scripts
SCEASM Zone
Extra Attributes
Select
Load Attributes
Copy Tab
RENDERING ENGINE ARCHITECTURE CONFERENCE 2023

AnimAO

transform: AnimAO

Transform Attributes

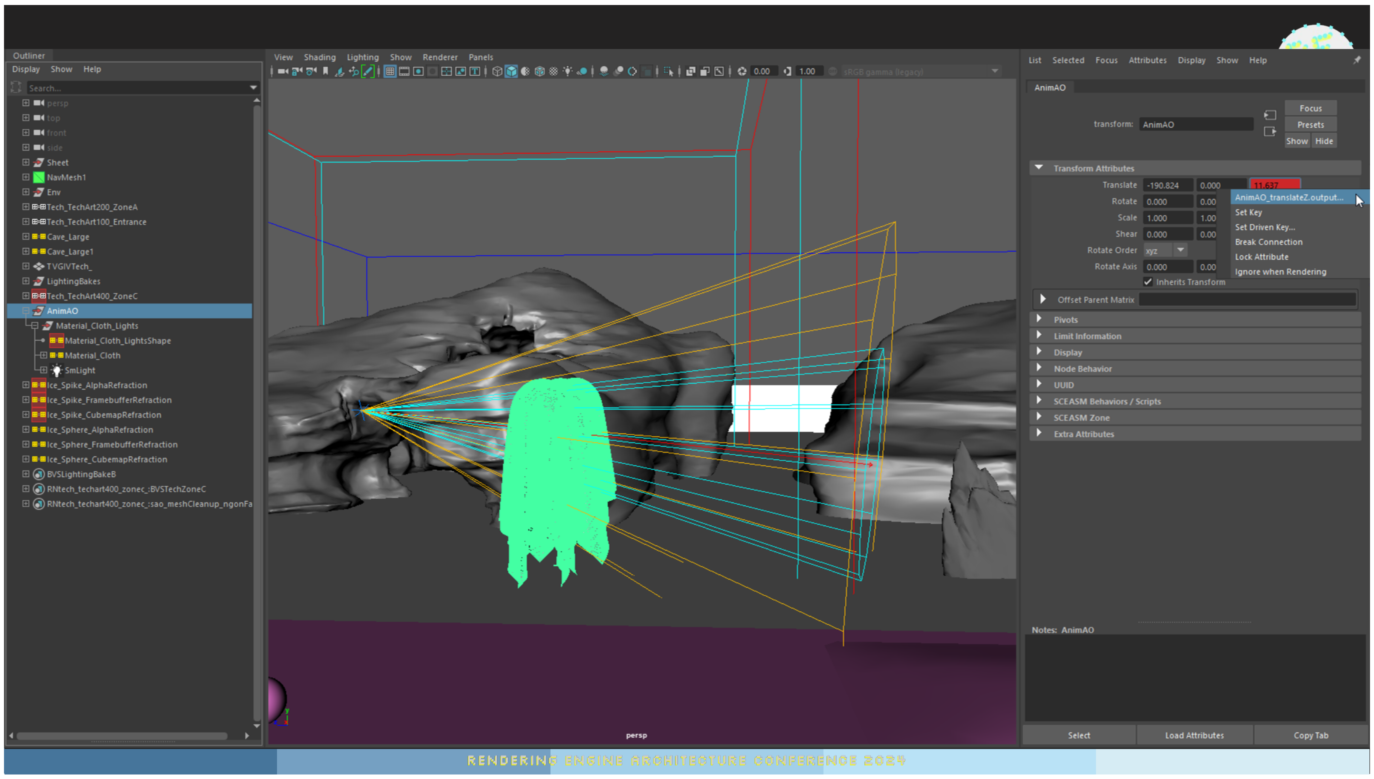
Translate	-190.824	0.000	11.637
Rotate	0.000	0.000	0.000
Scale	1.000	1.000	1.000
Shear	0.000	0.000	0.000
Rotate Order	xyz		
Rotate Axis	0.000	0.000	0.000

✓ Inherits Transform

Offset Parent Matrix

Notes: AnimAO

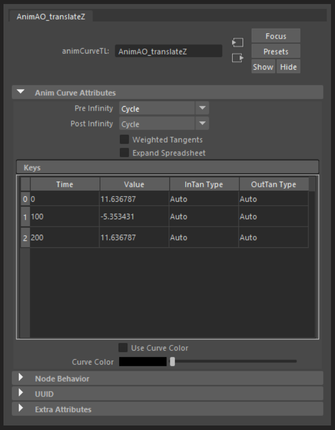
You’ll see the Z coordinate of the transform in red, meaning that it’s animated.



transform: AnimAO
Translate -190.824 0.000 11.637
Rotate 0.000 0.00
Scale 1.000 1.00
Shear 0.000 0.00
Rotate Order xyz
Rotate Axis 0.000 0.00
✓ Inherits Transform

If we right click on that and open it up, we see the animation we have assigned and can edit it.

Animation



AnimAO_translateZ

animCurveTL: AnimAO_translateZ

Anim Curve Attributes

Pre Infinity: Cycle

Post Infinity: Cycle

Weighted Tangents

Expand Spreadsheet

Keys

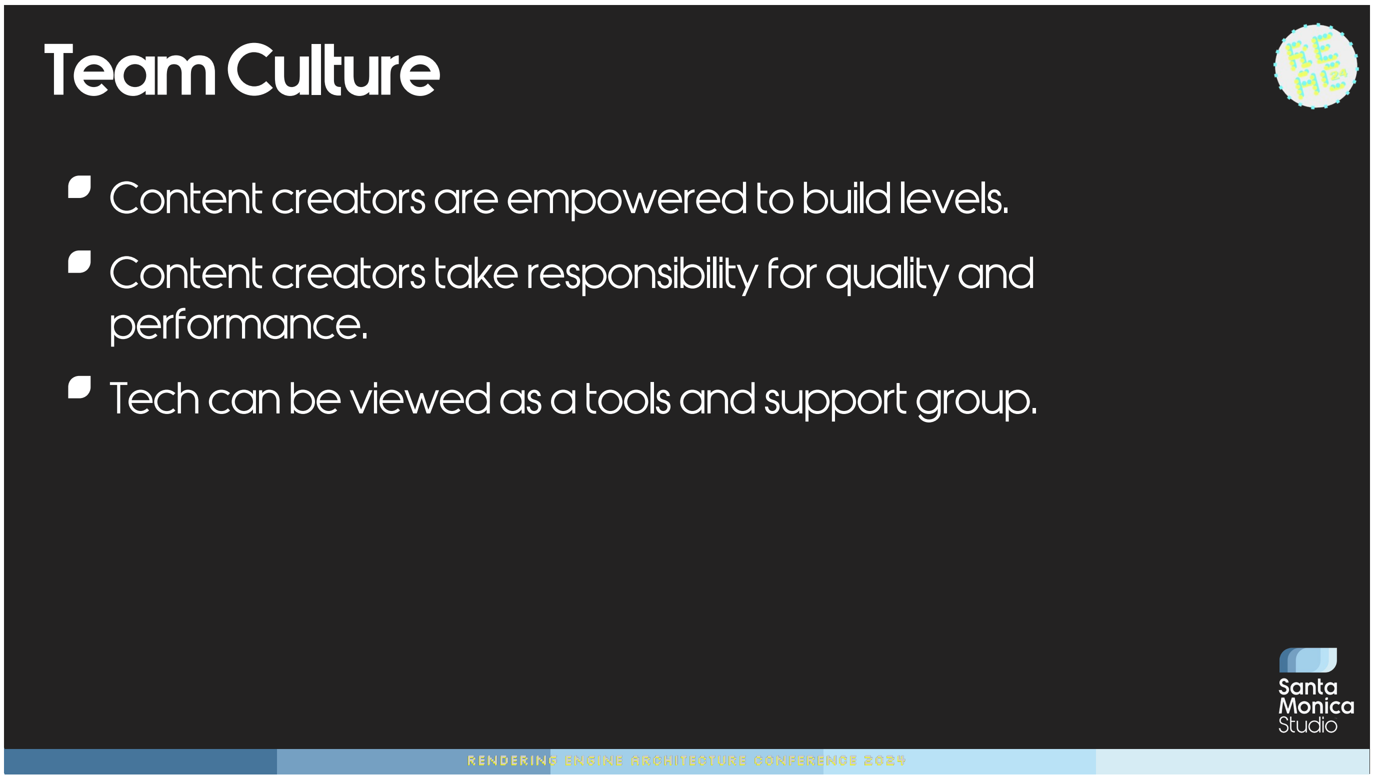
	Time	Value	InTan Type	OutTan Type
0	0	11.636787	Auto	Auto
1	100	-5.353431	Auto	Auto
2	200	11.636787	Auto	Auto

Use Curve Color

Curve Color

Santa Monica Studio

This is a pretty simple back and forth animation that we're editing in Maya, and it comes right into game.

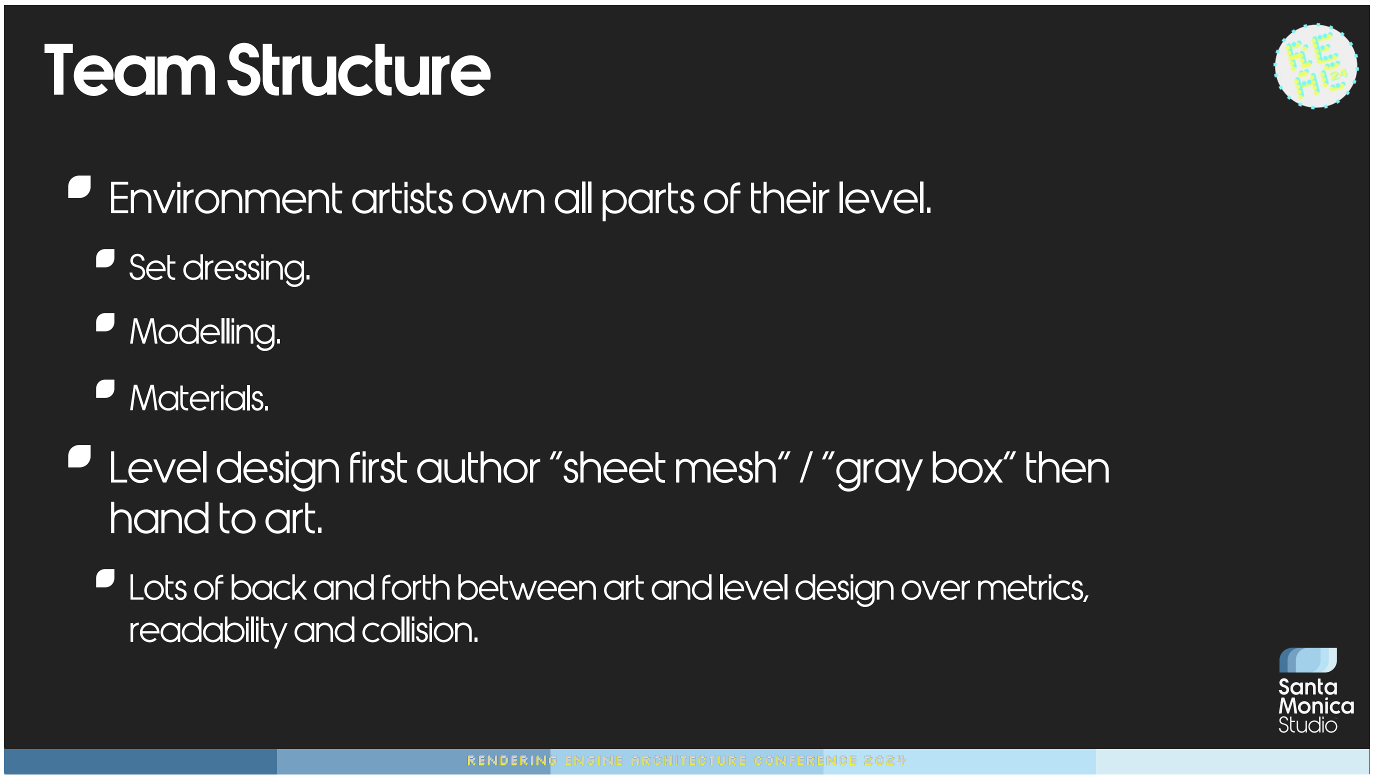


Team Culture

- Content creators are empowered to build levels.
- Content creators take responsibility for quality and performance.
- Tech can be viewed as a tools and support group.

Santa Monica Studio

I've occasionally seen the attitude from artists of "well, it's my responsibility to make this level look good, so I don't care about the technical features you want to add."



Team Structure

- Environment artists own all parts of their level.
  - Set dressing.
  - Modelling.
  - Materials.
- Level design first author "sheet mesh" / "gray box" then hand to art.
  - Lots of back and forth between art and level design over metrics, readability and collision.

But this empowerment over artists has an effect on our team structure. As artists are given so much control in Maya, typically an artist owns all parts of their level, from modelling to materials to set dressing. We do have a classic split between level design and art though – first level designers author sheet mesh, then hand to art. Of course there's then plenty of back and forth to ensure metrics and readability are respected, plus getting good collision.

The Future

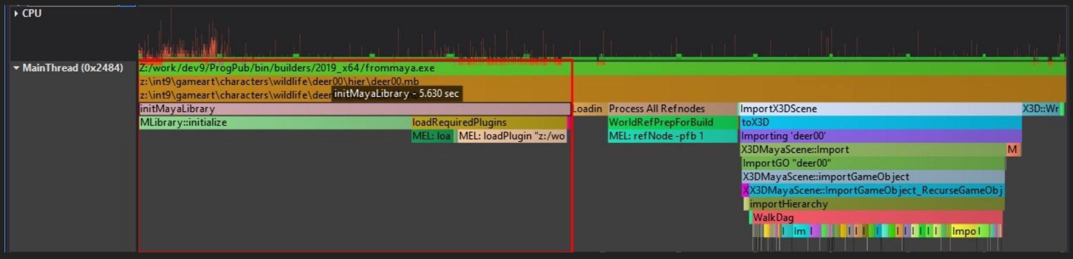
Problems with Maya as an Editor and our plans to overcome them

Problems with Maya as an Editor

- Build times
- Performance
- Everything must go through Maya
- Dependency on a third party application

Build Times

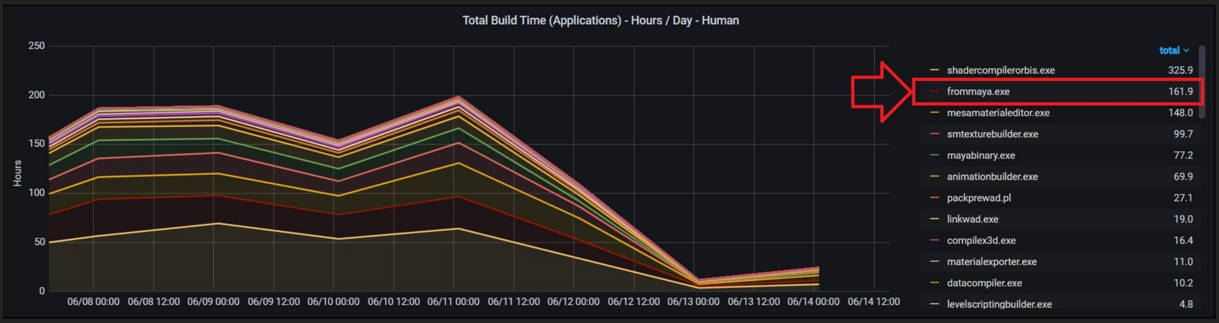
- Initializing the Maya library takes 5s:



Thanks Vadim for this info!

Build Times

- We spend ~160 hours/week to process content through the Maya API:



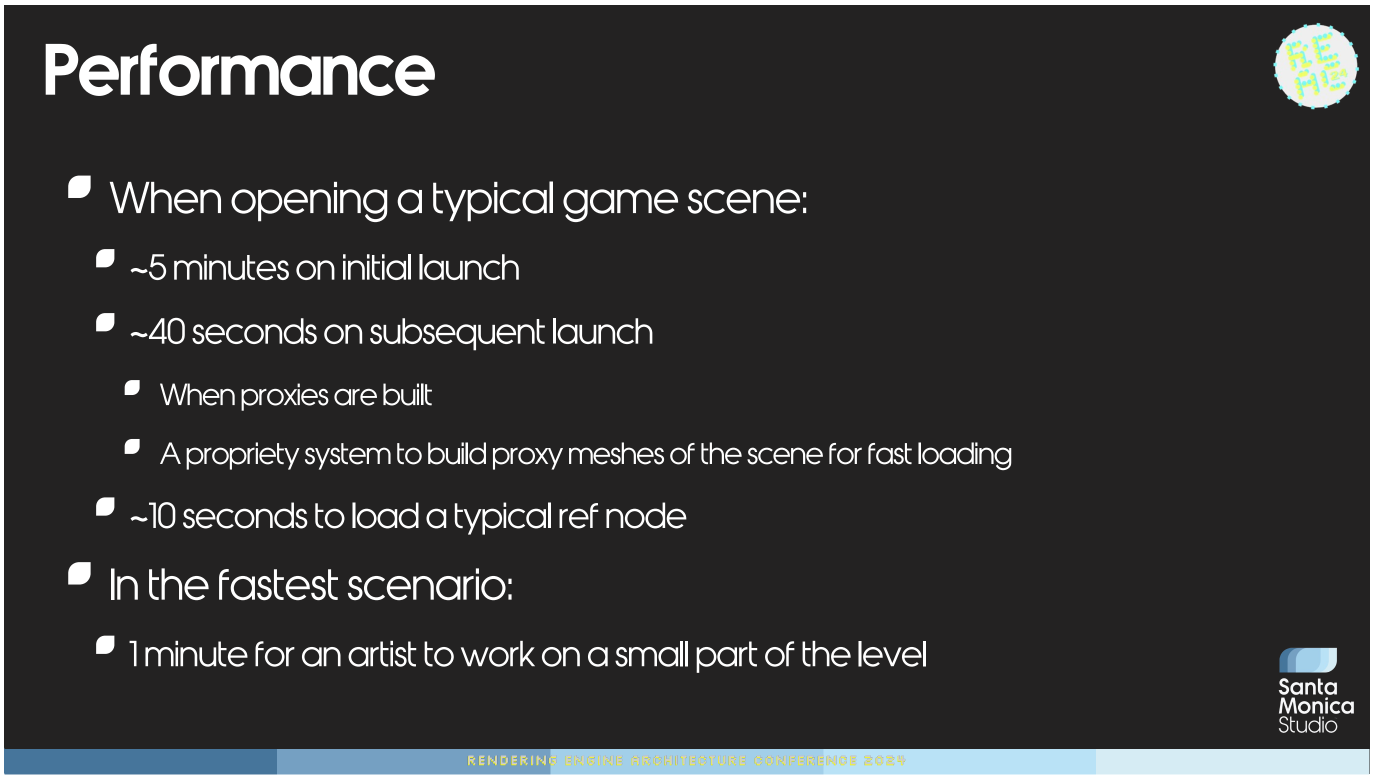
Thanks Vadim for this info!

These figures are from a year or two ago, but haven’t changed much.

Build Times

- We already optimized searching Maya files for dependencies.
- "MayaNinja":
  - Our custom Maya parsing tool.
  - Many times faster than using the Maya API.

Santa Monica Studio



Performance

- When opening a typical game scene:
  - ~5 minutes on initial launch
  - ~40 seconds on subsequent launch
    - When proxies are built
    - A propriety system to build proxy meshes of the scene for fast loading
  - ~10 seconds to load a typical ref node
- In the fastest scenario:
  - 1 minute for an artist to work on a small part of the level

Santa Monica Studio

Performance

- Scene culler tool:
  - For animators working on cinematics.
  - Cuts out the relevant portion of the .mb scene to the cinematic.
  - To significantly improve load times.

Santa Monica Studio

To solve performance problems with Maya and .mb files, we don't just have the proxy system. We also have this scene culler tool for animators.

Everything Goes Through Maya

- Everything in our game has to go through Maya.
- Maya isn't always the best tool for the job.
  - Houdini for procedural development.
  - Rise in junior artists who know Blender.
  - Simple level design work:
    - Placing audio emitters.

Santa Monica Studio

Dependency on a Third Party Tool

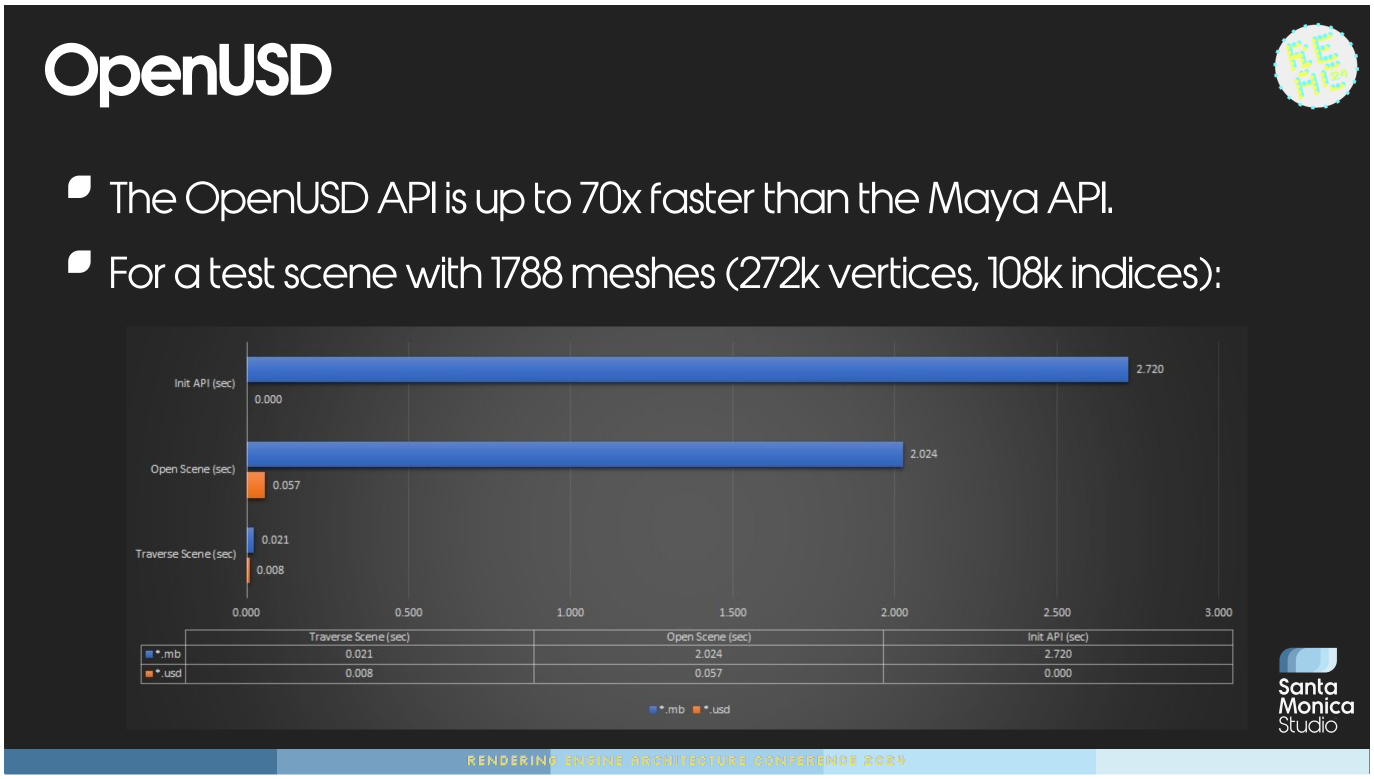
- It's unhealthy for the studio to be so dependent upon one third party tool
- Dependent upon Autodesk for support

Santa Monica Studio

OpenUSD

- OpenUSD solves many of our problems.
  - Fast API
  - Interoperability with many DCCs
  - OpenUSD editors already exist
    - e.g. NVIDIA Omniverse

Santa Monica Studio



OpenUSD

- The OpenUSD API is up to 70x faster than the Maya API.
- For a test scene with 1788 meshes (272k vertices, 108k indices):

	Traverse Scene (sec)	Open Scene (sec)	Init API (sec)
🟦 *.mb	0.021	2.024	2.720
🟧 *.usd	0.008	0.057	0.000

🟦 *.mb 🟧 *.usd

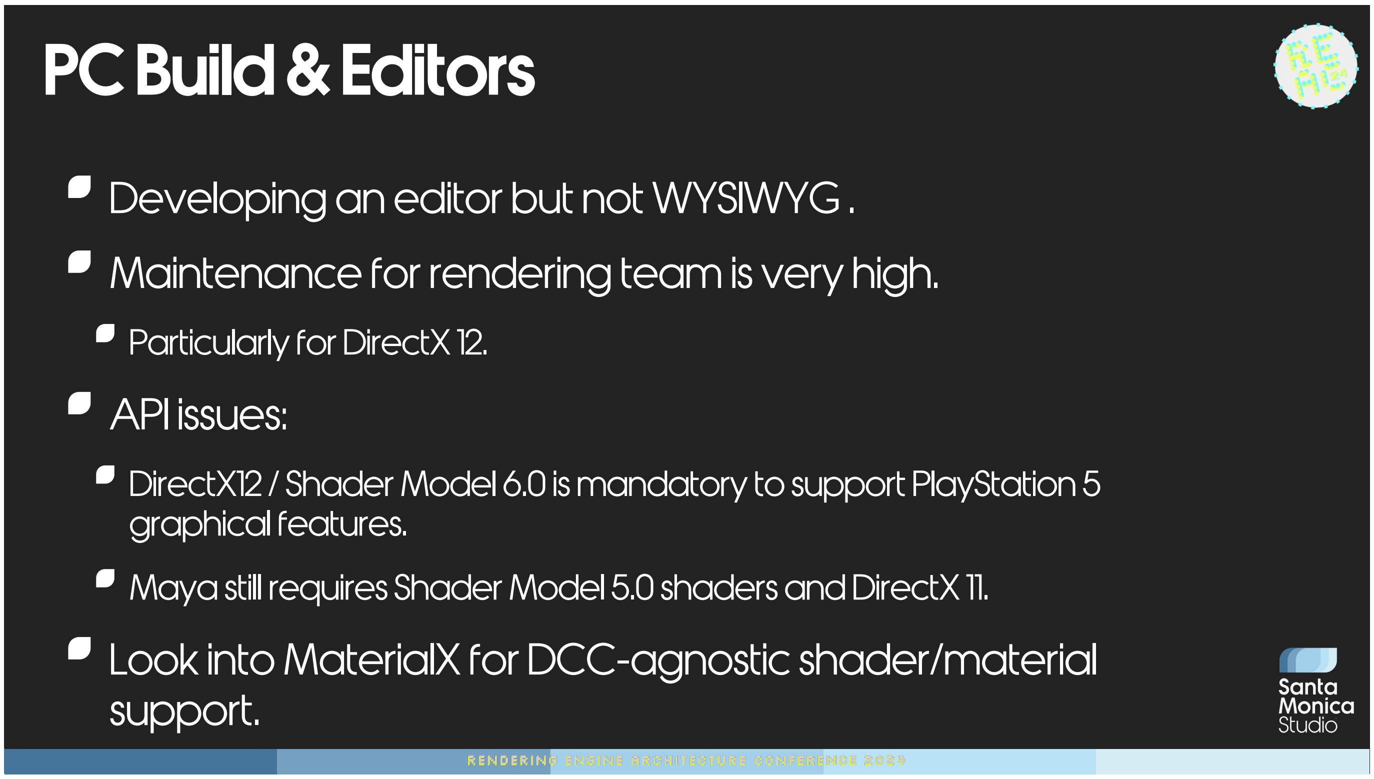
We compare Maya in blue and OpenUSD in orange… in total it's over a 7000% speed up!

OpenUSD

- Opening the Maya scene mentioned earlier:
  - 50 seconds to load the entire level, including all ref nodes.
  - No need for a proxy system.

Previously it was 40 seconds in the best case scenario to load the level minus any ref nodes, plus around 10 seconds to load an individual ref node. Often (if proxies weren't built) it would take a significant amount longer.

With USD we always hit our previous best case scenario… and actually have the whole level loaded. And we can remove our complicated proxy system from the engine.



PC Build & Editors

- Developing an editor but not WYSIWYG .
- Maintenance for rendering team is very high.
  - Particularly for DirectX 12.
- API issues:
  - DirectX12 / Shader Model 6.0 is mandatory to support PlayStation 5 graphical features.
  - Maya still requires Shader Model 5.0 shaders and DirectX 11.
- Look into MaterialX for DCC-agnostic shader/material support.

Santa Monica Studio

RENDERING ENGINE ARCHITECTURE CONFERENCE 2024

As mentioned earlier, Maya is not the best place for many level design tasks, such as placing audio emitters. We're developing an editor for those.

We preview our materials in Maya, so we need HLSL Shader Model 5.0 shaders.

Summary

Conclusion

- Using Maya as an editor has heavily influenced SMS:
  - Engine design and architecture
  - Artist workflows
  - Company culture
- Maya also has many limitations:
  - The future is OpenUSD

Santa Monica Studio

Thanks

- SMS:
  - Josh Hobson
  - Vadim Slyusarev
  - Mat Hendry
  - Sam Sternklar
  - Nathan Kennedy
  - Kyle Bromley
  - All past and current contributors to our engine
- REAC:
  - Angelo Pesce
  - Stephanie Sampson
  - Natalya Tatarchuk
  - Michael Vance

Santa Monica Studio

Thank You!

Stephen McAuley
Technical Director
stephen.mcauley@sony.com
@stevemcauley

Matt Pettineo
Lead Rendering Programmer
matt.pettineo@sony.com
@mynameismjp

Santa Monica Studio

STUDIOS

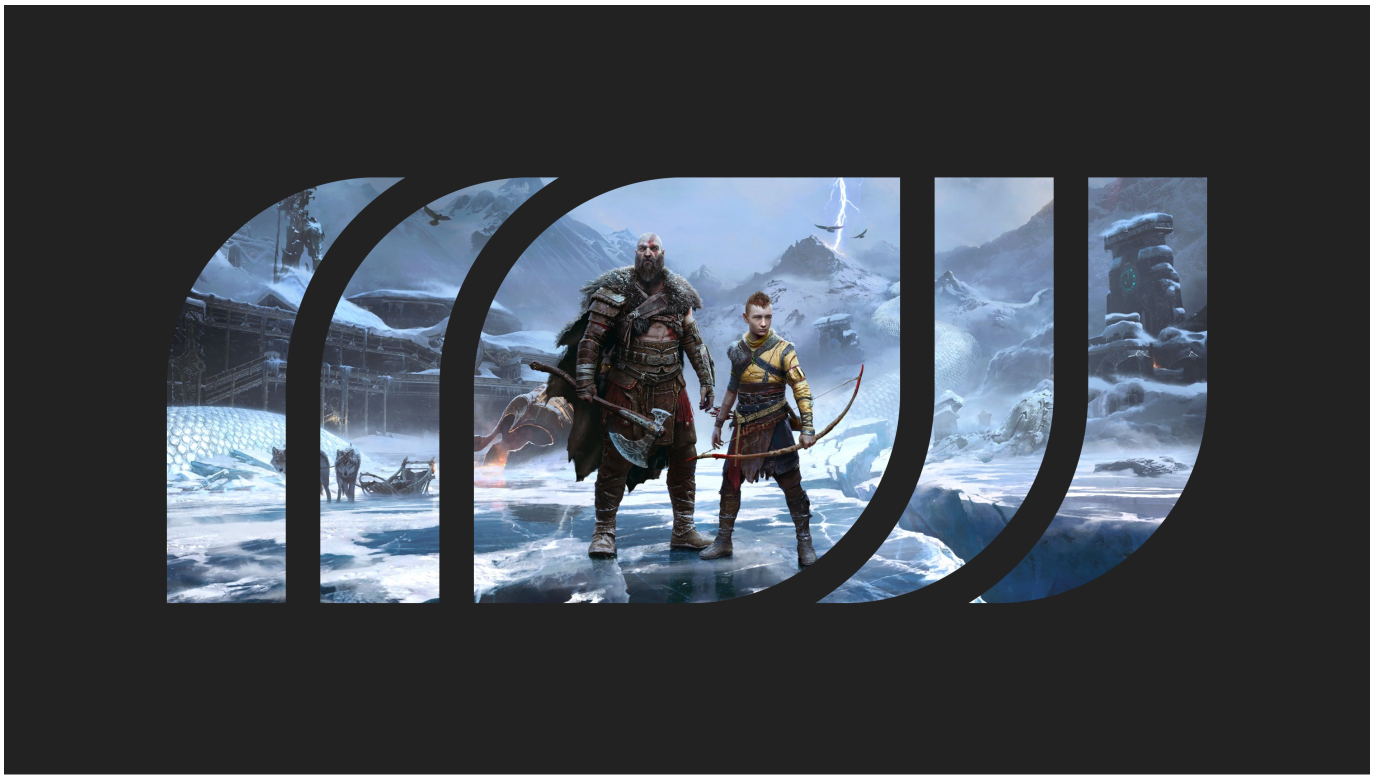
Santa Monica Studio

Our journey
Your story

We're hiring for what's next!

We're expanding our family across disciplines and would love to meet you. Please visit sms.playstation.com/careers for all openings or drop us a line at sms@sony.com

@ santamonicastudio @ SonySantaMonica @ santamonicastudio



Santa
Monica
Studio