VRDC

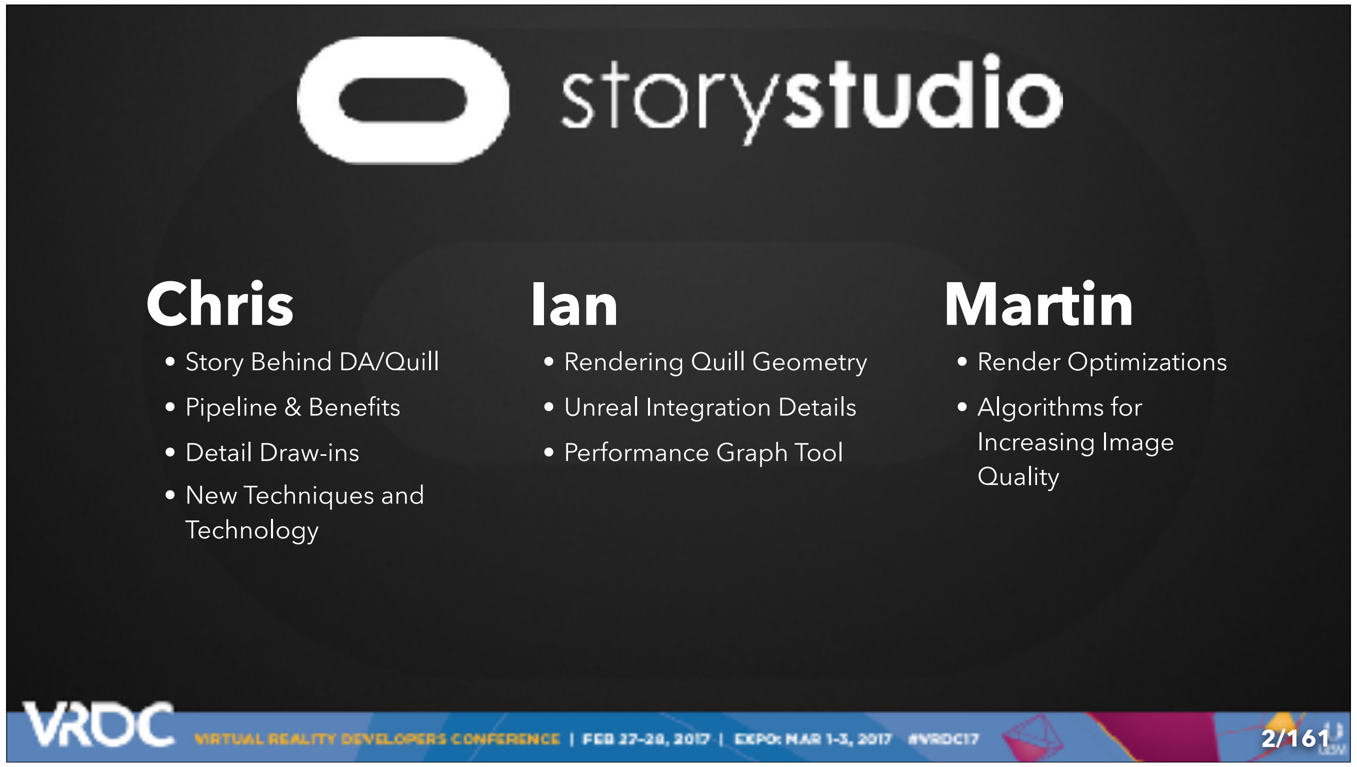
Oculus Story Studio's Dear Angelica: Inventing, Integrating, and Rendering Quill

Chris Horne - CG Supervisor
chorne@oculus.com @distastee

Ian Wakelin - Coder
ian.wakelin@oculus.com

Martin Mittring - Coder
martin.mittring@oculus.com @mittringmartin

Hey Everyone - I'm Chris Horne - CG Supervisor on Dear Angelica. This is Ian Wakelin - our Quill integration Lead - and Martin Mittring - our core graphics architect.



storystudio

Chris

- Story Behind DA/Quill
- Pipeline & Benefits
- Detail Draw-ins
- New Techniques and Technology

Ian

- Rendering Quill Geometry
- Unreal Integration Details
- Performance Graph Tool

Martin

- Render Optimizations
- Algorithms for Increasing Image Quality

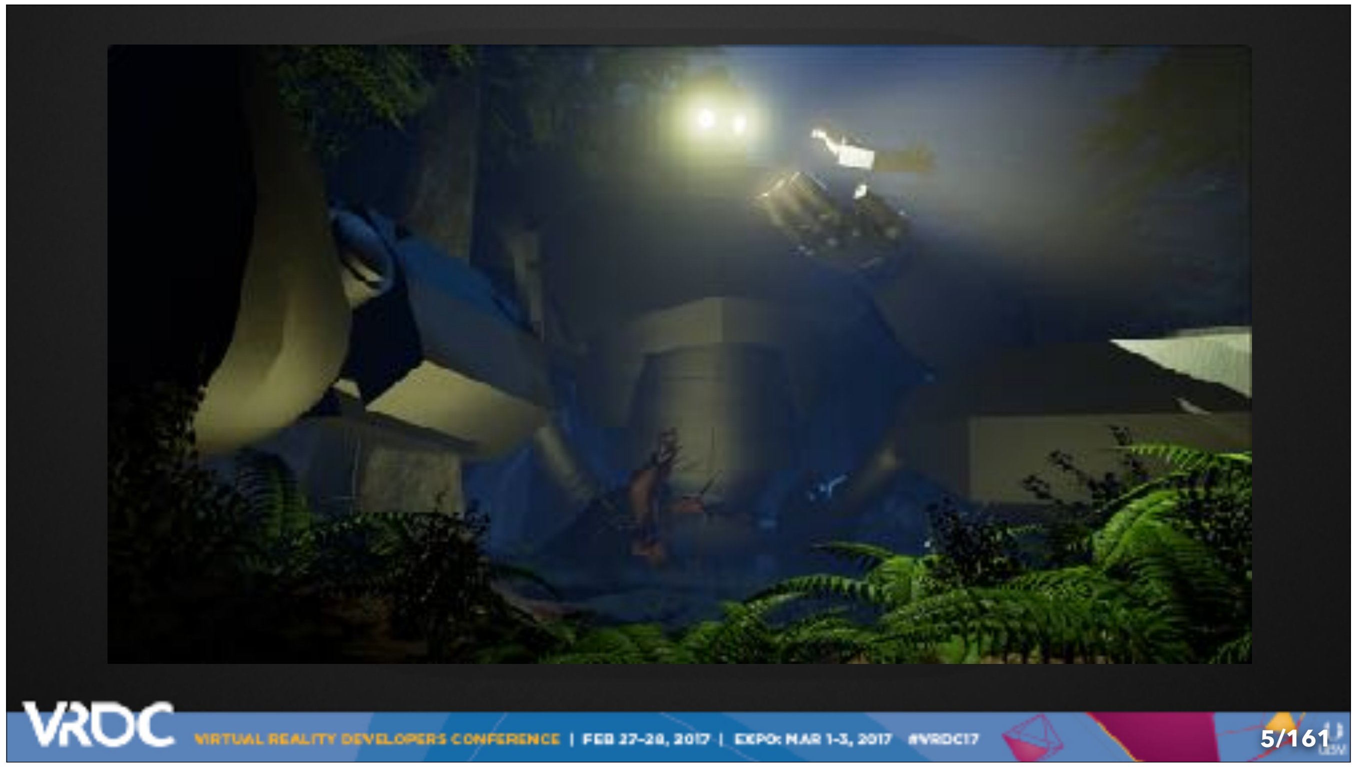
VRDC VIRTUAL REALITY DEVELOPERS CONFERENCE | FEB 27-28, 2017 | EXPO: MAR 1-3, 2017 #VRDC17

We all work at Story Studio - which is Oculus' first party film studio. We're here to talk about some of the technical and creative aspects of Dear Angelica and Quill. Mine is more of a medium level overview of a bunch of topics - whereas Ian and Martin will do a deep dive into the technical underpinnings of the rendering technology and it's implementation in Unreal.

But first - who here has seen Dear Angelica? Great! Who here has used Quill? For those of you who didn't raise your hands - I should explain that Dear Angelica is a 13 minute short film available for free on the Oculus store. The entire thing was drawn in Quill, a standalone OpenGL illustration app we created to help make this film. So here's a quick trailer of Dear Angelica:

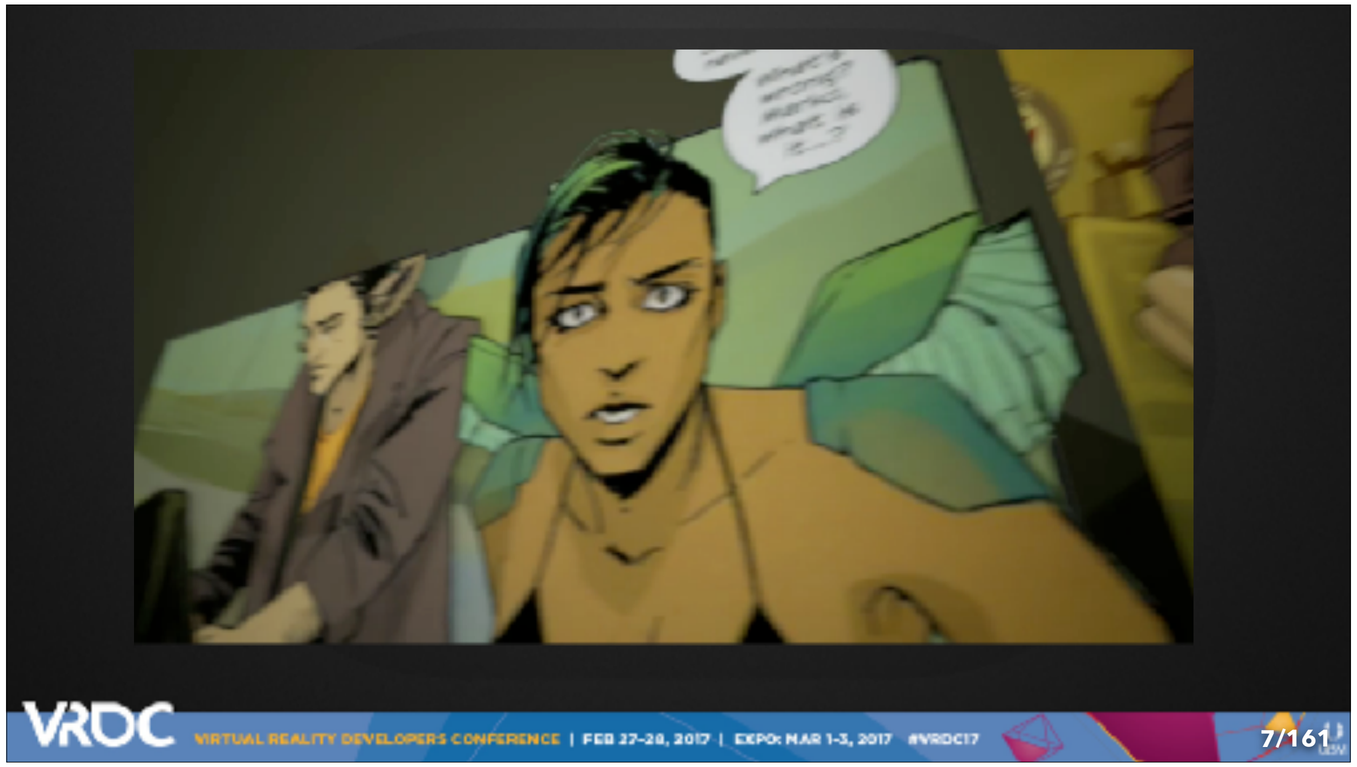
[intro video]

The seeds for Dear Angelica were originally planted back in 2014. When Story Studio was still just 4 people using DK2s,

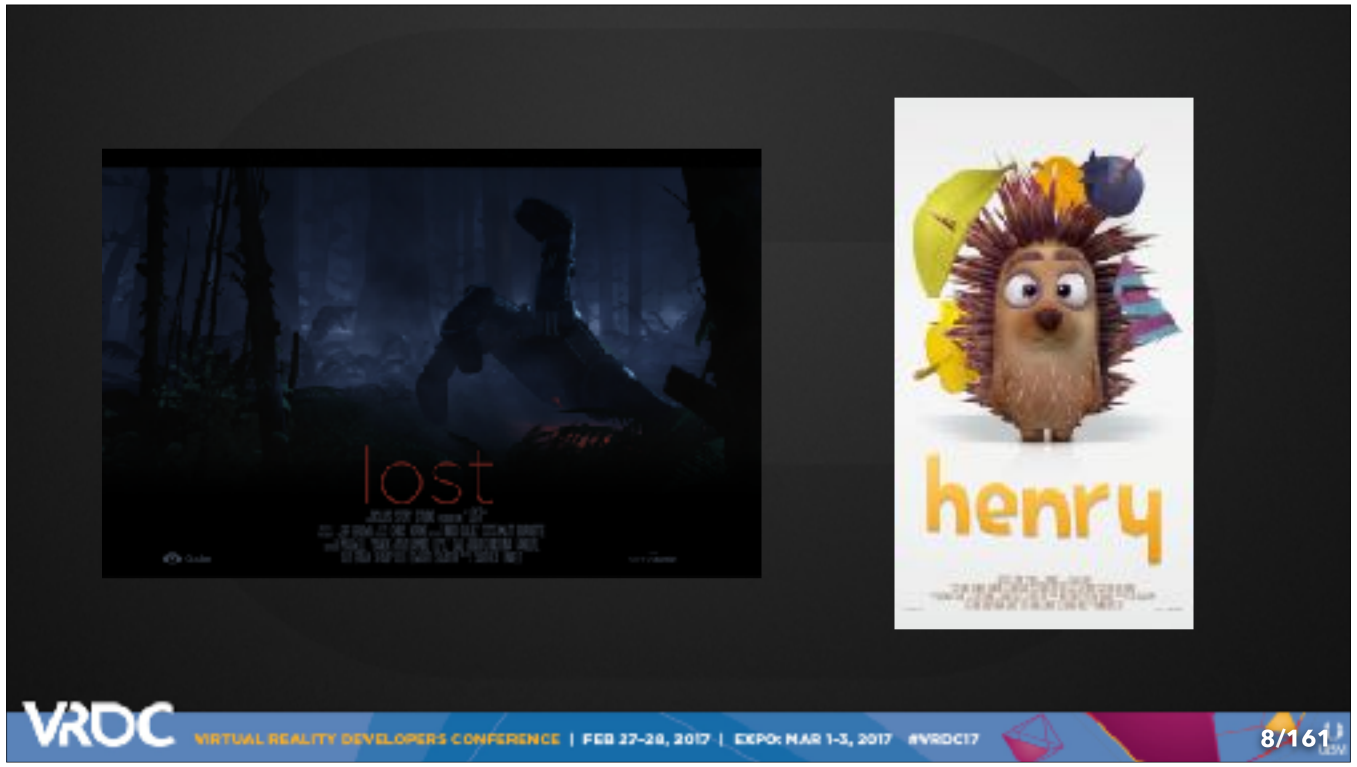

VRDC
VIRTUAL REALITY DEVELOPERS CONFERENCE | FEB 27-28, 2017 | EXPO: MAR 1-3, 2017 #VRDC17
5/161

Lost had no shaders,

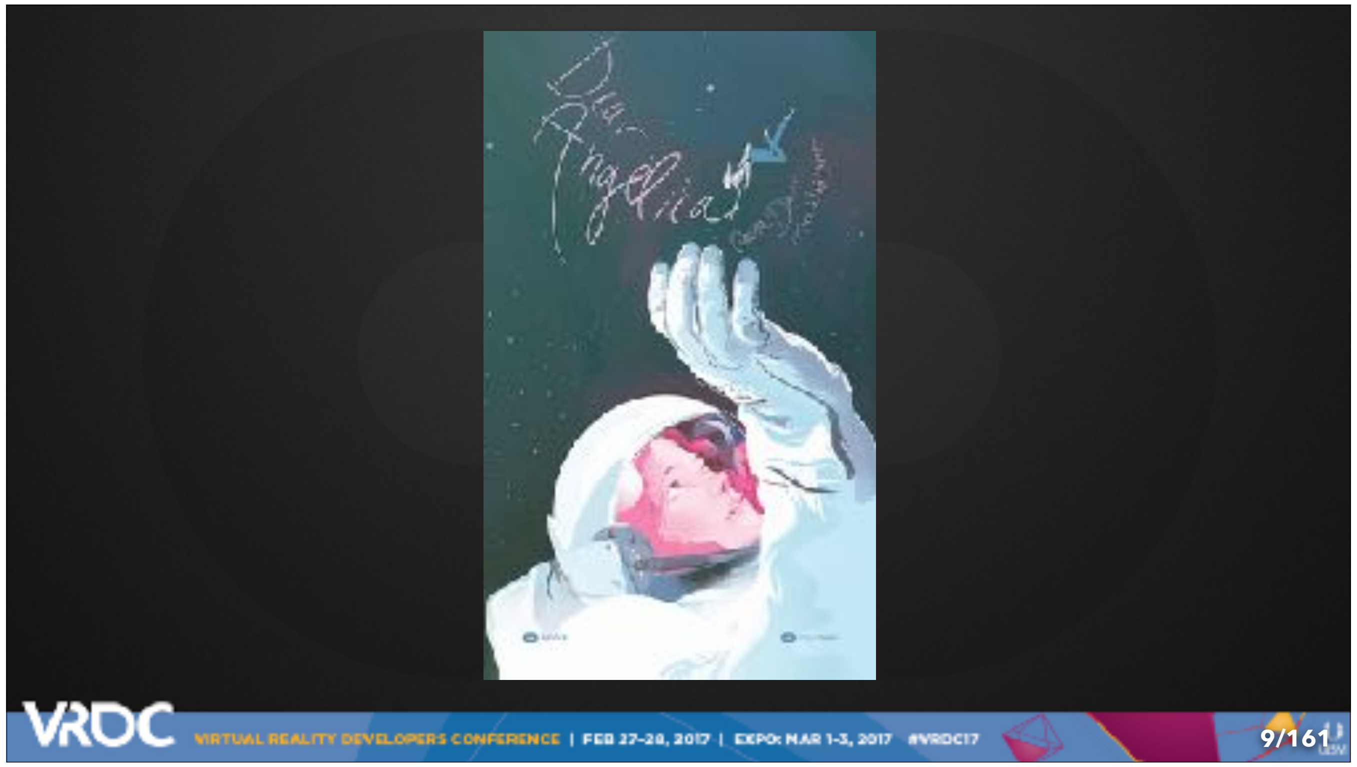
and Henry was still a porcupine that loved balloons. We had the big idea of bringing comic books into VR - but weren’t quite sure how to do it or what it would look like.


VRDC
VIRTUAL REALITY DEVELOPERS CONFERENCE | FEB 27-28, 2017 | EXPO: MAR 1-3, 2017 #VRDC17
7/161

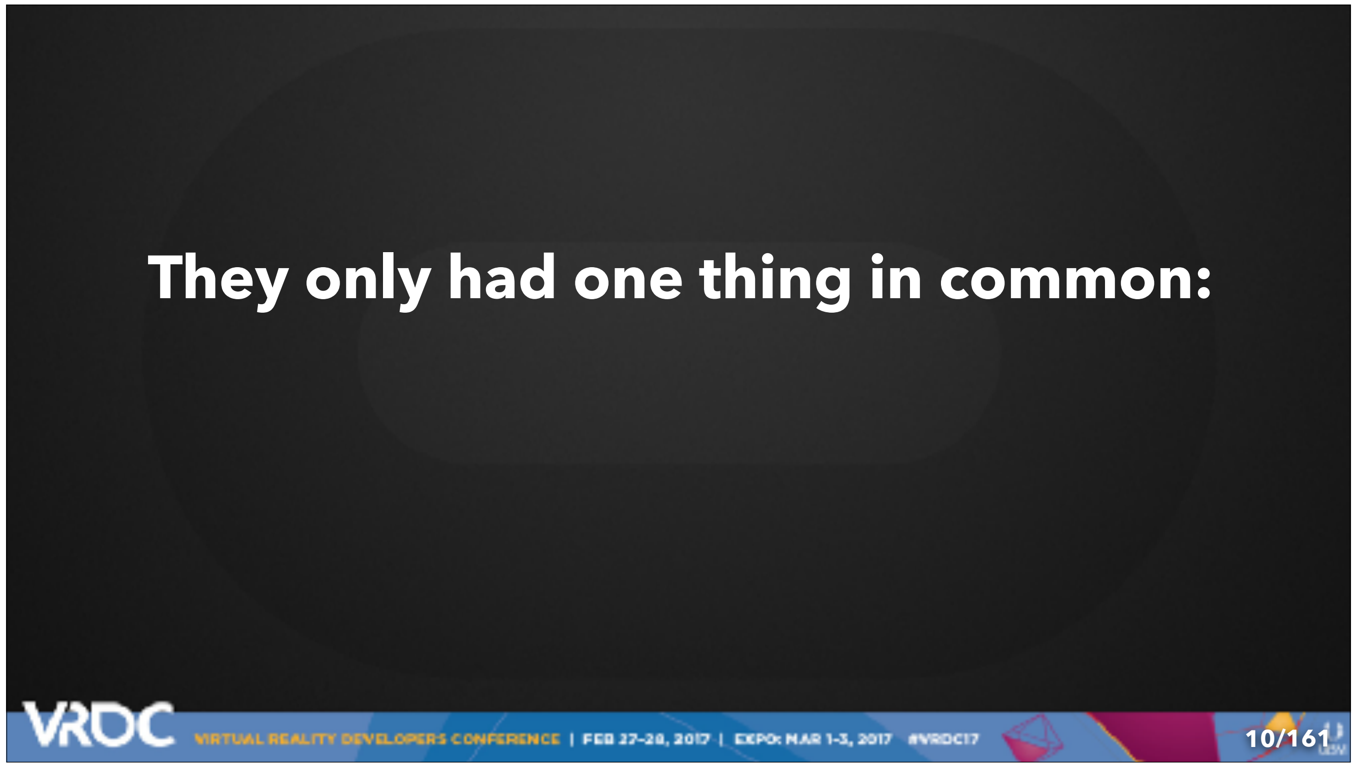
One of the very first prototypes we did as a studio was taking some scans of a comic book and split the images out in depth so they would get a little bit of parallax. We made it so that as you look left or right - the panels would kinda scroll on by. We showed this early prototype to Carmack one day and he said “Cool………..but keep working”


lost
henry
VRDC
VIRTUAL REALITY DEVELOPERS CONFERENCE | FEB 27-28, 2017 | EXPO: MAR 1-3, 2017 #VRDC17
8/161

He was right. Over the next year and a half while we worked on our more immediate problems of story in VR and characters in VR,


Dear Angelica
VRDC VIRTUAL REALITY DEVELOPERS CONFERENCE | FEB 27-28, 2017 | EXPO: MAR 1-3, 2017 #VRDC17
9/161

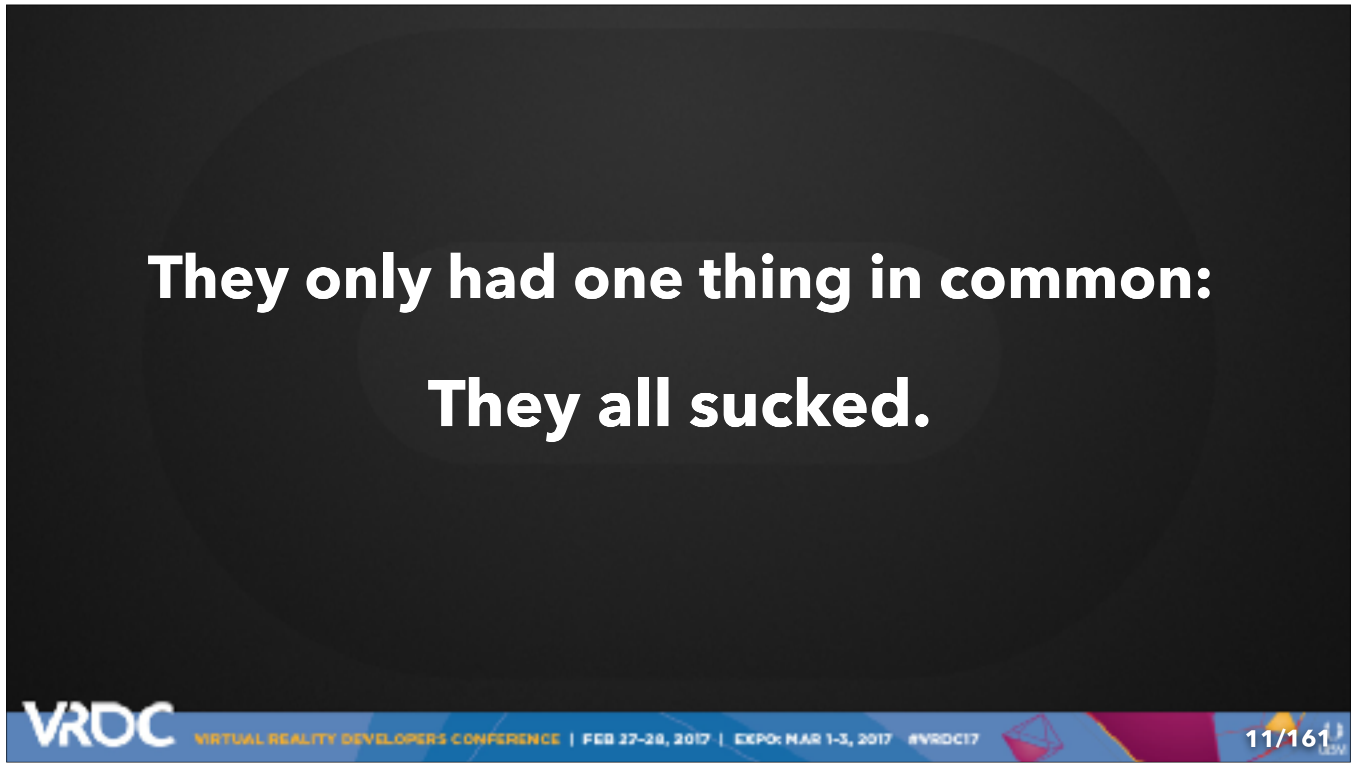
we subconsciously cranked on this problem of comics and illustration in VR. By the time Henry was done - Dear Angelica was a rough script and a giant pile of fun visual ideas to try.



They only had one thing in common:

VRDC VIRTUAL REALITY DEVELOPERS CONFERENCE | FEB 27-28, 2017 | EXPO: MAR 1-3, 2017 #VRDC17

This pile of ideas had only one thing in common:



They only had one thing in common:

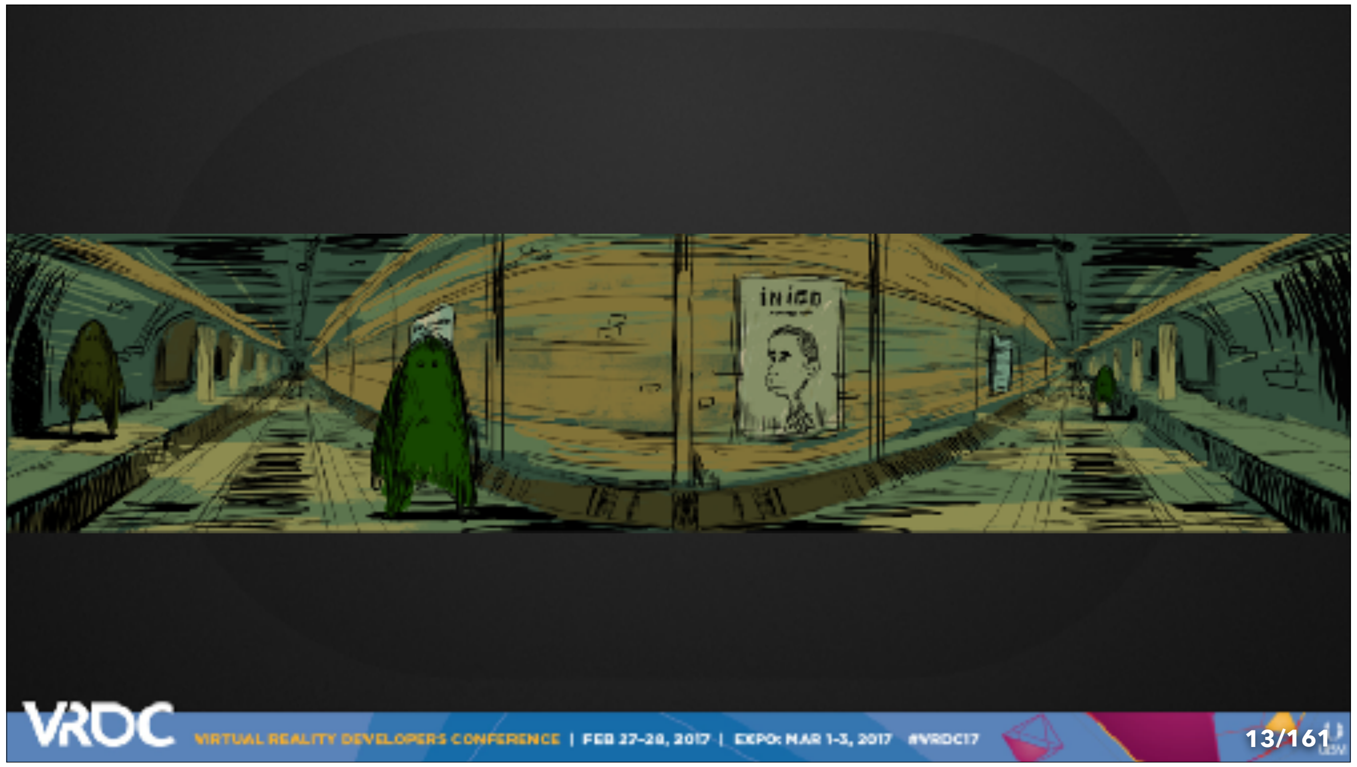
They all sucked.

VRDC VIRTUAL REALITY DEVELOPERS CONFERENCE | FEB 27-28, 2017 | EXPO: MAR 1-3, 2017 #VRDC17

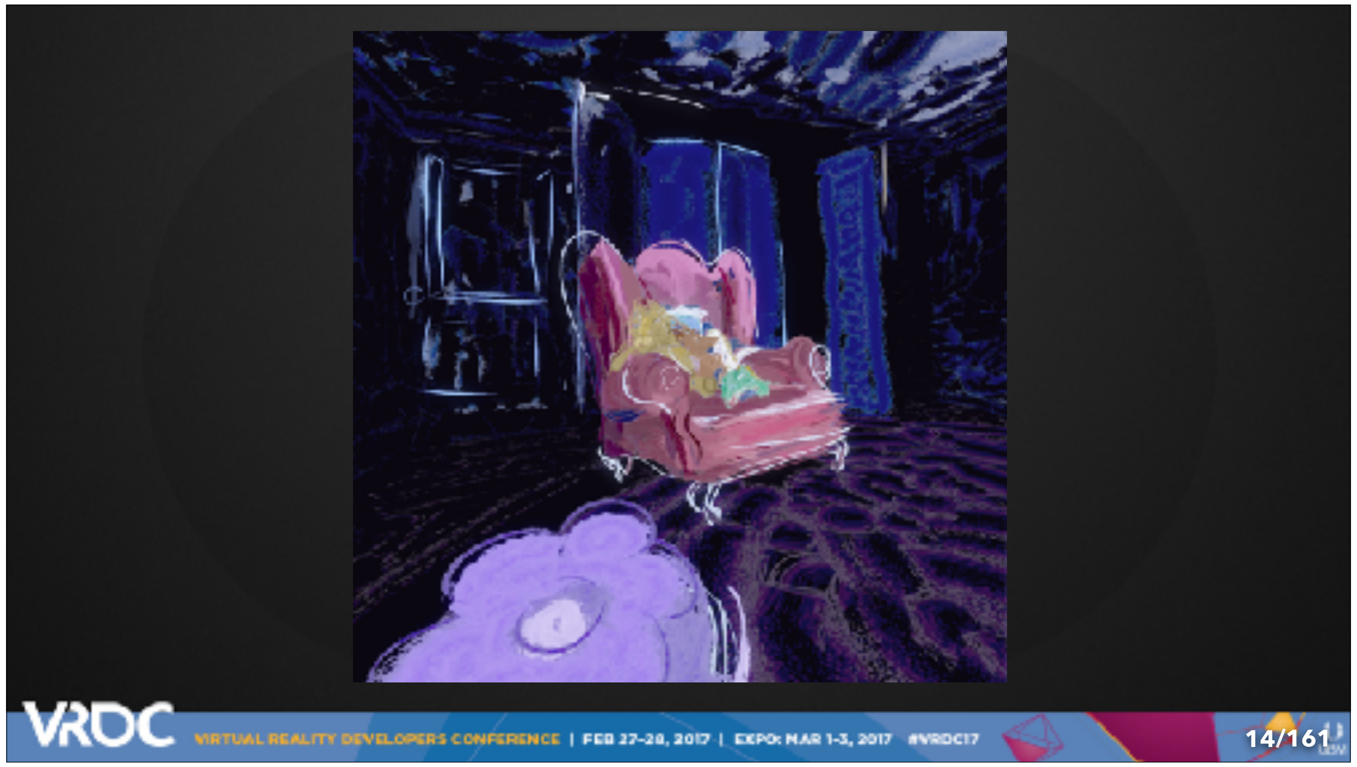
They all sucked.

VRDC
VIRTUAL REALITY DEVELOPERS CONFERENCE | FEB 27-28, 2017 | EXPO: MAR 1-3, 2017 #VRDC17
12/161

We tried modeling the scenes and then painting on it.


INIGO
VRDC
VIRTUAL REALITY DEVELOPERS CONFERENCE | FEB 27-28, 2017 | EXPO: MAR 1-3, 2017 #VRDC17
13/161

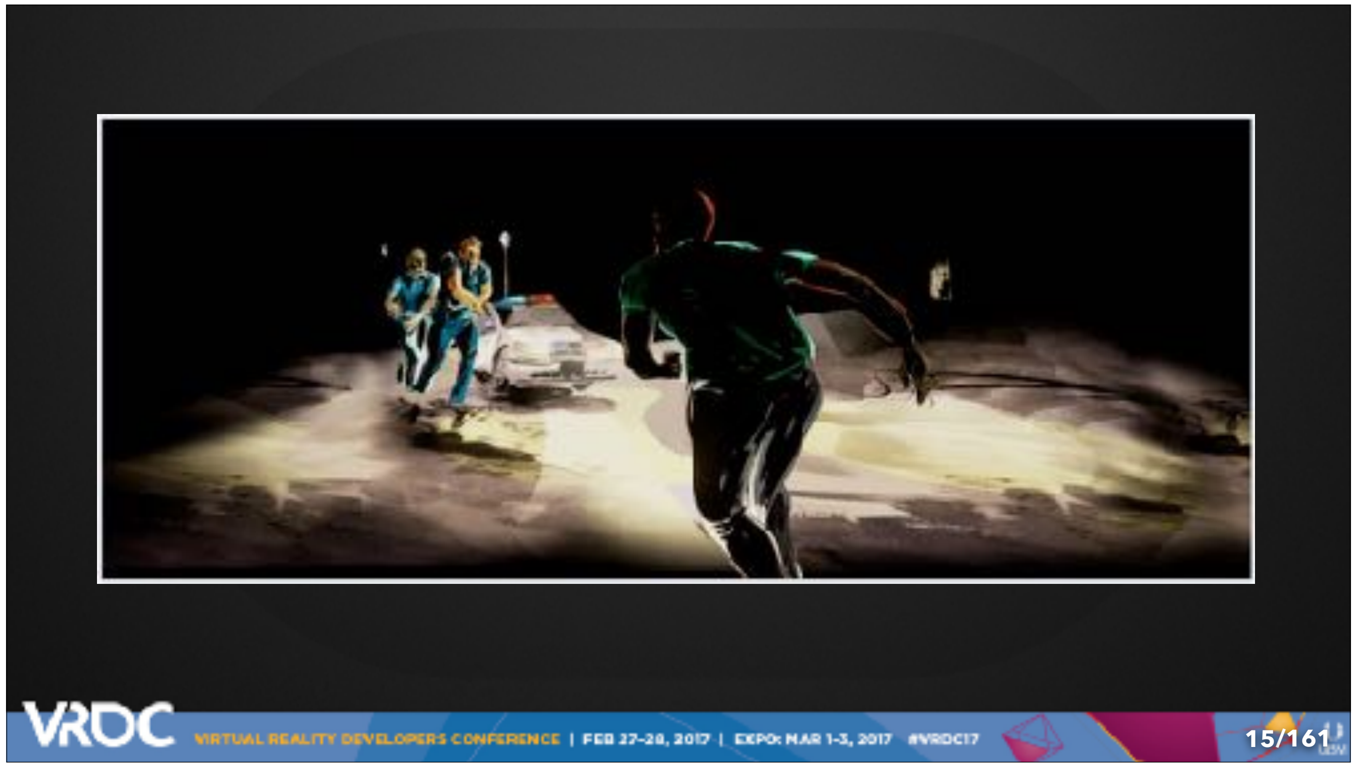
We tried painting the sketches first and projecting it.



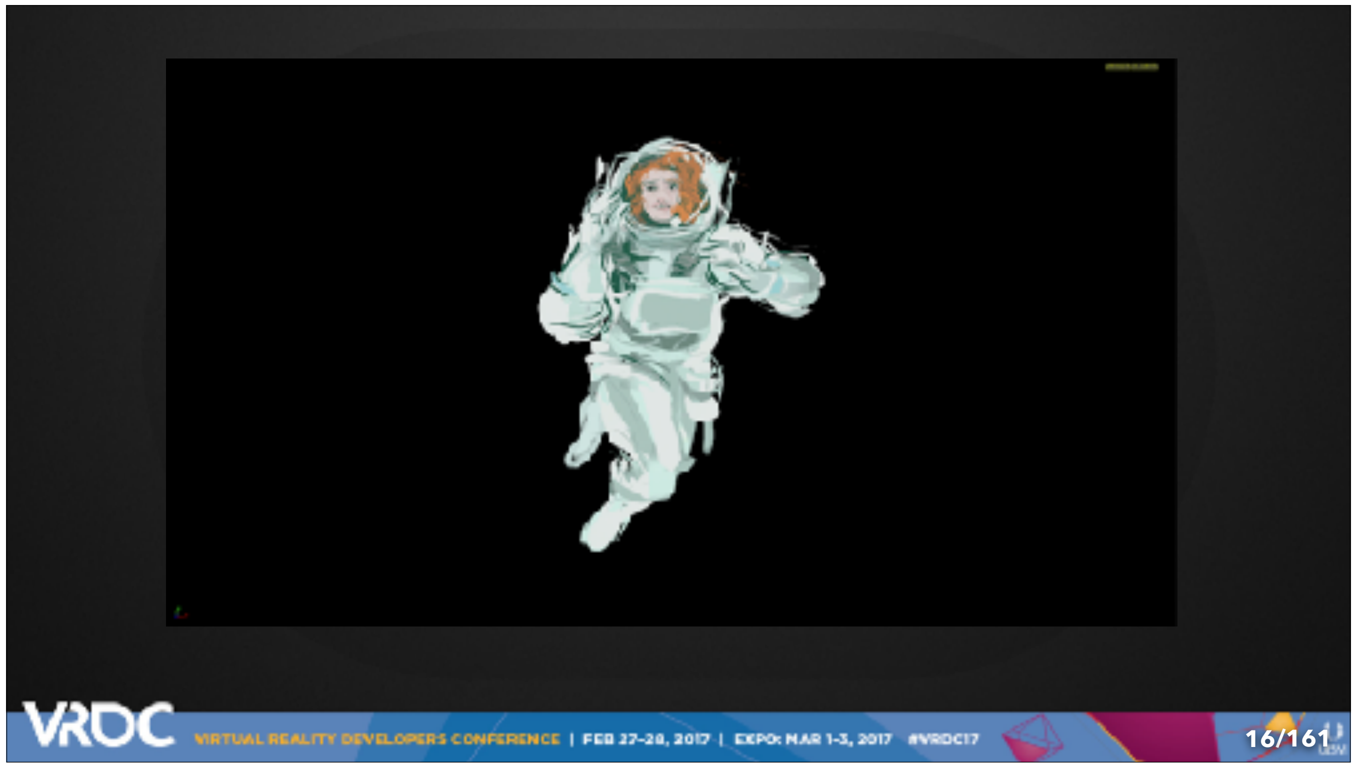
VRDC VIRTUAL REALITY DEVELOPERS CONFERENCE | FEB 27-28, 2017 | EXPO: MAR 1-3, 2017 #VRDC17

14/161

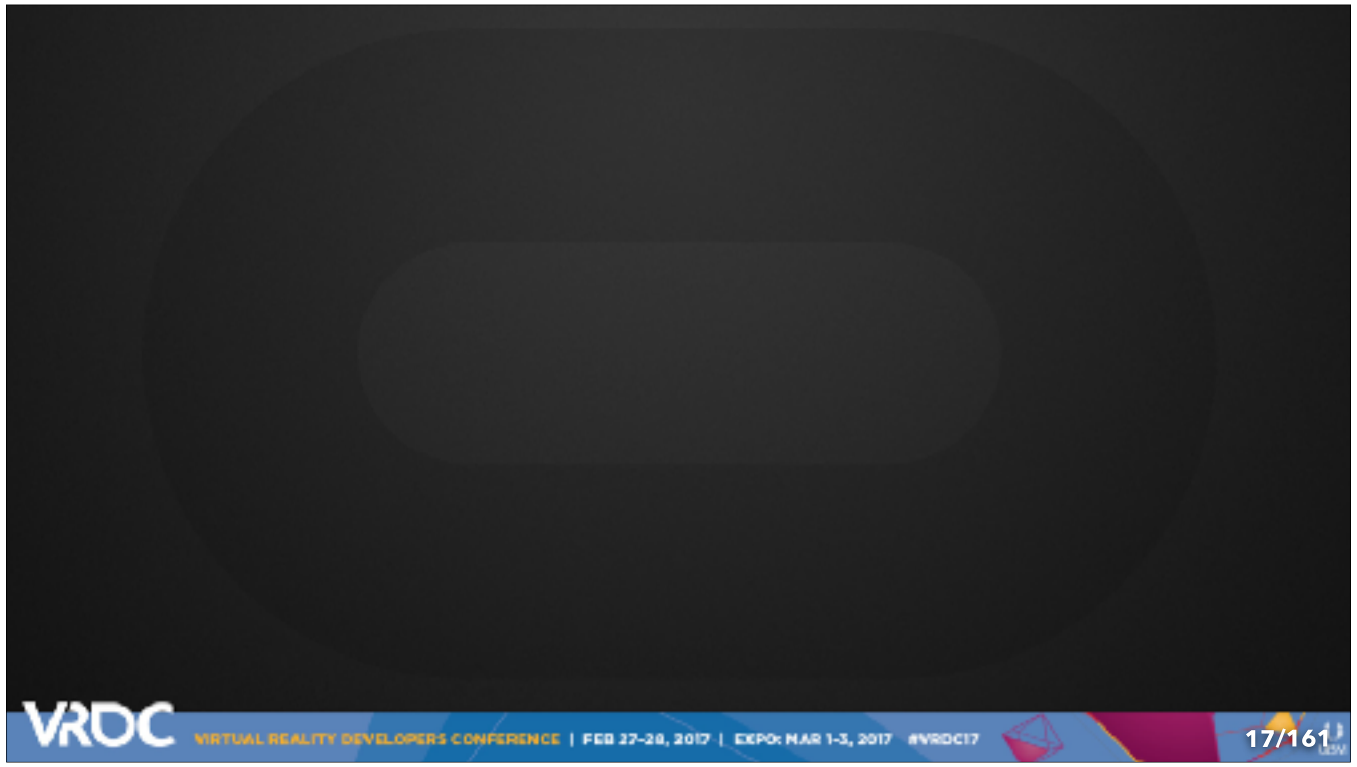
We tried textures and flow maps.


VRDC
VIRTUAL REALITY DEVELOPERS CONFERENCE | FEB 27-28, 2017 | EXPO: MAR 1-3, 2017 #VRDC17
15/161

We tried lighting.

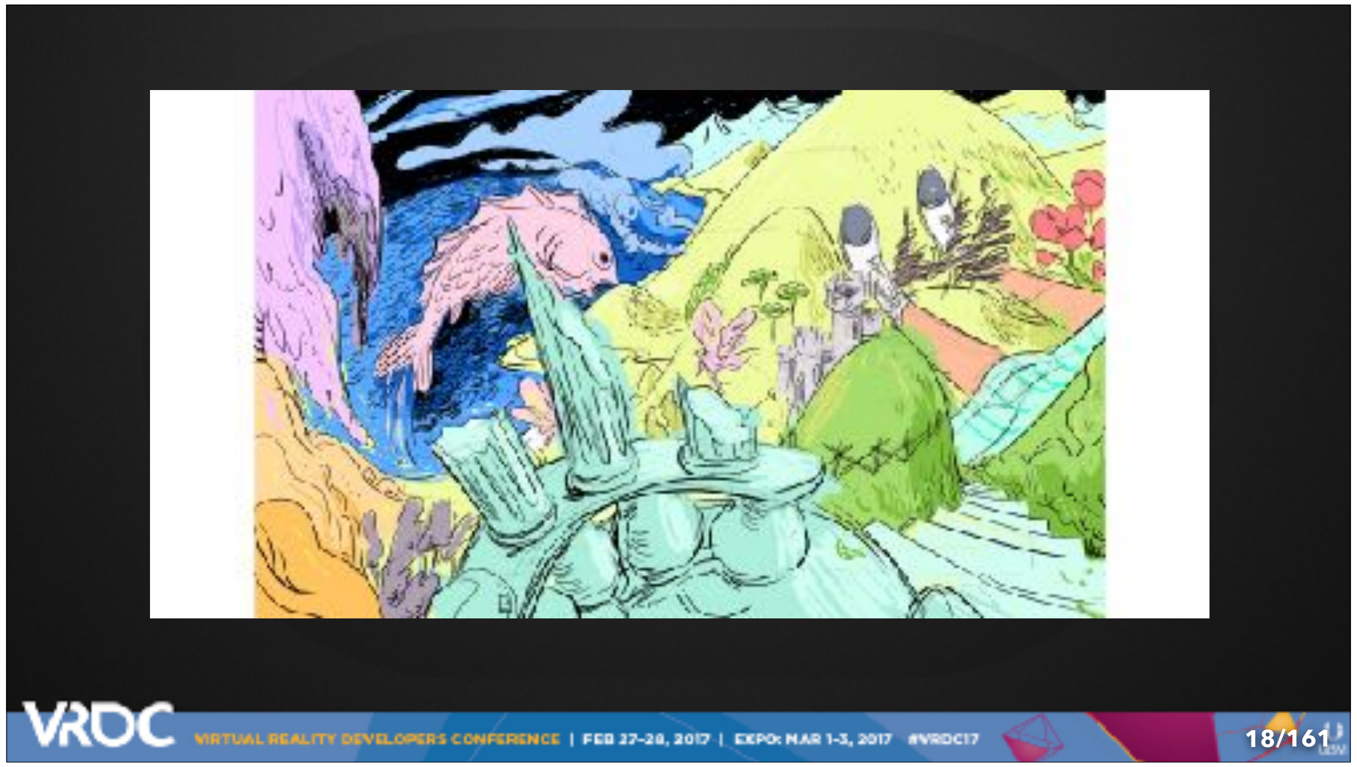

VRDC
VIRTUAL REALITY DEVELOPERS CONFERENCE | FEB 27-28, 2017 | EXPO: MAR 1-3, 2017 #VRDC17
16/161

We tried high-fidelity animation.


VIRTUAL REALITY DEVELOPERS CONFERENCE | FEB 27-28, 2017 | EXPO: MAR 1-3, 2017 #VRDC17

17/161

None of it was working.

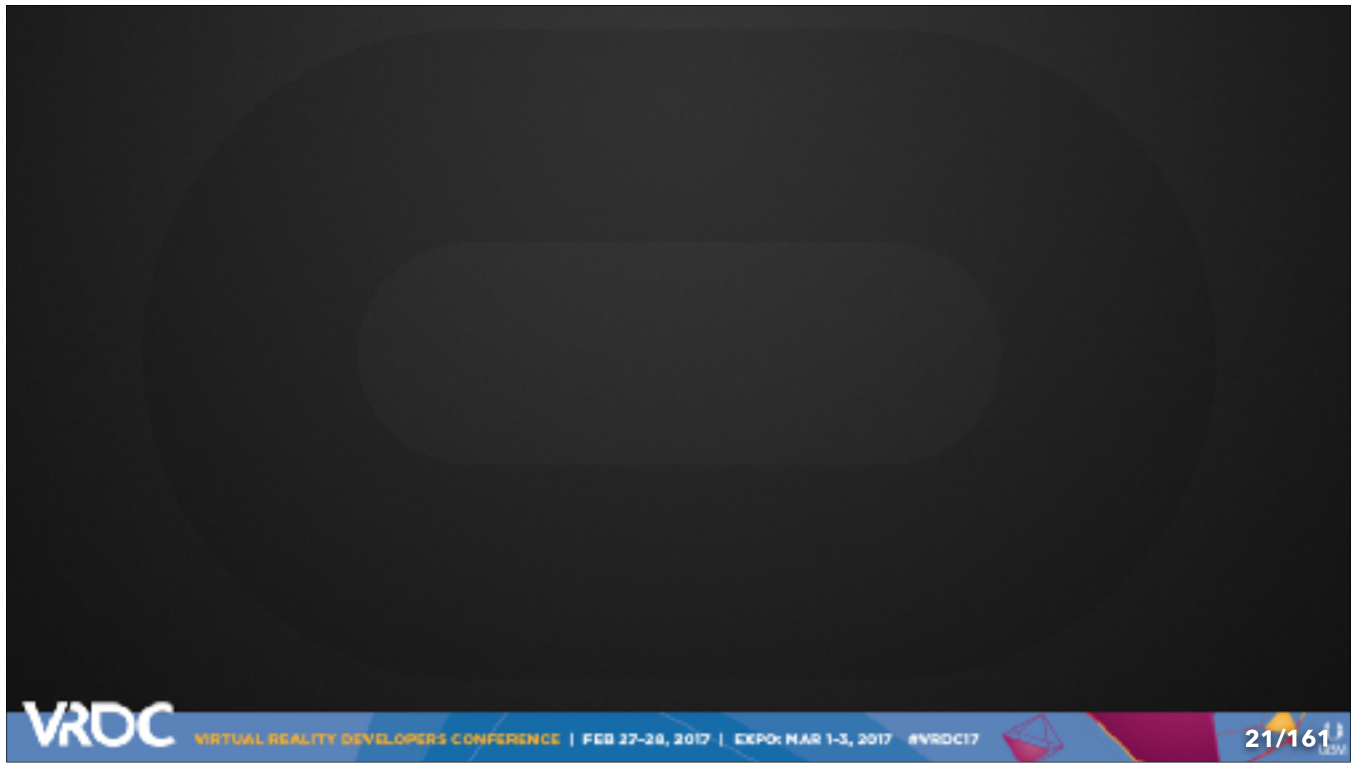

VRDC
VIRTUAL REALITY DEVELOPERS CONFERENCE | FEB 27-28, 2017 | EXPO: MAR 1-3, 2017 #VRDC17
18/161

The problem was - none of it FELT like a drawing. You could always tell it was a 3d model with a texture slapped on it. It didn't have the looseness or the depth or the artistic opinion that a drawing would have. Lighting was too smooth and felt "computery" - and animation wasn't chunky enough to have that stopmotion feel. Then of course

VRDC
VIRTUAL REALITY DEVELOPERS CONFERENCE | FEB 27-28, 2017 | EXPO: MAR 1-3, 2017 #VRDC17
19/161

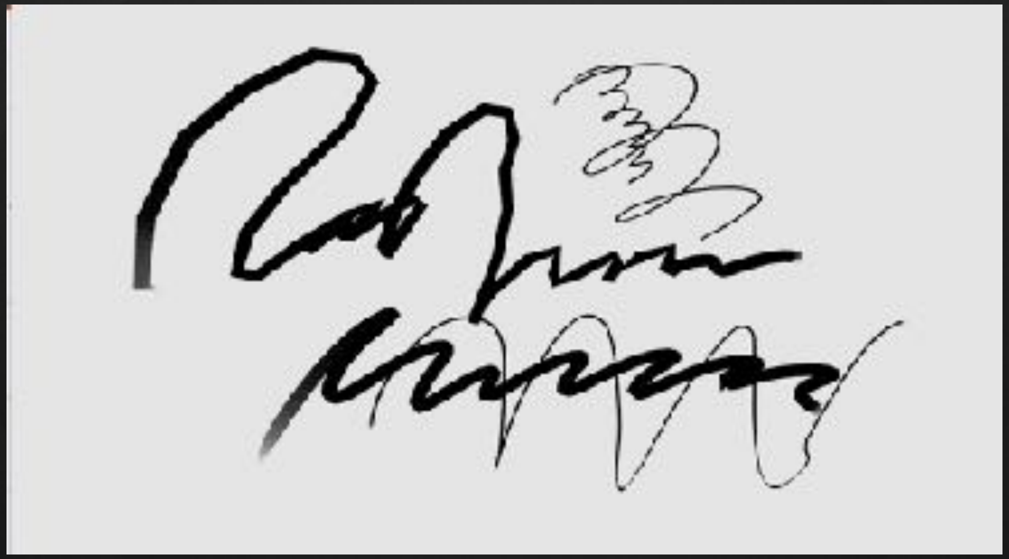
we got one of the first prototypes of Oculus Touch. Until that point I had never tried out a tracked controller - so admittedly I was just trying out weird ideas. For some reason I had made a pepper shaker in Unreal that would spit particles all over the place. It was oddly fun so people crowded around to try it out.

That's about when our FX lead, Robert, asked “what if we turned off gravity?!”


VRDC
VIRTUAL REALITY DEVELOPERS CONFERENCE | FEB 27-28, 2017 | EXPO: MAR 1-3, 2017 #VRDC17
21/161

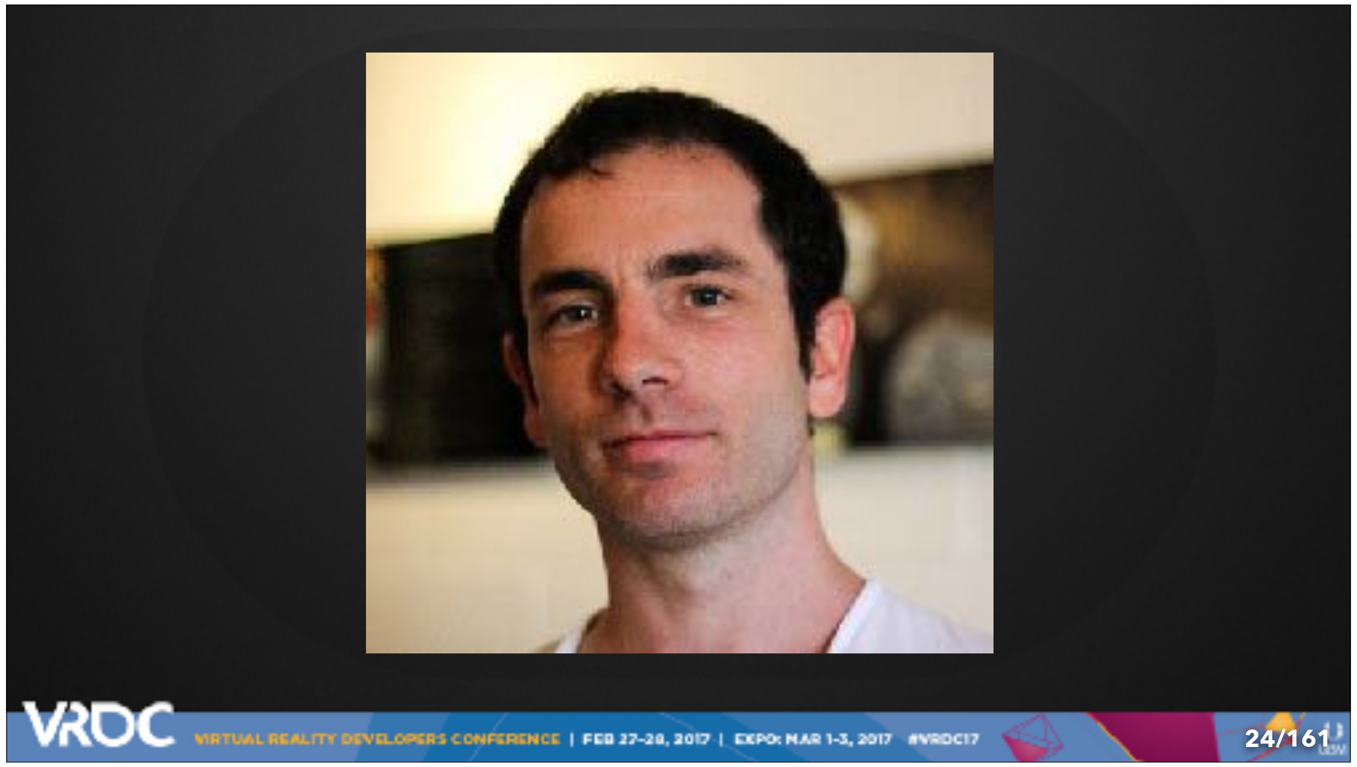
so we turned off gravity

Then he asked “What if we connected the particles together like ribbons?!”

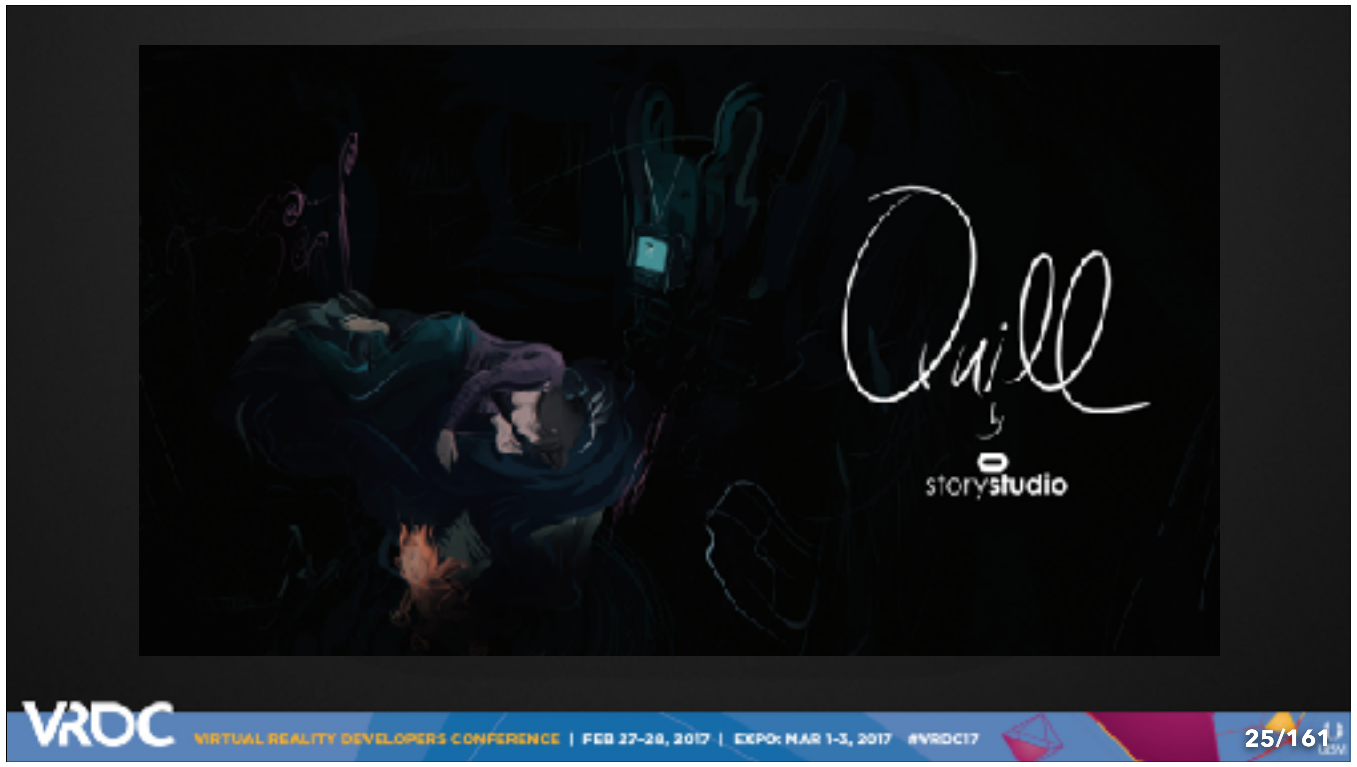


VRDC VIRTUAL REALITY DEVELOPERS CONFERENCE | FEB 27-28, 2017 | EXPO: MAR 1-3, 2017 #VRDC17

so we connected the particles together like ribbons. And instantly, accidentally, we were drawing in VR. We started goofing around and thats when


VRDC
VIRTUAL REALITY DEVELOPERS CONFERENCE | FEB 27-28, 2017 | EXPO: MAR 1-3, 2017 #VRDC17
24/161

Inigo took one look at this, said “AH HA! I HAVE AN IDEA!” and scurried off. Now - if you ever get to work with Inigo and he says that - you just stay out of the way. Don’t disturb him. Because what he had realized was that the solution to illustration in VR was to actually *illustrate in VR*. And 2 days later

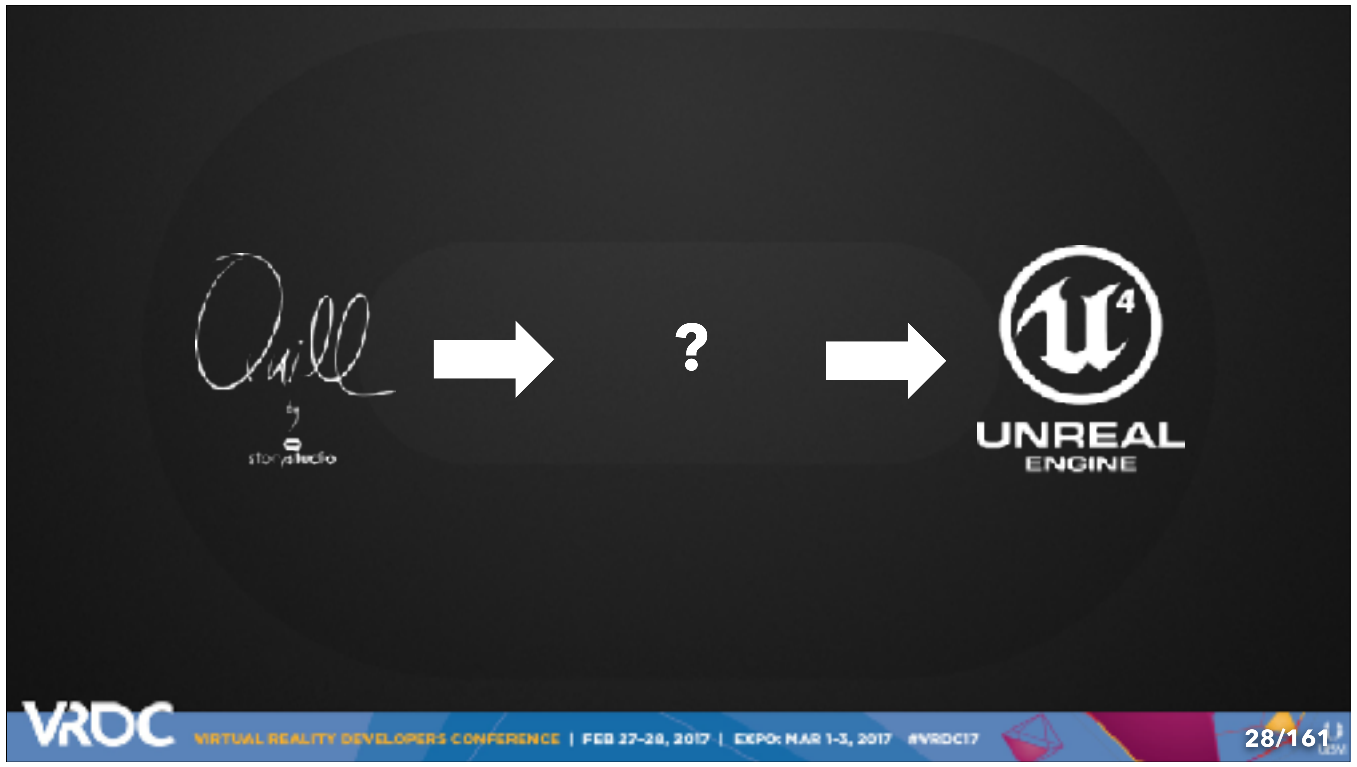

Quill
by
storystudio
VRDC VIRTUAL REALITY DEVELOPERS CONFERENCE | FEB 27-28, 2017 | EXPO: MAR 1-3, 2017 #VRDC17
25/161

he proved that to us with the first version of Quill. From this we eventually built, from the ground up, a standalone OpenGL application that had all the advanced features we would need to make a film - like layers, file exports, rich color controls, and a very precise feel to the brushes. After trying it out - everyone on the team was immediately convinced that this was the answer we had been looking for.

What happens when you suddenly, accidentally, create the solution to all of your problems?

What happens when you suddenly, accidentally, create the solution to all of your problems?

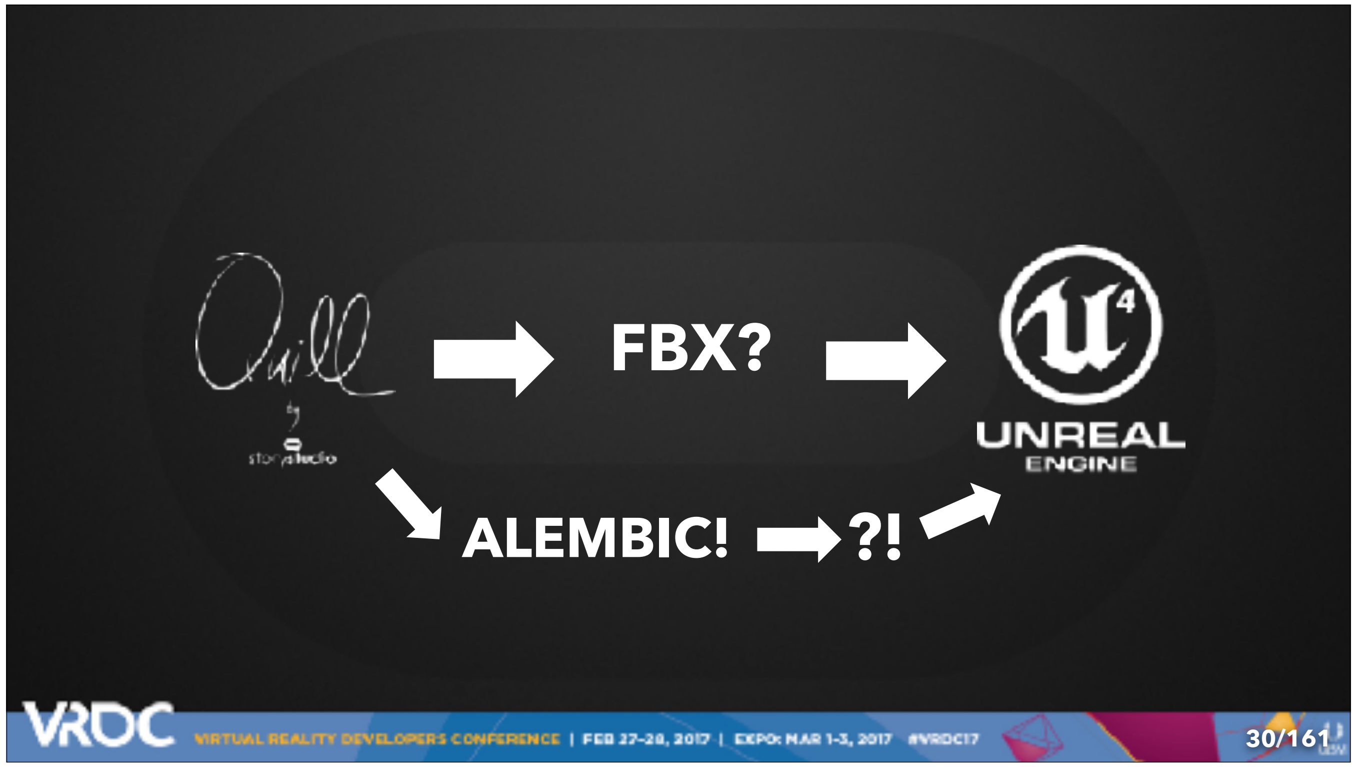
You create a whole lot of new problems.


Quill
by
story studio
?
UNREAL
ENGINE
VRDC
VIRTUAL REALITY DEVELOPERS CONFERENCE | FEB 27-28, 2017 | EXPO: MAR 1-3, 2017 #VRDC17
28/161

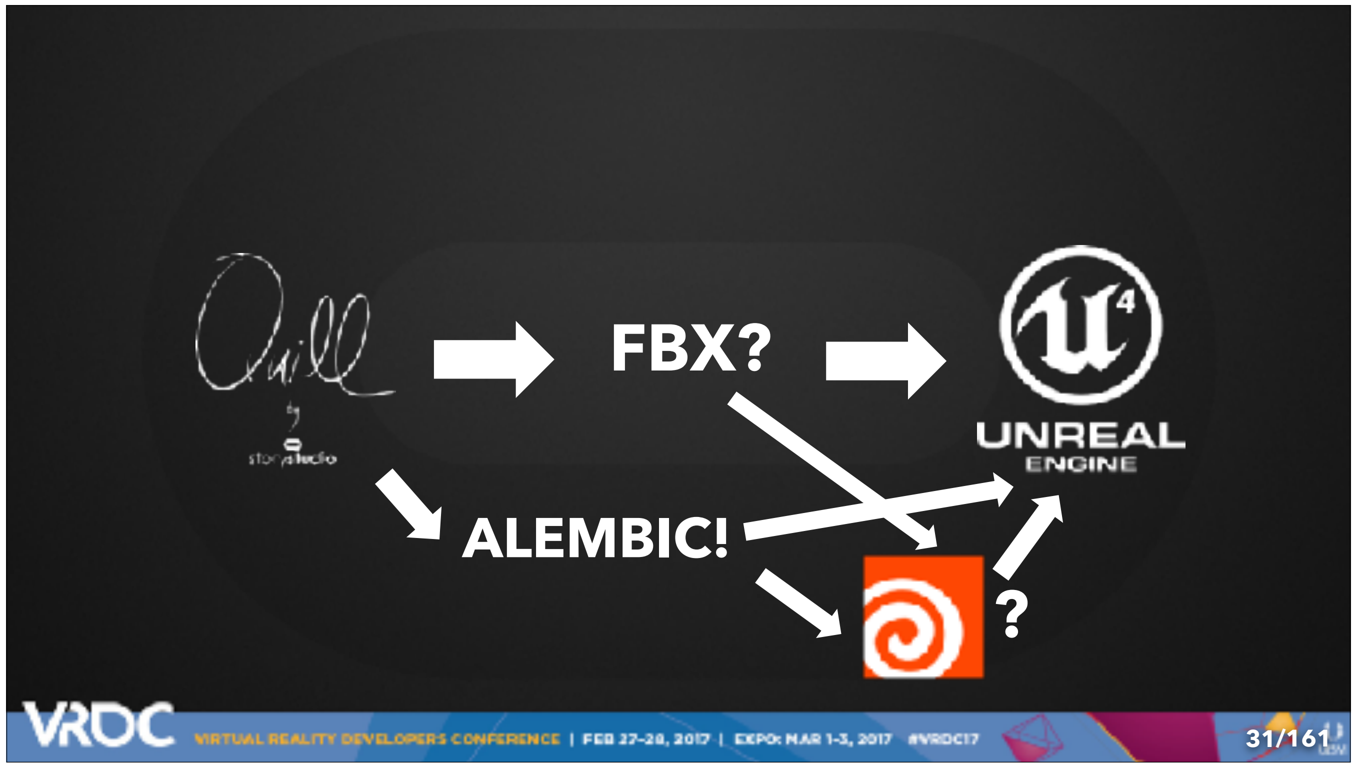
Namely - we’ve got Quill with a new data format and all sorts of different performance characteristics from traditional geometry. We're not quite sure how to package it up. We're not sure what kind of pipeline we need. We're not sure how to best render this either. We only knew two things: that we need to eventually put it into Unreal so we can piece this whole thing together....and that we were already behind schedule. So do you use FBX?

$\rightarrow$ FBX! $\rightarrow$

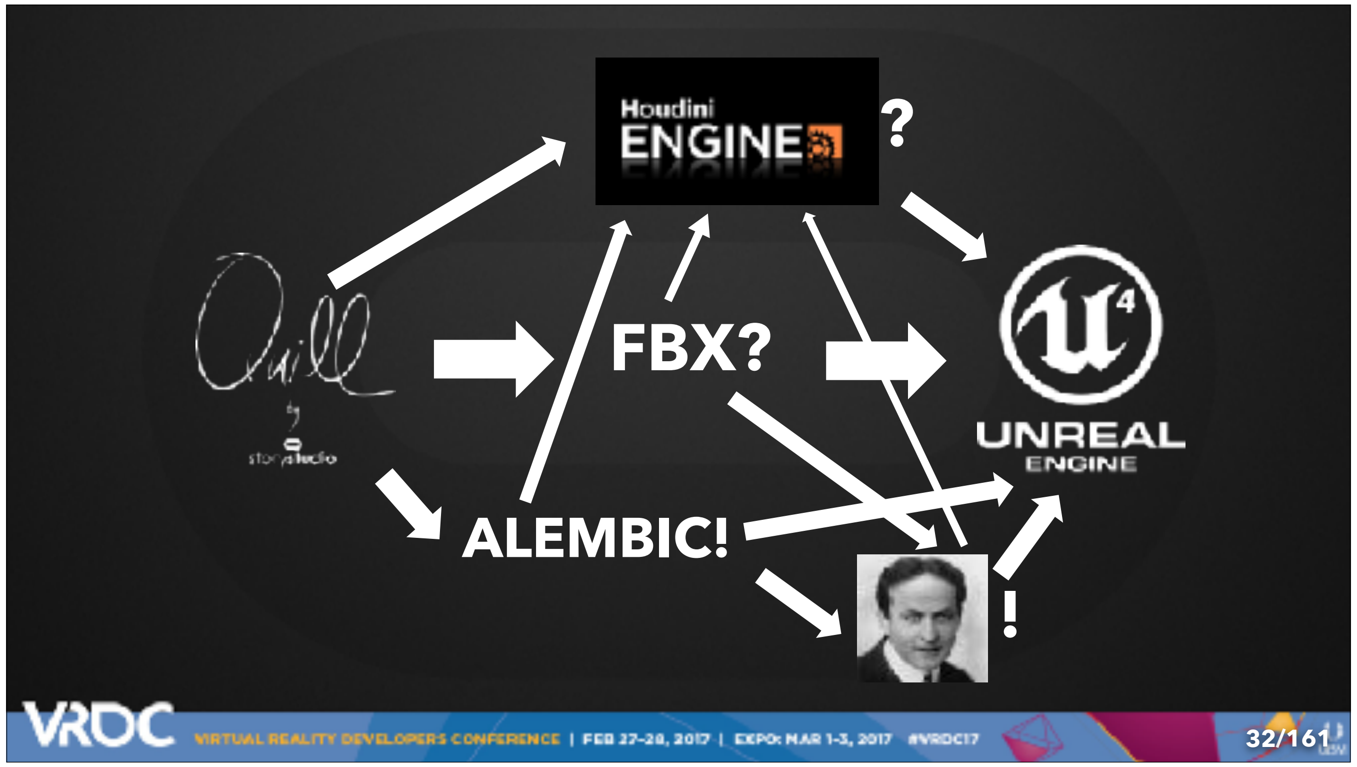
Sure! What about Alembic?



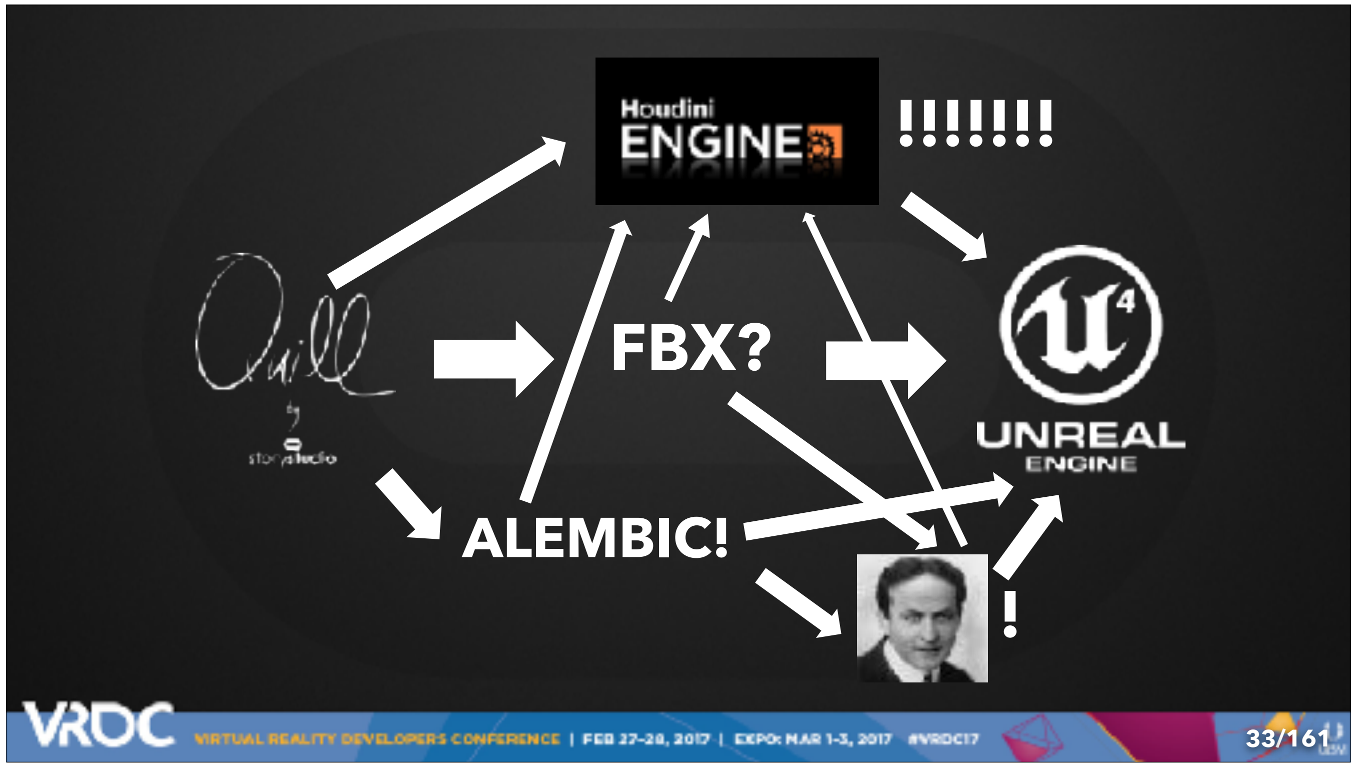
OF COURSE! Where does it go?


FBX?
ALEMBIC!
UNREAL ENGINE
?
VIRTUAL REALITY DEVELOPERS CONFERENCE | FEB 27-28, 2017 | EXPO: MAR 1-3, 2017 #VRDC17
31/161

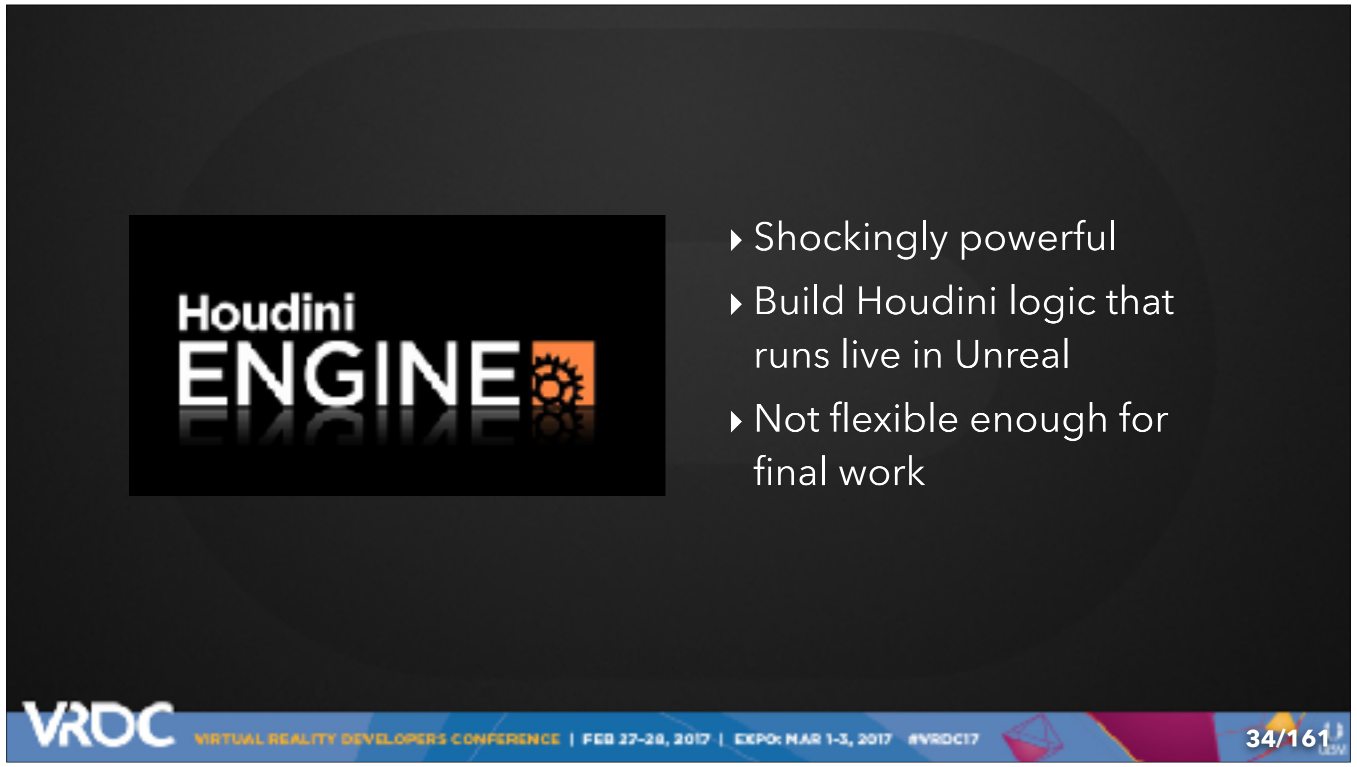
Directly into Unreal! What about Houdini?


Houdini ENGINE
?
Quill by story studio
FBX?
UNREAL ENGINE
ALEMBIC!
!
VRDC VIRTUAL REALITY DEVELOPERS CONFERENCE | FEB 27-28, 2017 | EXPO: MAR 1-3, 2017 #VRDC17
32/161

I LOVE THAT GUY! And Houdini Engine for speed?


Houdini ENGINE
!!!!!!!
Quill by story studio
FBX?
UNREAL ENGINE
ALEMBIC!
!
VRDC VIRTUAL REALITY DEVELOPERS CONFERENCE | FEB 27-28, 2017 | EXPO: MAR 1-3, 2017 #VRDC17
33/161

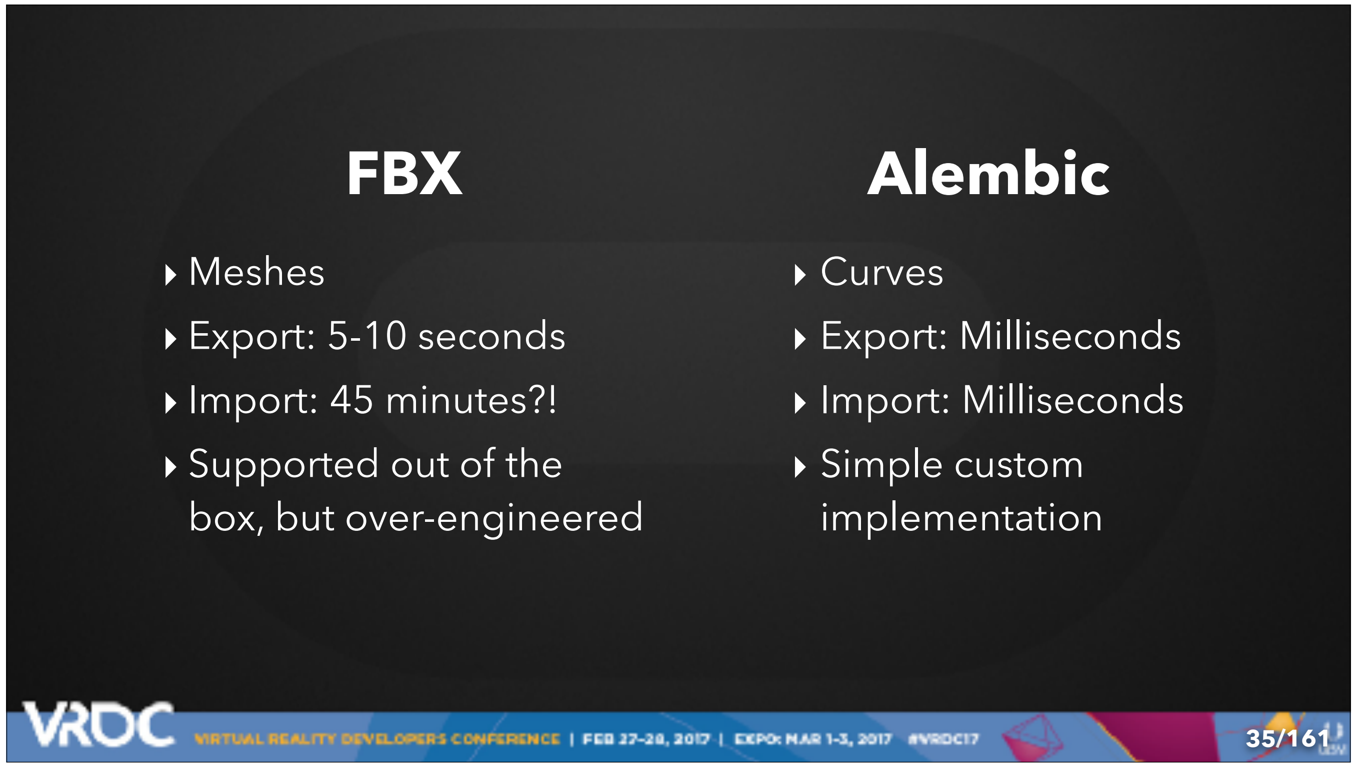
WHY NOT?! And very quickly we had this hilariously awful, slow, and buggy pipeline. We just didn't know what kind of data we would need out of Quill, how any of that data would be used, or how to cope with the sudden flood of ideas that were coming out of the team.



- Shockingly powerful
- Build Houdini logic that runs live in Unreal
- Not flexible enough for final work

VRDC VIRTUAL REALITY DEVELOPERS CONFERENCE | FEB 27-28, 2017 | EXPO: MAR 1-3, 2017 #VRDC17

But eventually we figured it out. At the studio - we love Houdini as it's super powerful for creating simulations and procedurally generating content. There's a reason that almost everyone in film uses it. The ability to run custom Houdini logic live in Unreal using Houdini Engine was too good to pass up. Houdini Engine turned out to be great for the more generic draw-in draw-out work during previz - but our shot work ended up being so specific that we ended up not using it once production started.



FBX

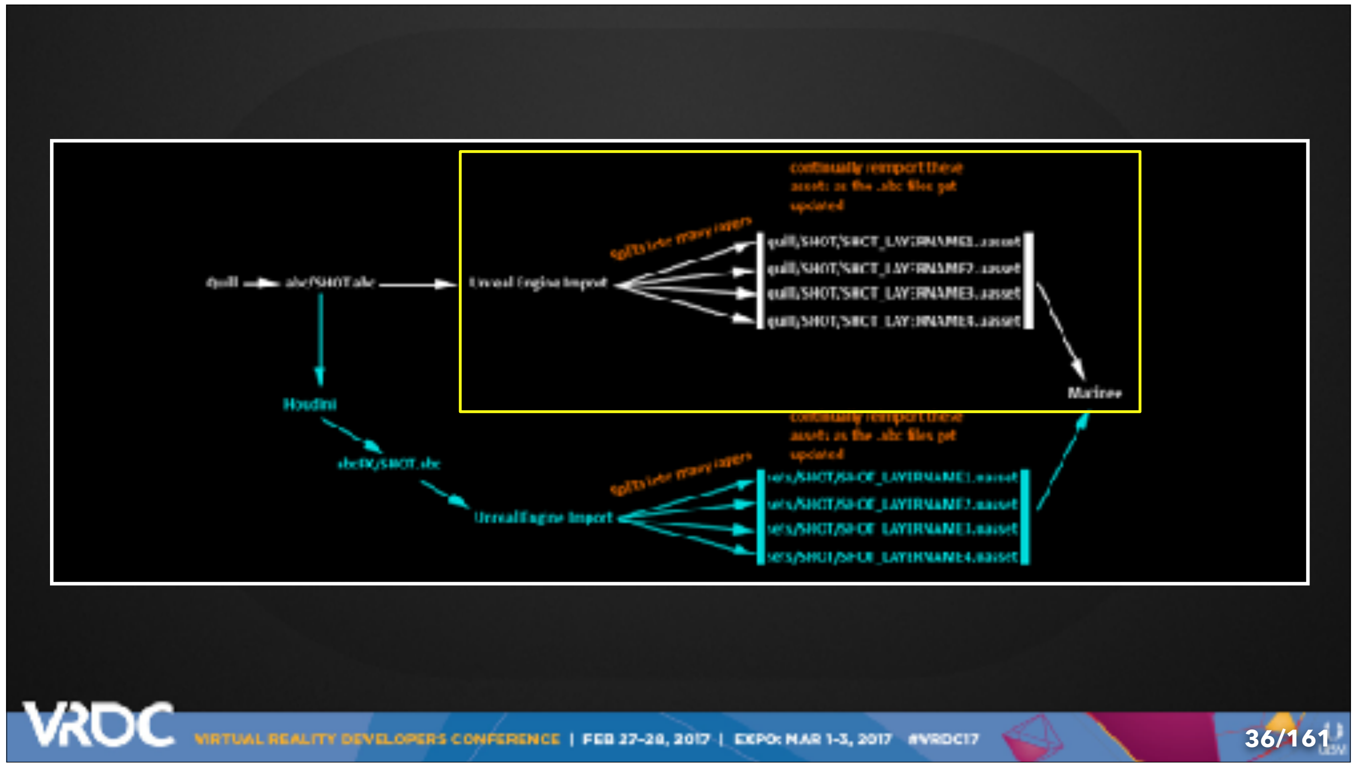
- Meshes
- Export: 5-10 seconds
- Import: 45 minutes?!
- Supported out of the box, but over-engineered

Alembic

- Curves
- Export: Milliseconds
- Import: Milliseconds
- Simple custom implementation

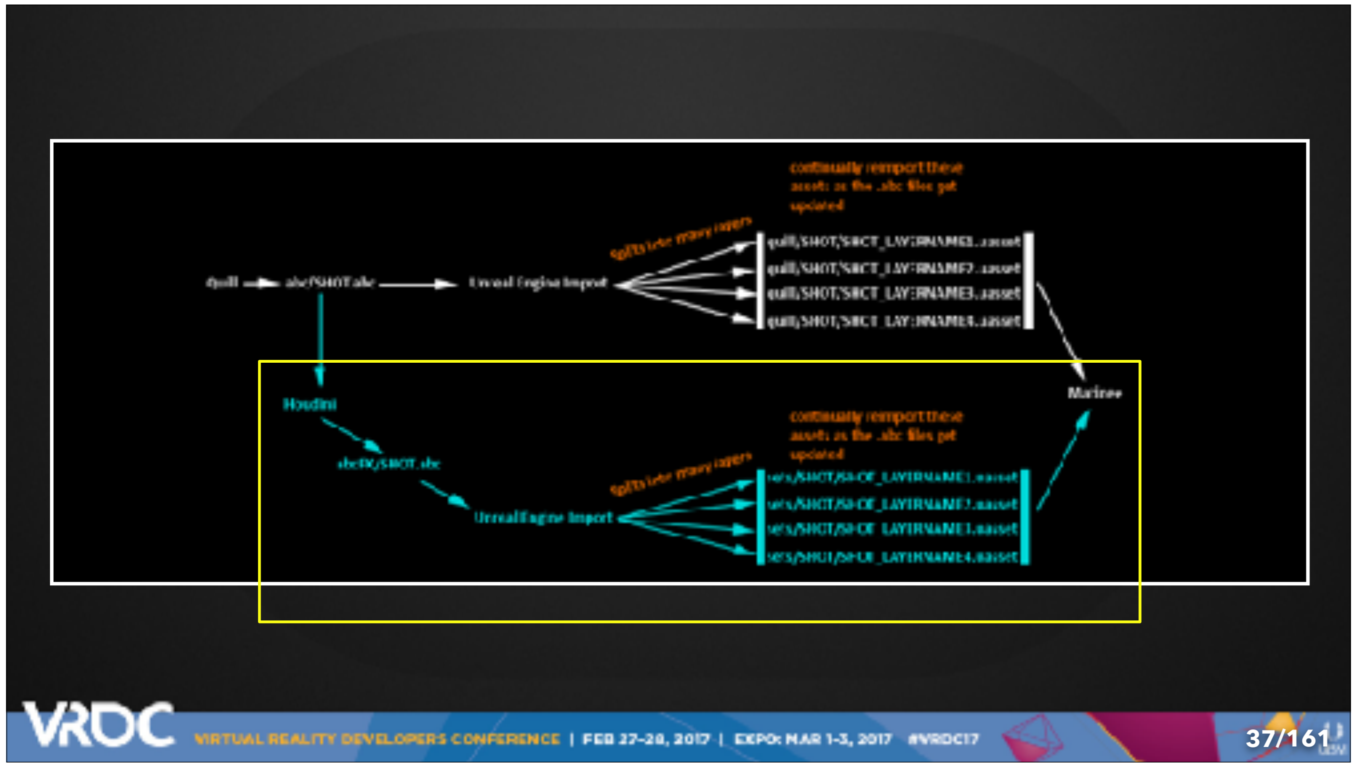
VRDC VIRTUAL REALITY DEVELOPERS CONFERENCE | FEB 27-28, 2017 | EXPO: MAR 1-3, 2017 #VRDC17

We also figured out that using FBX for Quill meshes was a bad idea. The type of geometry Quill cranks out was taking 5-10 seconds to export to FBX. Then these FBX files easily ended up taking 45 minutes to an hour to load into Unreal. When we switched to curves and alembic, exports and imports went down to milliseconds. The first time I exported an alembic file I thought it didn't work because it was so quick.

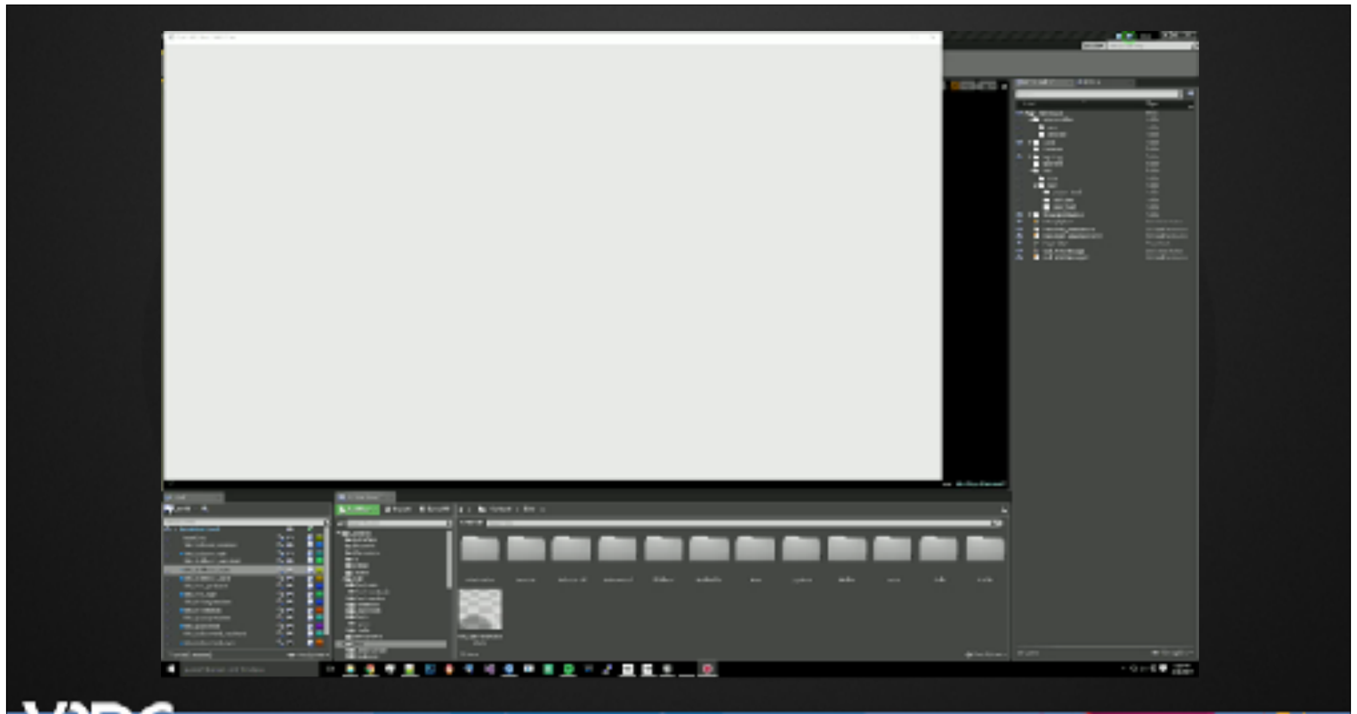


Quill
abc/SHOT.abc
Unreal Engine Import
splits into many layers
continually reimport these assets as the .abc files get updated
quill/SHOT/SHOT_LAYERNAME1.uasset
quill/SHOT/SHOT_LAYERNAME2.uasset
quill/SHOT/SHOT_LAYERNAME3.uasset
quill/SHOT/SHOT_LAYERNAME4.uasset
Matinee
Houdini
abcFX/SHOT.abc
Unreal Engine Import
splits into many layers
continually reimport these assets as the .abc files get updated
sets/SHOT/SHOT_LAYERNAME1.uasset
sets/SHOT/SHOT_LAYERNAME2.uasset
sets/SHOT/SHOT_LAYERNAME3.uasset
sets/SHOT/SHOT_LAYERNAME4.uasset
VRDC VIRTUAL REALITY DEVELOPERS CONFERENCE | FEB 27-28, 2017 | EXPO: MAR 1-3, 2017 #VRDC17
36/161

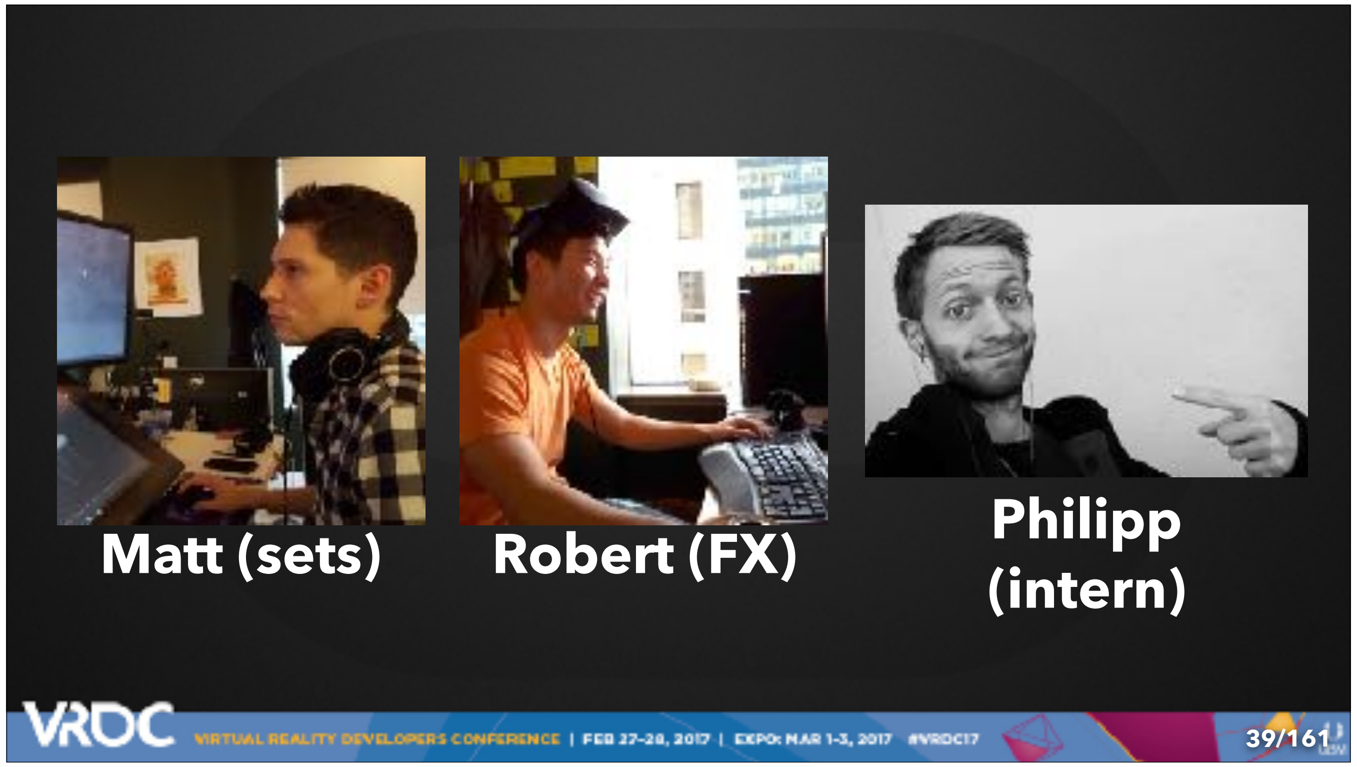
So to short circuit your own possible misery - this is the pipeline we landed on: Quill exports the entire scene into alembic with a ton of metadata about the curves. From there many of our smaller assets were read directly into Unreal, split into layers and wired into Matinee. Using naming conventions - we could retain links between the layers in alembic and the assets in Unreal. That way we could just continually refresh the files as needed.

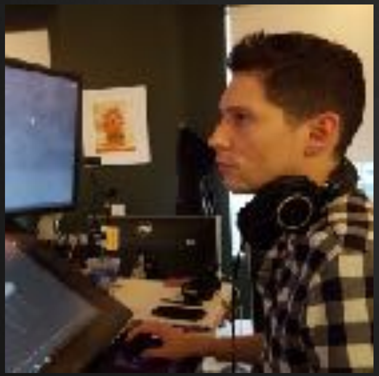

Quill
abc/SHOT.abc
Unreal Engine Import
splits into many layers
continually reimport these assets as the .abc files get updated
quill/SHOT/SHOT_LAYERNAME1.uasset
quill/SHOT/SHOT_LAYERNAME2.uasset
quill/SHOT/SHOT_LAYERNAME3.uasset
quill/SHOT/SHOT_LAYERNAME4.uasset
Matinee
Houdini
abcFX/SHOT.abc
Unreal Engine Import
splits into many layers
continually reimport these assets as the .abc files get updated
sets/SHOT/SHOT_LAYERNAME1.uasset
sets/SHOT/SHOT_LAYERNAME2.uasset
sets/SHOT/SHOT_LAYERNAME3.uasset
sets/SHOT/SHOT_LAYERNAME4.uasset
VRDC
VIRTUAL REALITY DEVELOPERS CONFERENCE | FEB 27-28, 2017 | EXPO: MAR 1-3, 2017 #VRDC17
37/161

Our larger assets were diverted off along this bottom path into Houdini for some custom FX work. In Houdini we would do things like modify the draw-in timing, move curves, delete curves, or generate procedural animation. These modified assets were then dropped into alembic files that got imported into Unreal and wired into Matinee. To show you how this looks - I created a quick video



This is all real time - no editing. [video of pipeline]





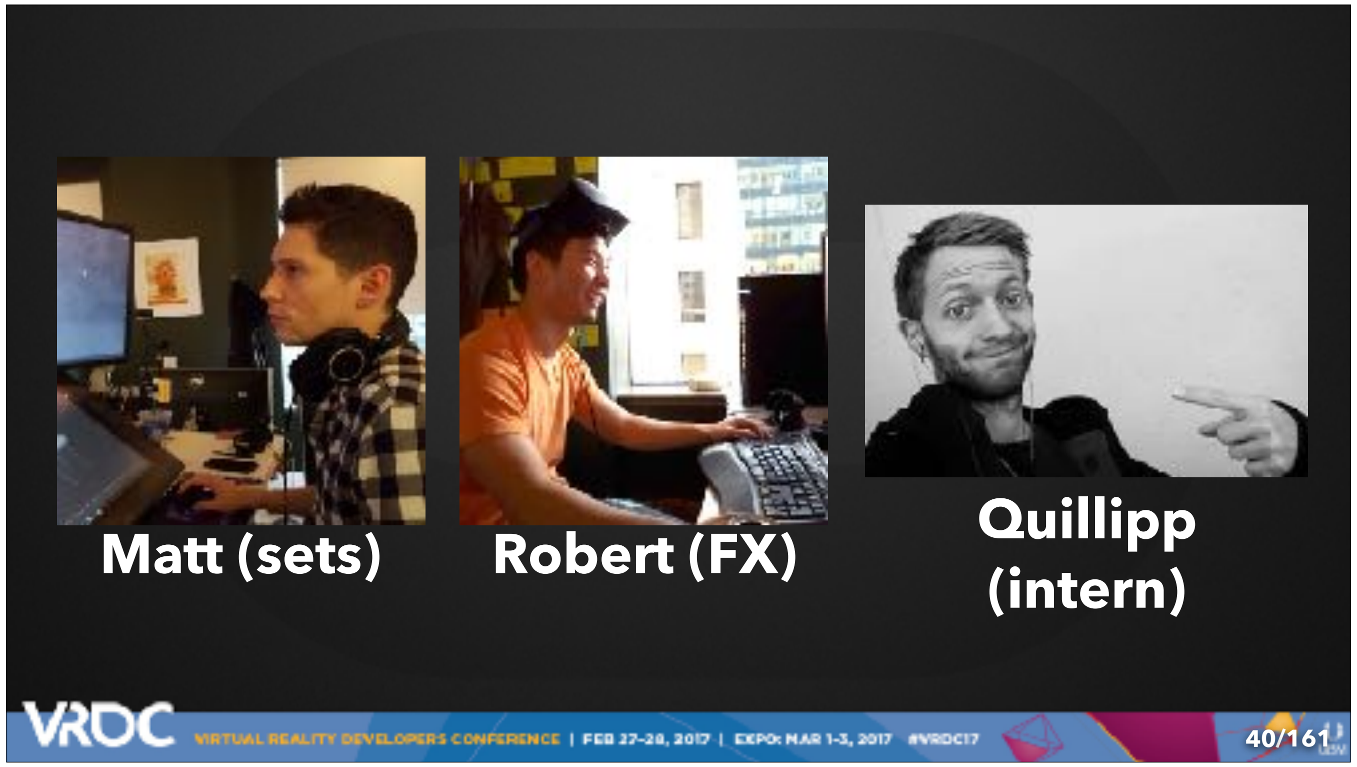
Matt (sets)

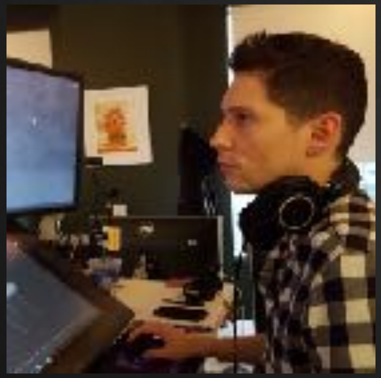
Robert (FX)

Philipp (intern)

The joy here is manyfold:

1.: We've created a very tight iteration loop so that people can quickly try out their ideas in 360. Normally on a production you would have to get a sets team, an FX team, or our intern Philipp - whom we ended up calling



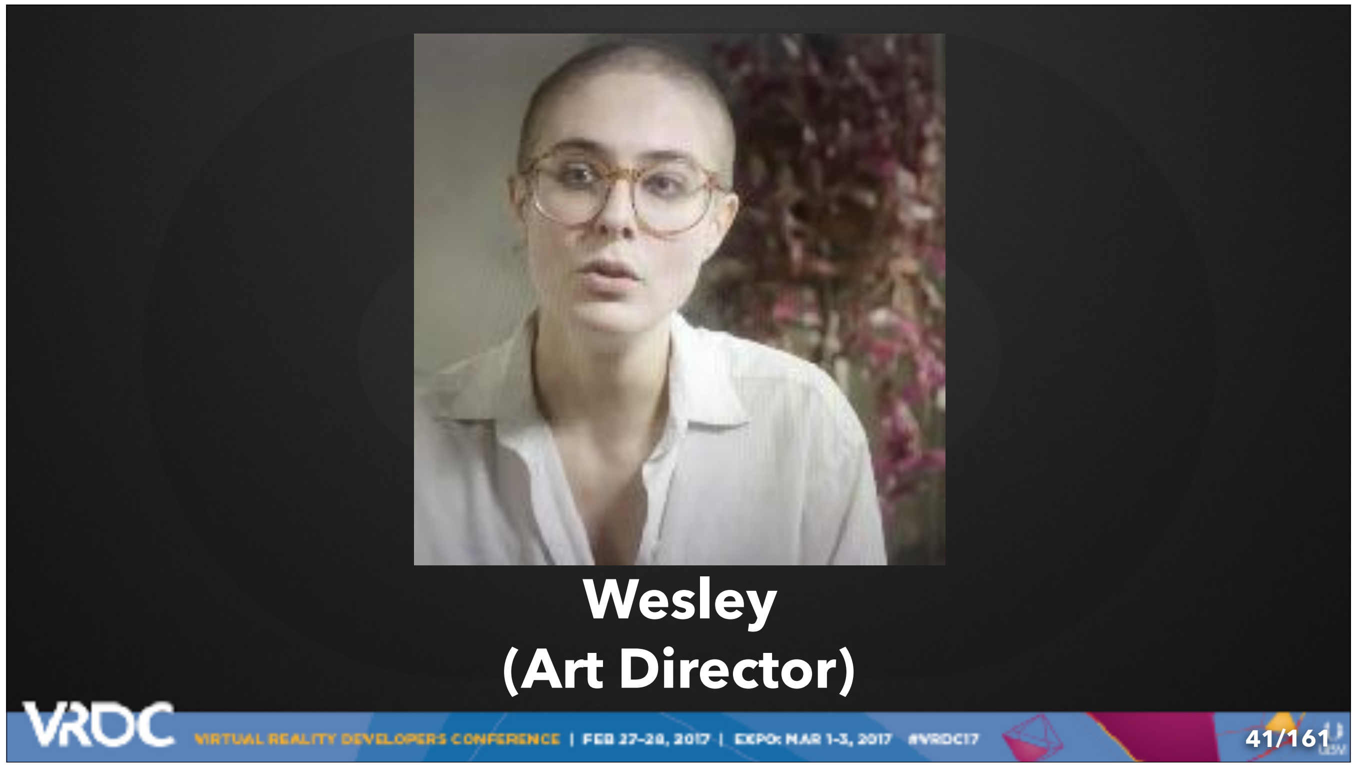


Matt (sets)

Robert (FX)

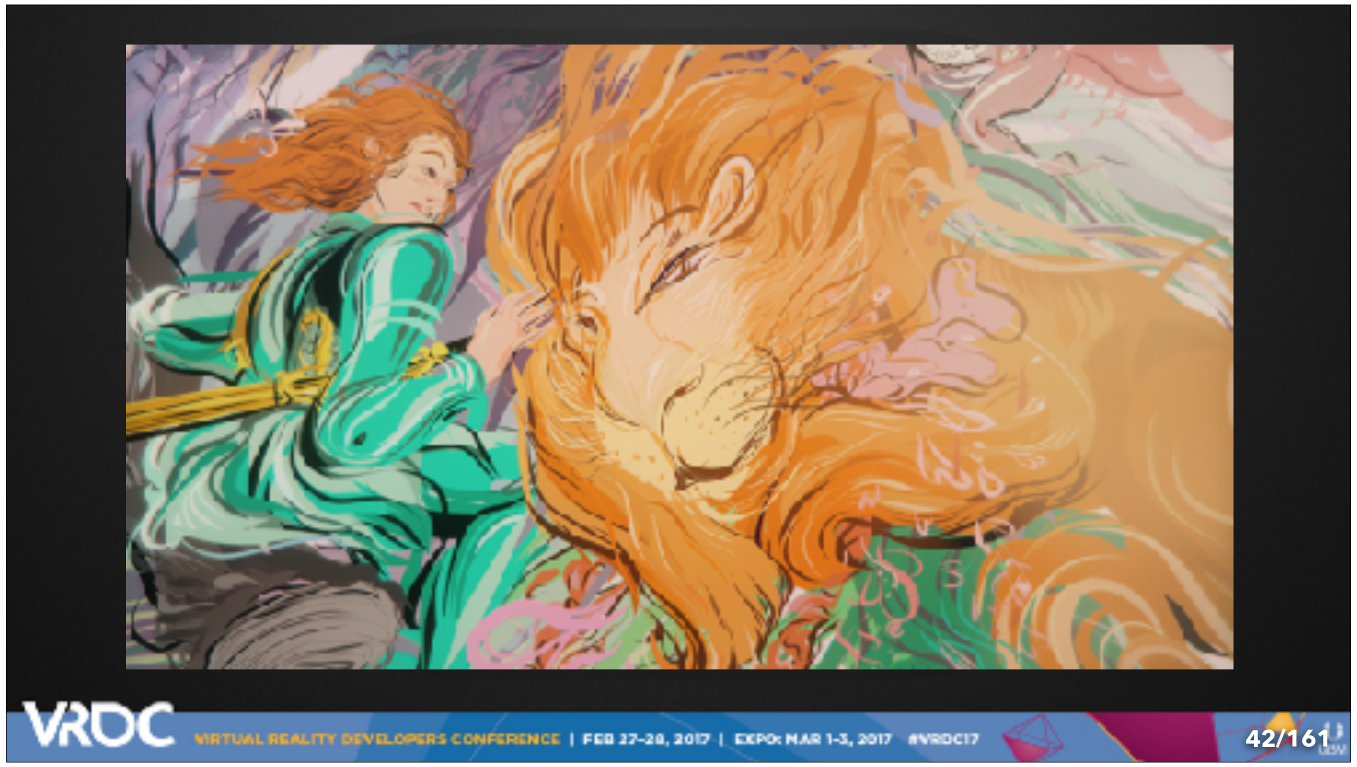
Quillipp (intern)

Quillipp - involved to try out an idea. Thats a lot of people for something that might not pan out.



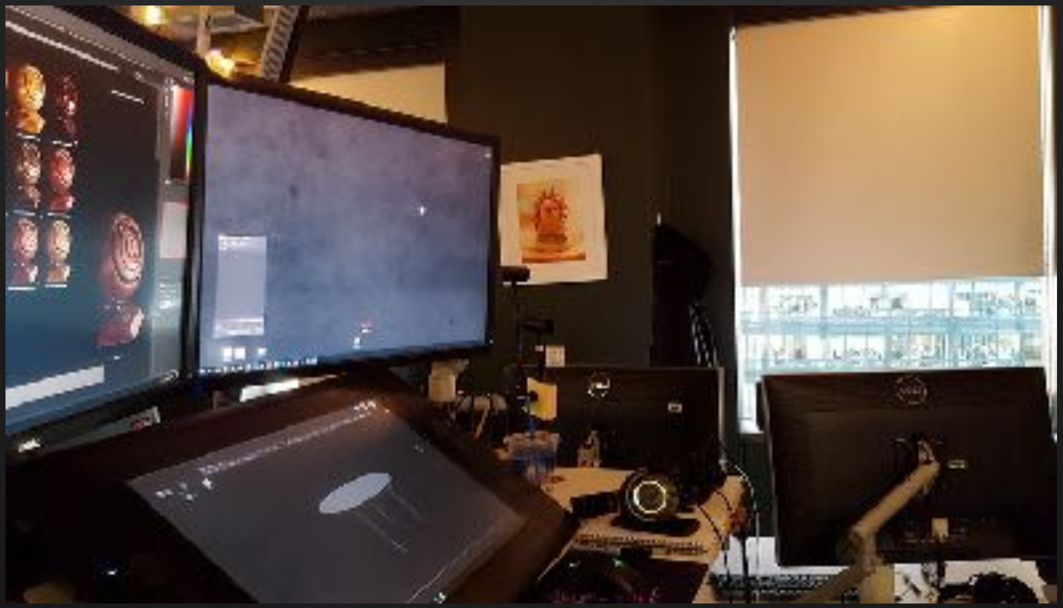
Wesley
(Art Director)

With Quill - Wesley completely by herself could paint, recompose, and generally workshop on an idea live with the director during a working session. It's similar to having Maya, Substance Painter, and Unreal Engine all in the same piece of software.

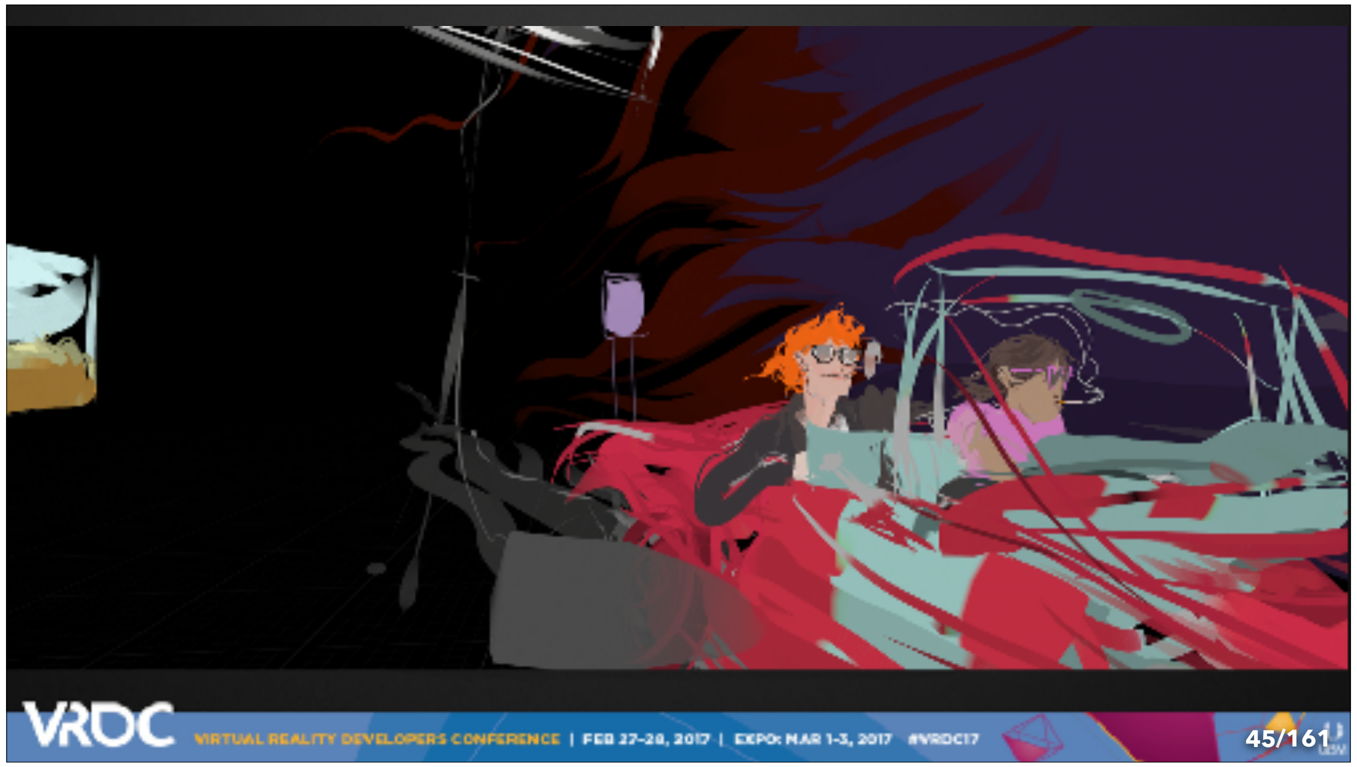

VRDC
VIRTUAL REALITY DEVELOPERS CONFERENCE | FEB 27-28, 2017 | EXPO: MAR 1-3, 2017 #VRDC17
42/161

Plus these decisions were being made in context - in VR - rather than trying to make a decision by looking at it on the monitor. We ended up rarely moving things around in Unreal Engine because it was faster, easier, and correcterer(?!) in Quill.

2 And when I say “didn’t need to get a sets team involved” - what I mean is



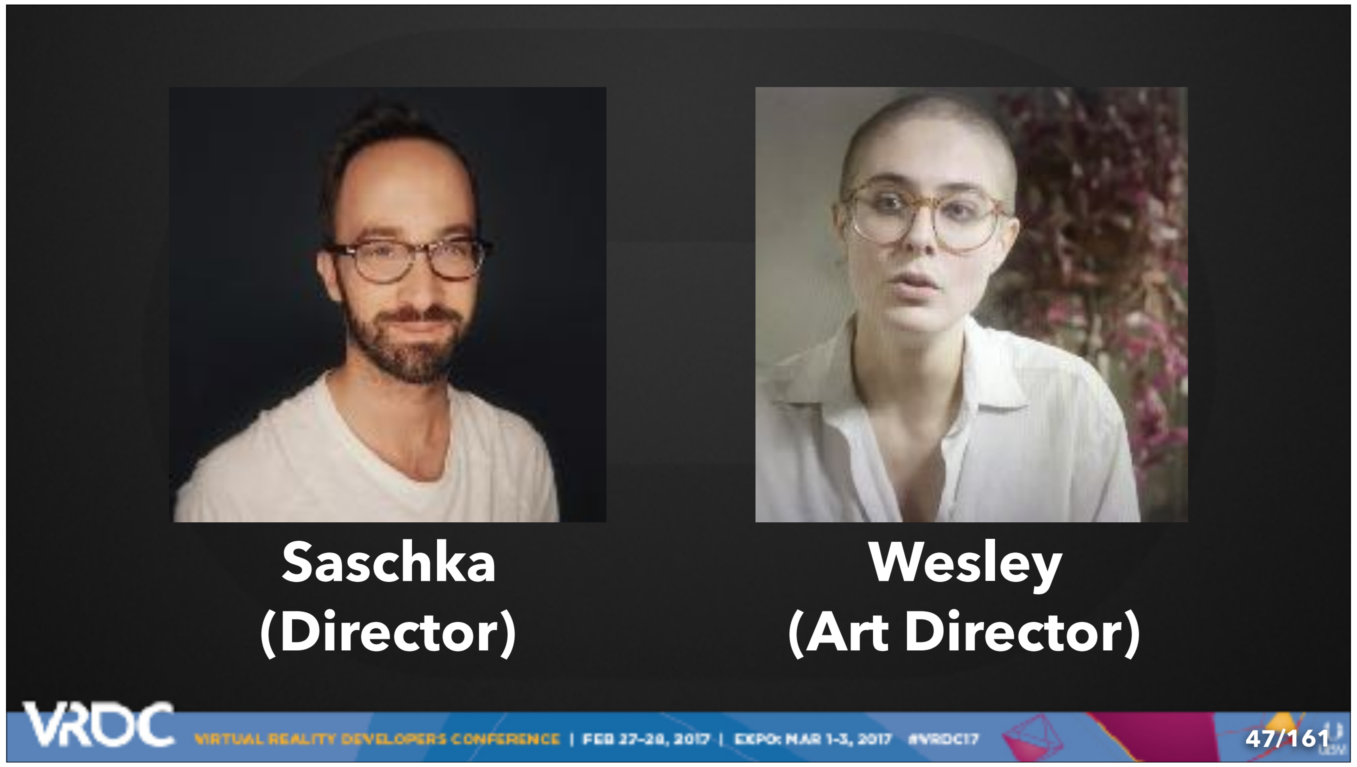
we didn’t have a sets team at all. Once we had Quill, Wesley single-handedly became the art, modeling, shading, and layout departments. She actually painted the entire thing by herself.


VRDC
VIRTUAL REALITY DEVELOPERS CONFERENCE | FEB 27-28, 2017 | EXPO: MAR 1-3, 2017 #VRDC17
45/161

On Dear Angelica the only difference between “concept art” and “final asset” was the amount of time Wesley put into the drawing. And the amount of time per asset was small - she would often delete and redraw whole characters in an afternoon. You would never do that if you were using 3d models as it would be too costly to re-model, re-texture, and re-light the scene

VRDC VIRTUAL REALITY DEVELOPERS CONFERENCE | FEB 27-28, 2017 | EXPO: MAR 1-3, 2017 #VRDC17
46/161

On Dear Angelica the only difference between “concept art” and “final asset” was the amount of time Wesley put into the drawing. And the amount of time per asset was small - she would often delete and redraw whole characters in an afternoon. You would never do that if you were using 3d models as it would be too costly to re-model, re-texture, and re-light the scene

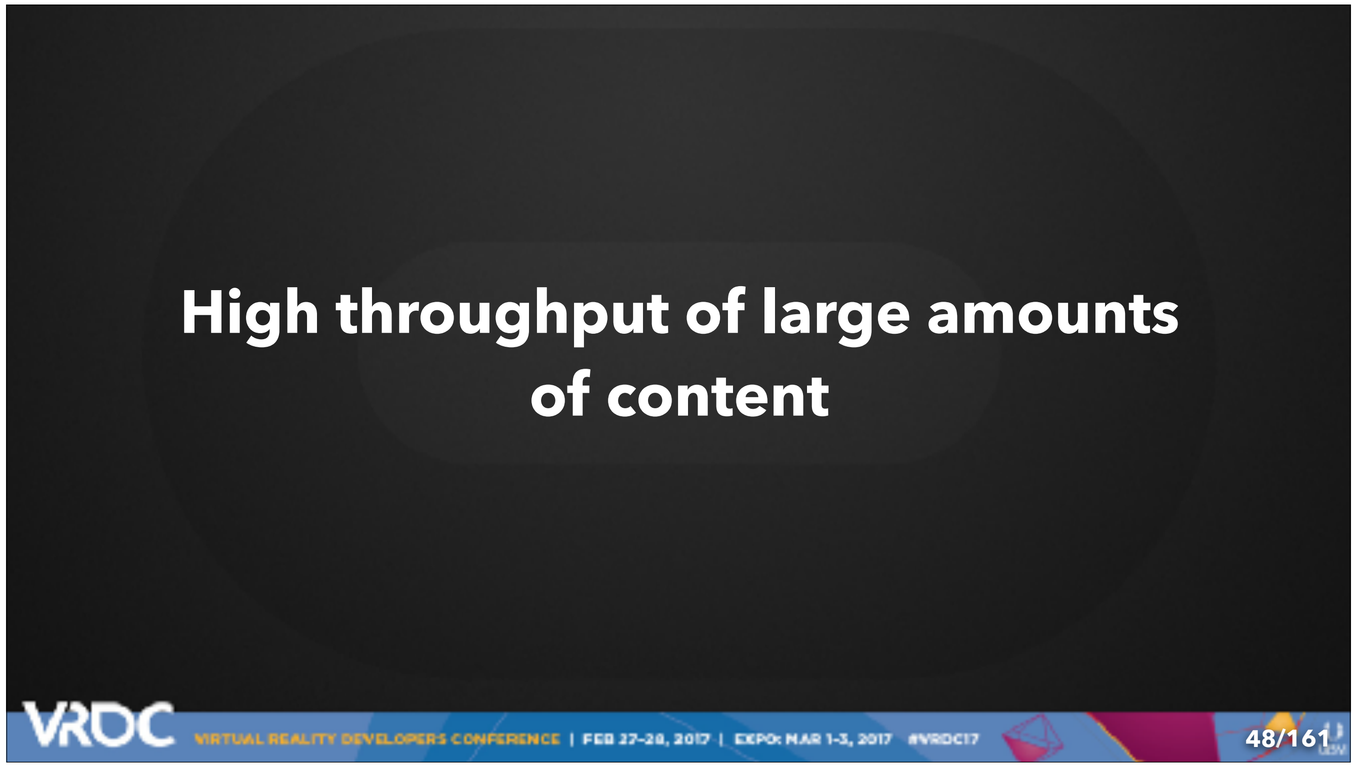


Saschka
(Director)

Wesley
(Art Director)

VRDC VIRTUAL REALITY DEVELOPERS CONFERENCE | FEB 27-28, 2017 | EXPO: MAR 1-3, 2017 #VRDC17

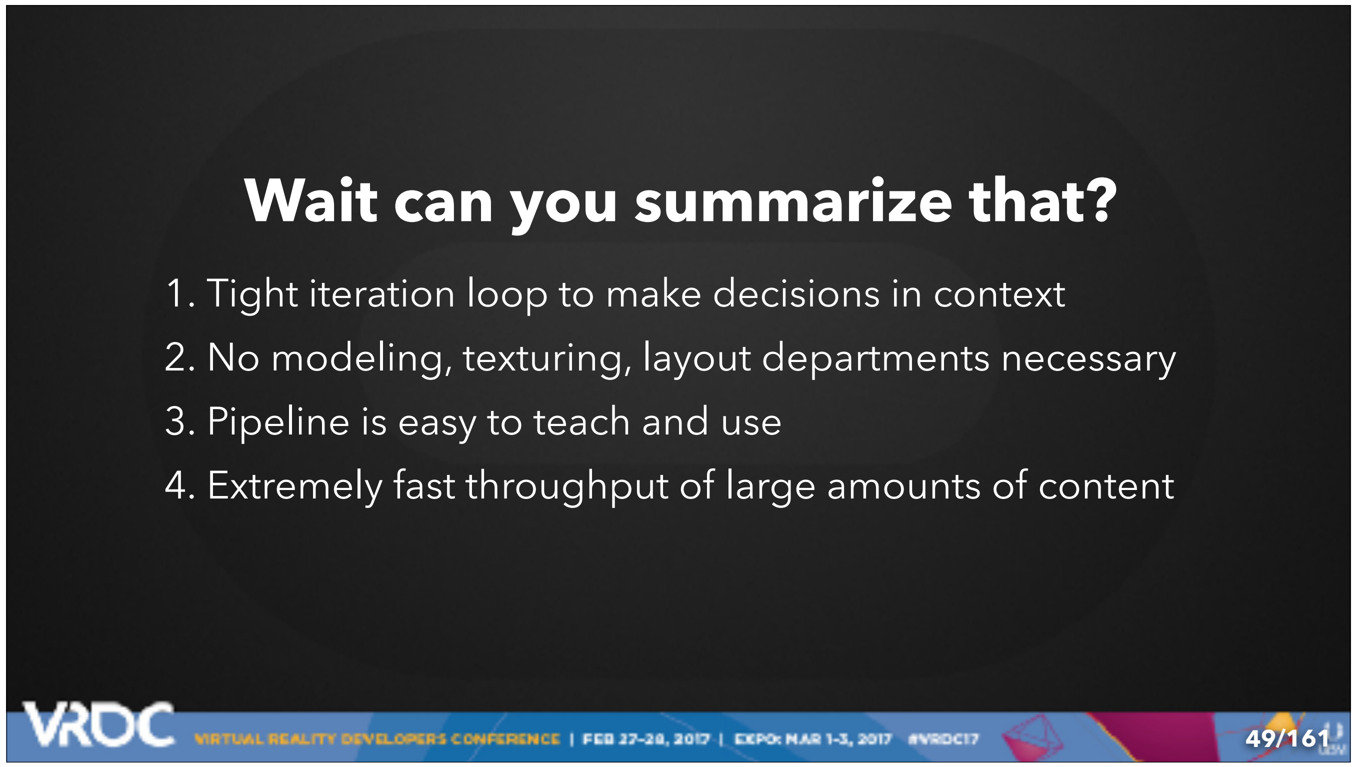
3: We've also created a pipeline that's easy enough for a director and art director to use...completely unsupervised. Which was a scary prospect that gave me a bunch of nightmares which actually came true one weekend. I left work on Friday with one version of the film, and on Monday I walked into a completely different edit of the film with entirely new dialogue and a ton of new assets. And I realized that this nightmare was actually the dream - that VR could actually be a place for creative expression and exploration without all the frustration that had bogged us down previously.



High throughput of large amounts of content

VRDC VIRTUAL REALITY DEVELOPERS CONFERENCE | FEB 27-28, 2017 | EXPO: MAR 1-3, 2017 #VRDC17

4. It allowed us to pump out a huge amount of content in a short amount of time. A little more on this later.

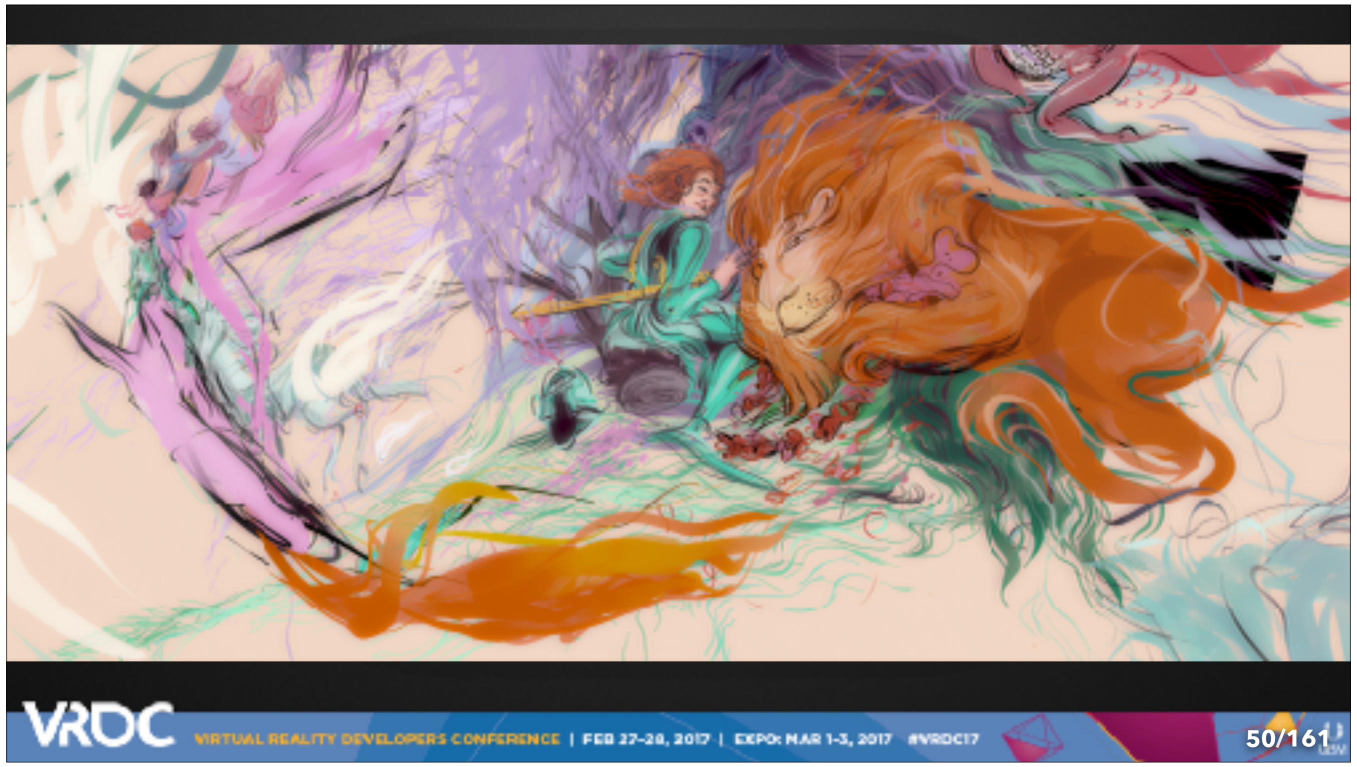


Wait can you summarize that?

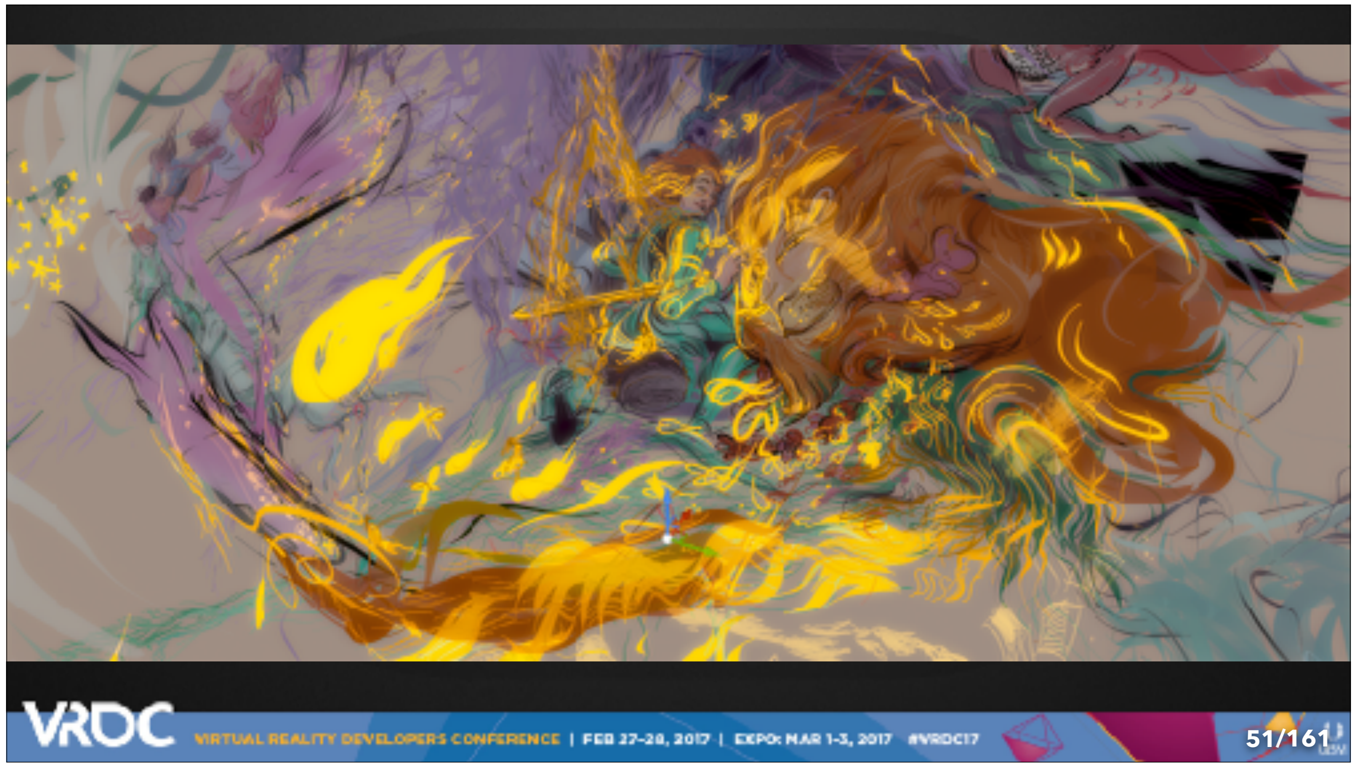
1. Tight iteration loop to make decisions in context
2. No modeling, texturing, layout departments necessary
3. Pipeline is easy to teach and use
4. Extremely fast throughput of large amounts of content

VRDC VIRTUAL REALITY DEVELOPERS CONFERENCE | FEB 27-28, 2017 | EXPO: MAR 1-3, 2017 #VRDC17

So here's the quick summary. I figure you can read. I'm going to talk about that last part though - as it allowed us to do something we called Detail Draw-Ins, or DDI for short.

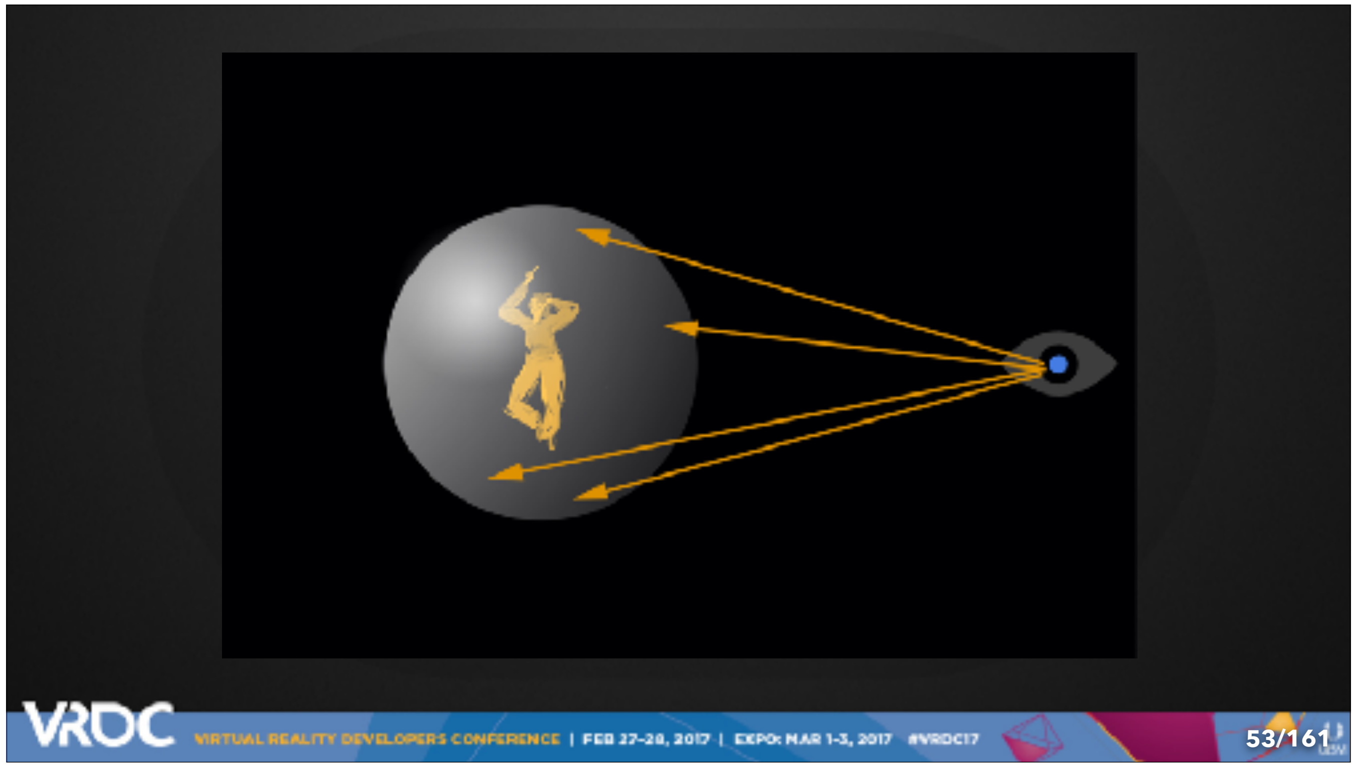

VRDC
VIRTUAL REALITY DEVELOPERS CONFERENCE | FEB 27-28, 2017 | EXPO: MAR 1-3, 2017 #VRDC17
50/161

So what's DDI? They're these hidden drawings placed in our scenes that are activated through a combination of proximity and gaze direction. They're ALL OVER the place. This is what one of our scenes looks like normally without any DDI active.

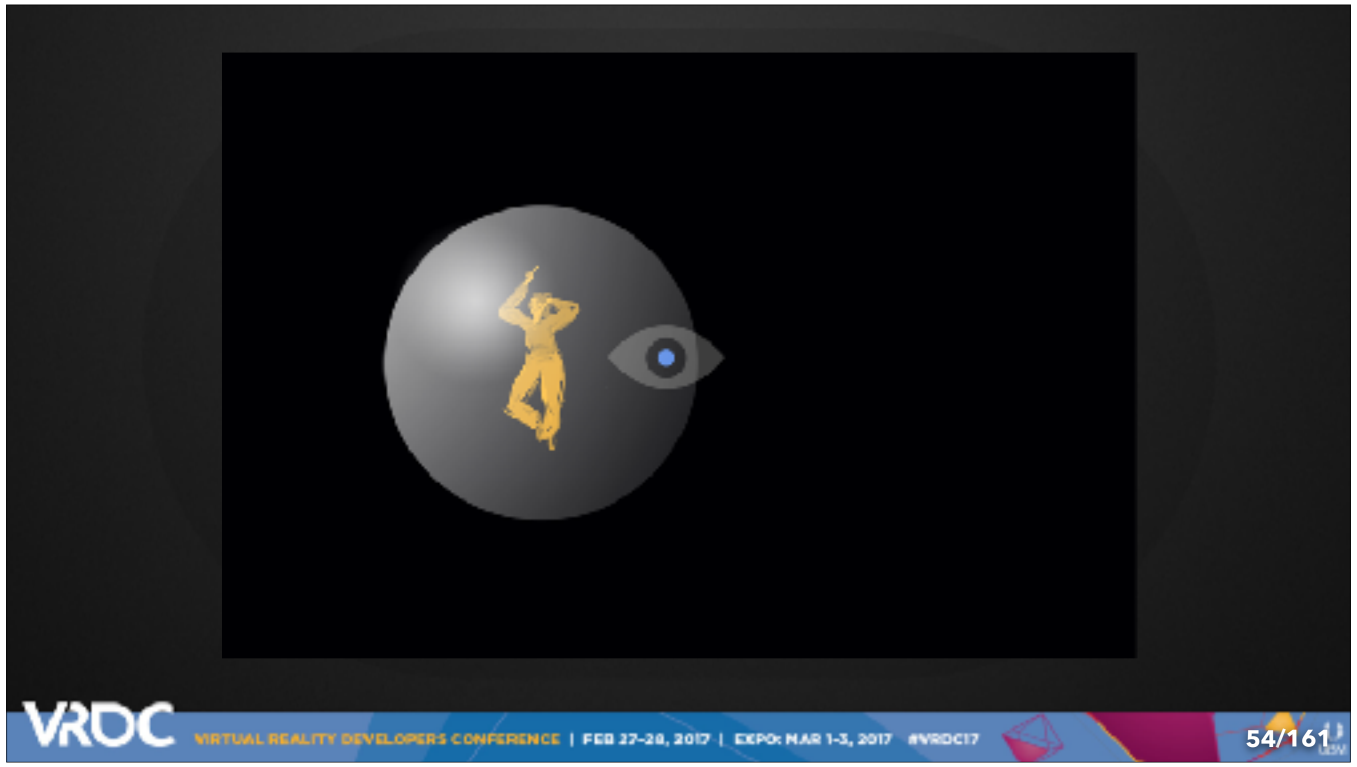

VRDC
VIRTUAL REALITY DEVELOPERS CONFERENCE | FEB 27-28, 2017 | EXPO: MAR 1-3, 2017 #VRDC17
51/161

And everything in gold here is DDI available to be activated. The reason we created these is because when we first started showing off Dear Angelica - many people reported that the loved being able to paint in the scenery as they looked around. This wasn't actually true - but whatever I wasn't going to correct them. About 80% of the time people were looking exactly where we wanted them to - due to a combination of visual and audio cues. However - we wanted to reward people during that other 20% - so we created a bunch of Detail Draw In's. As you can see we use it for different purposes - sometimes it's to add additional lines and detail to the image - sometimes its for narrative purposes - and sometimes

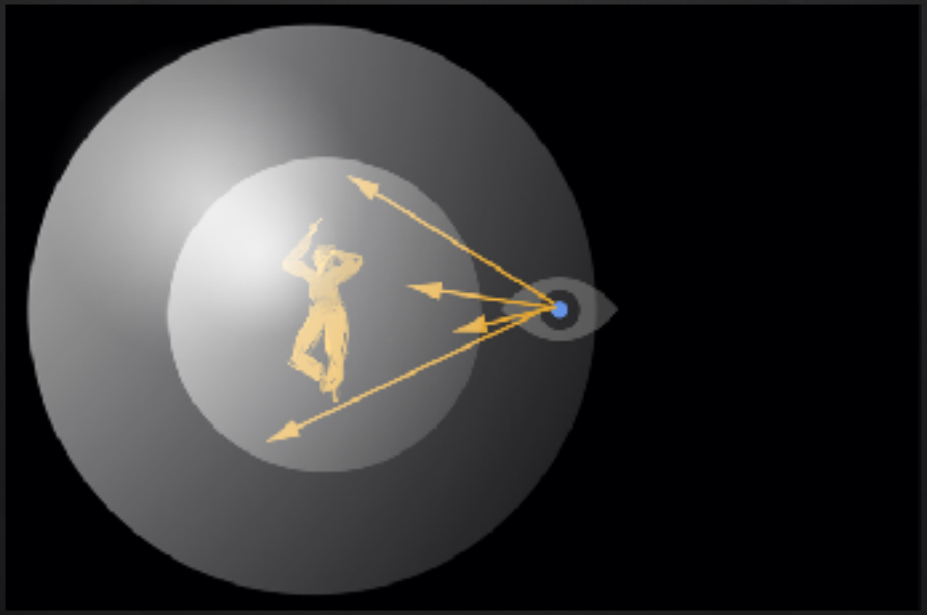
its for Easter Eggs. This is a self-portrait Wesley snuck into the film. The first time I saw it was a few weeks ago while working on this part of the presentation.


VRDC
VIRTUAL REALITY DEVELOPERS CONFERENCE | FEB 27-28, 2017 | EXPO: MAR 1-3, 2017 #VRDC17
53/161

So the way it works is - in Unreal we have invisible spheres hidden in our scenes that each contain a hidden drawing. Some of them are activated just by looking at them - like this.

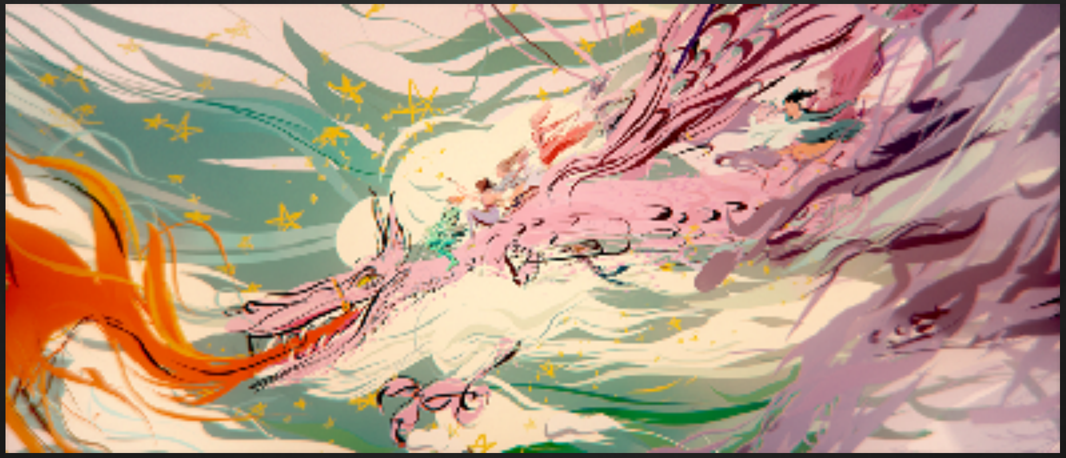

VRDC
VIRTUAL REALITY DEVELOPERS CONFERENCE | FEB 27-28, 2017 | EXPO: MAR 1-3, 2017 #VRDC17
54/161

Some of them are driven purely by proximity - as you see here.



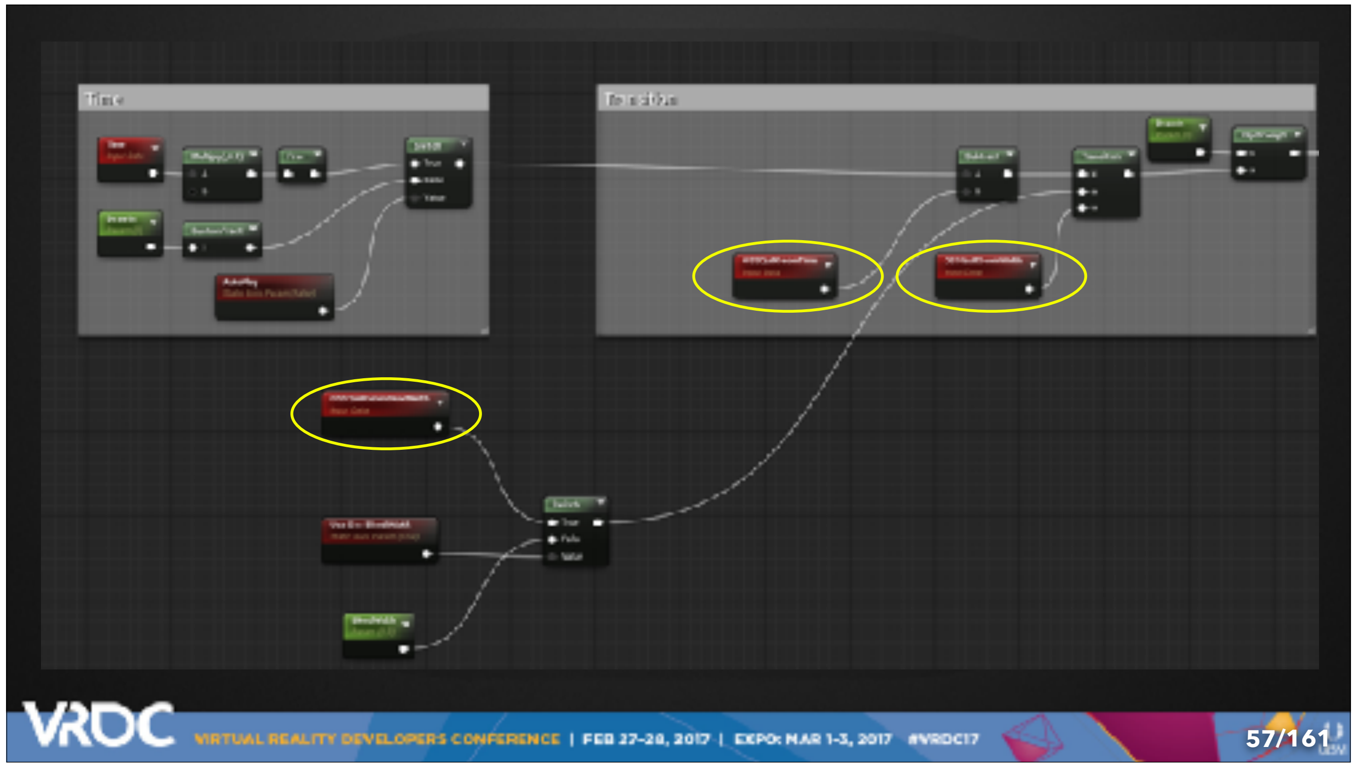
VRDC VIRTUAL REALITY DEVELOPERS CONFERENCE | FEB 27-28, 2017 | EXPO: MAR 1-3, 2017 #VRDC17

Lastly - some require you to be within a certain proximity AND looking at them to be activated. The decision to use which type was based on the size of the content and whether the content was aesthetic, narrative, or an easter egg. The type you see here tended to be used for easter eggs and narrative DDI.

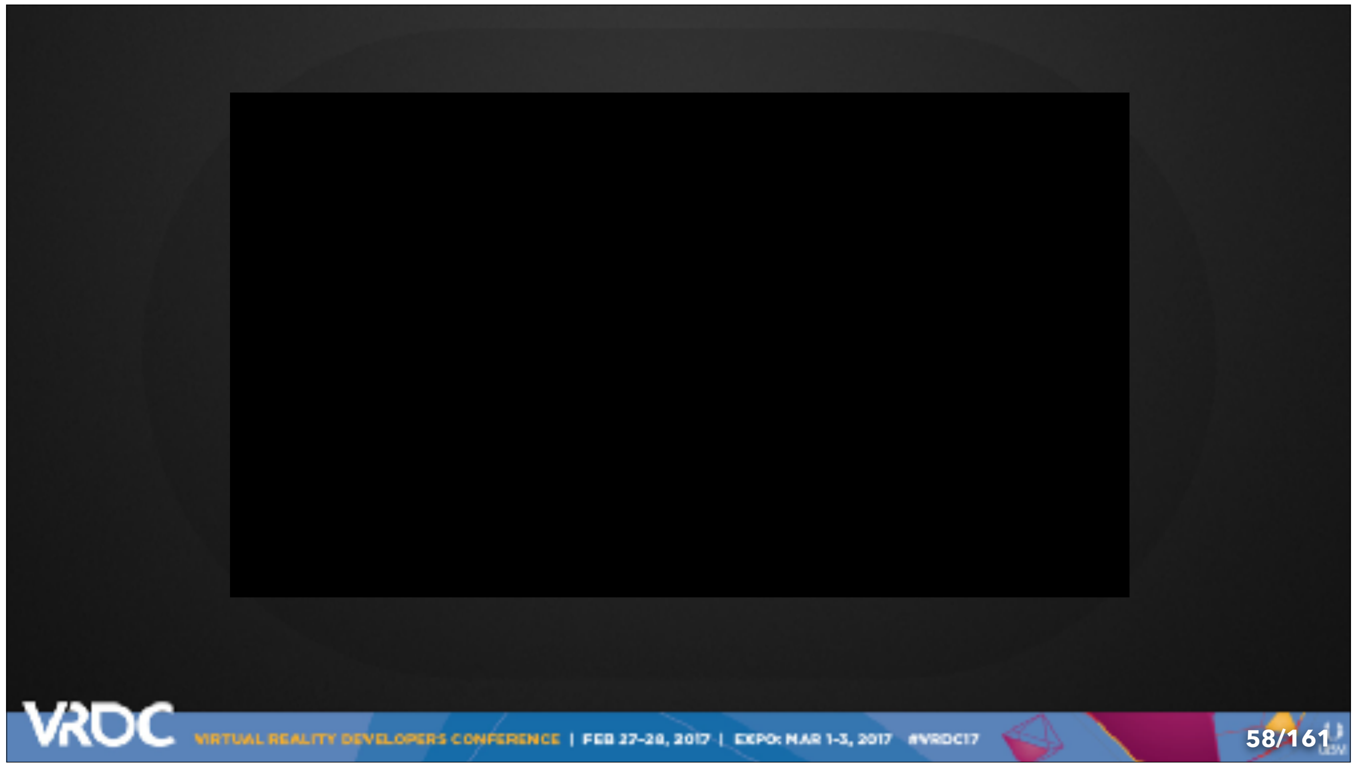


VRDC VIRTUAL REALITY DEVELOPERS CONFERENCE | FEB 27-28, 2017 | EXPO: MAR 1-3, 2017 #VRDC17

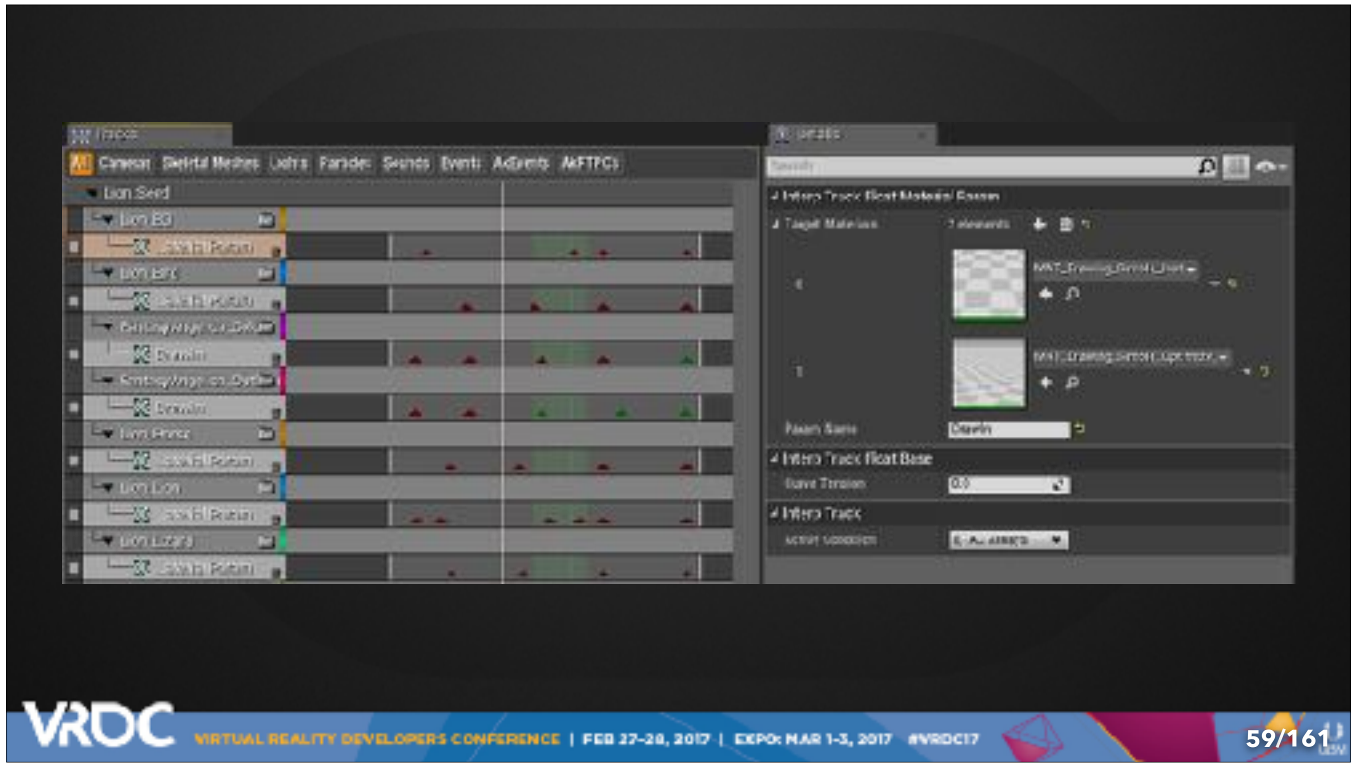
It's not really important that you notice it exists - what's important is that the project and the environment feels richer and more alive because of it. Personally I call this "reactivity" - having the environment subtly reacting to the user in a way that validates their presence. I LOVE LOVE LOVE when things are reactive.


VRDC
VIRTUAL REALITY DEVELOPERS CONFERENCE | FEB 27-28, 2017 | EXPO: MAR 1-3, 2017 #VRDC17
57/161

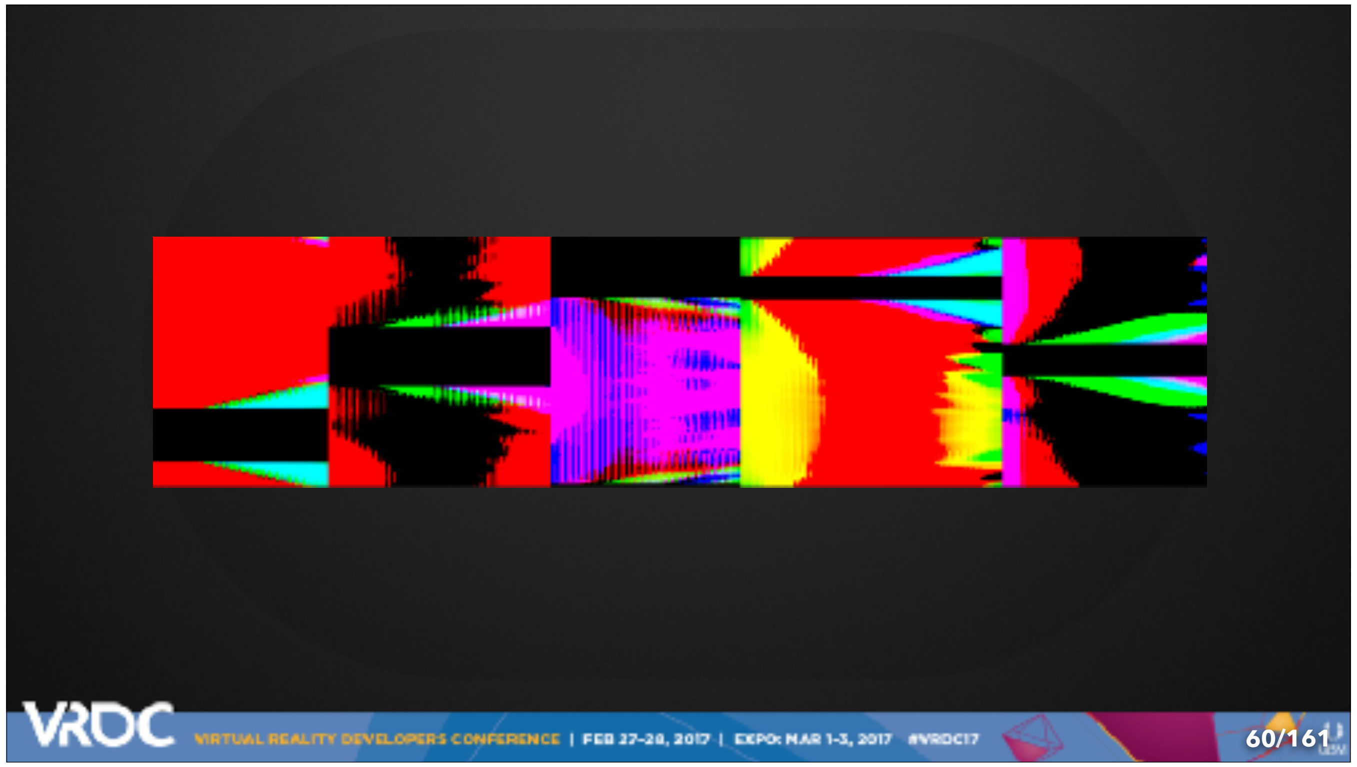
Now one of the key factors in being able to push all this content out was because of this shader - it controls how we draw the the curves in and out. If you look at the 3 circled inputs - these are all parameters coming directly from Quill that we can use to re-create exactly how Wesley drew everything. We use width and blend width so that - rather than fading in - the front edge of the lines actually grow in - as if you were laying down paint and it starts to bleed out on the page.


VIRTUAL REALITY DEVELOPERS CONFERENCE | FEB 27-28, 2017 | EXPO: MAR 1-3, 2017 #VRDC17

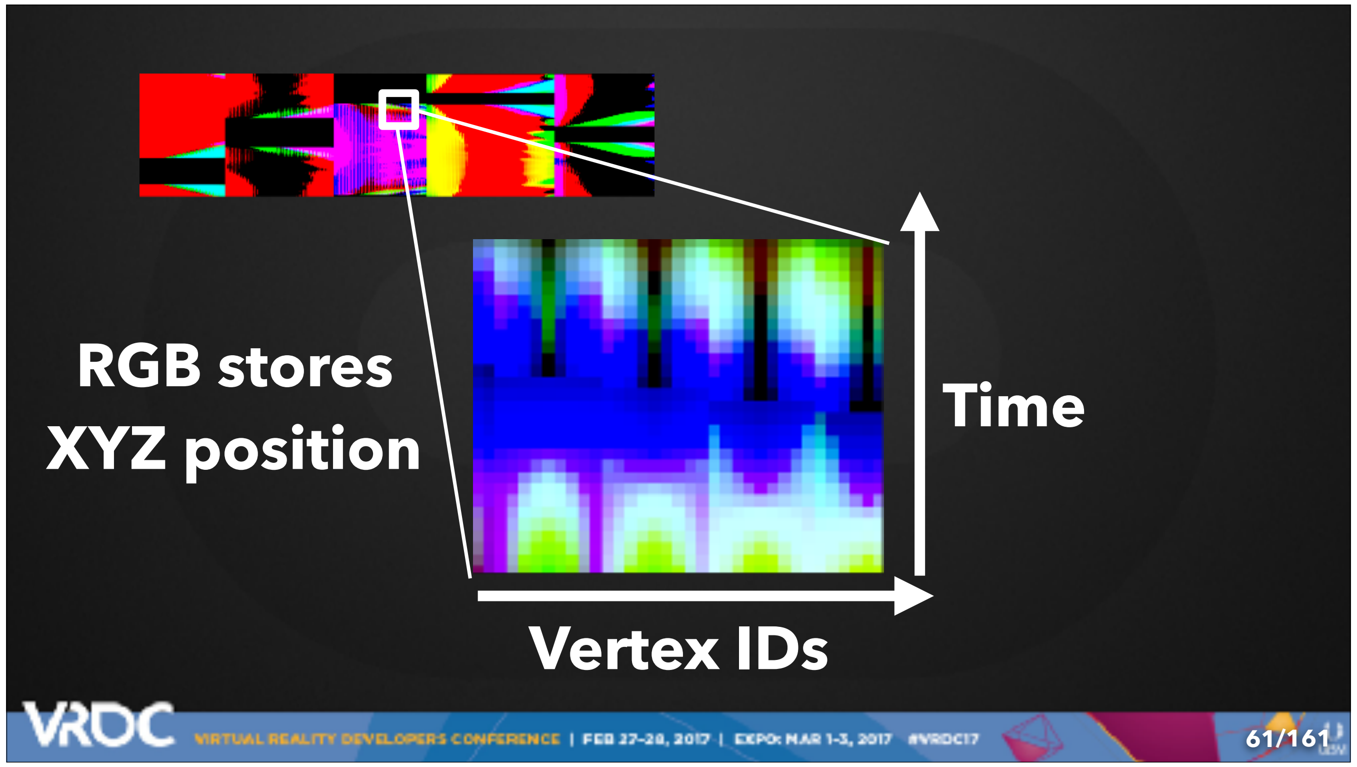
[video] We use the time parameter to draw the lines in the order and speed that Wesley did in Quill. I'm visualizing the parameter here - and as you can see the brightest areas draw in first and get progressively darker. For bigger assets we would modify these values in Houdini and bake them back into the curves - but for smaller assets and DDI we draw the pieces in exactly as Wesley drew them.



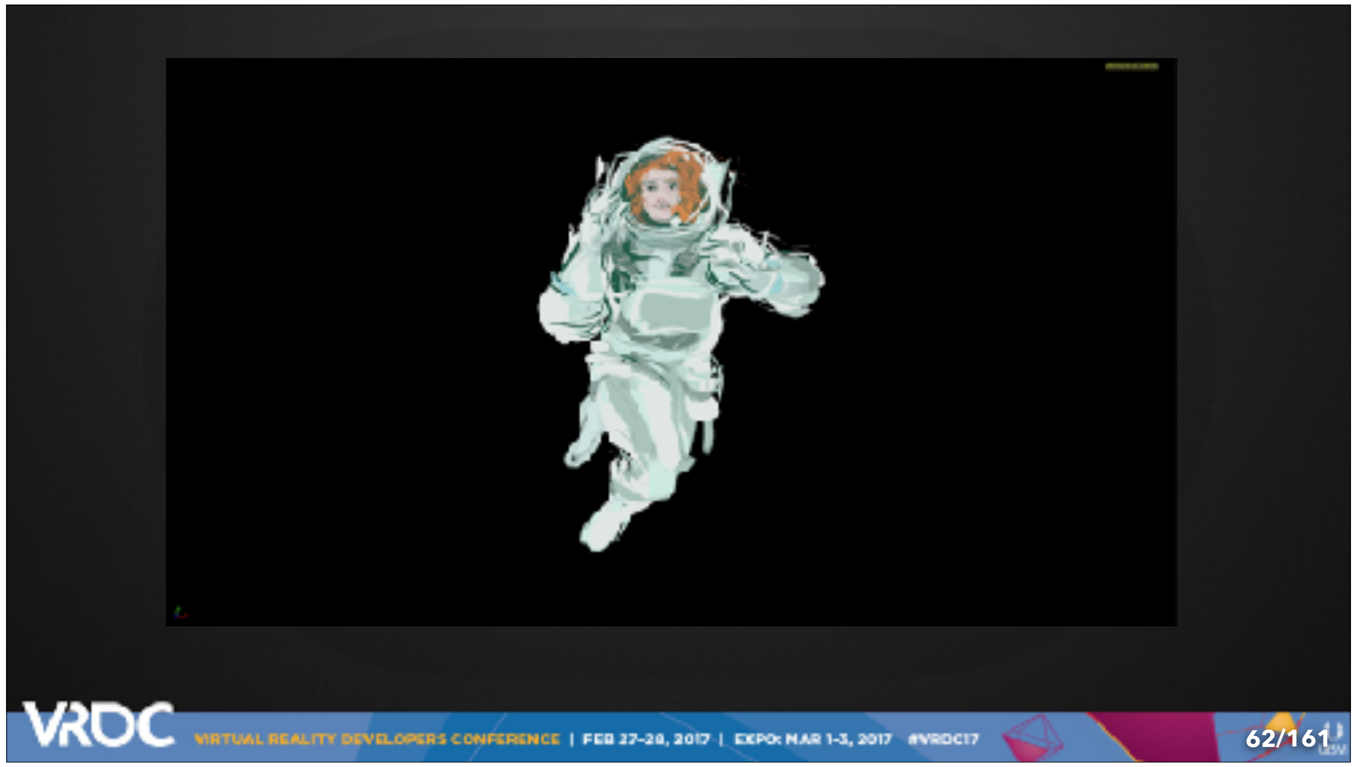
One cool thing is since we're just thresholding along value in a shader - we can control all of our draw-in and draw-out timing via a shader parameter in Matinee and get live feedback in the viewport. This made the editorial process as as easy as laying out a few keys in Matinee - which is actually pretty difficult - but whatever that's in the past now. We have sequencer. I'm not bitter.


VRDC
VIRTUAL REALITY DEVELOPERS CONFERENCE | FEB 27-28, 2017 | EXPO: MAR 1-3, 2017 #VRDC17
60/161

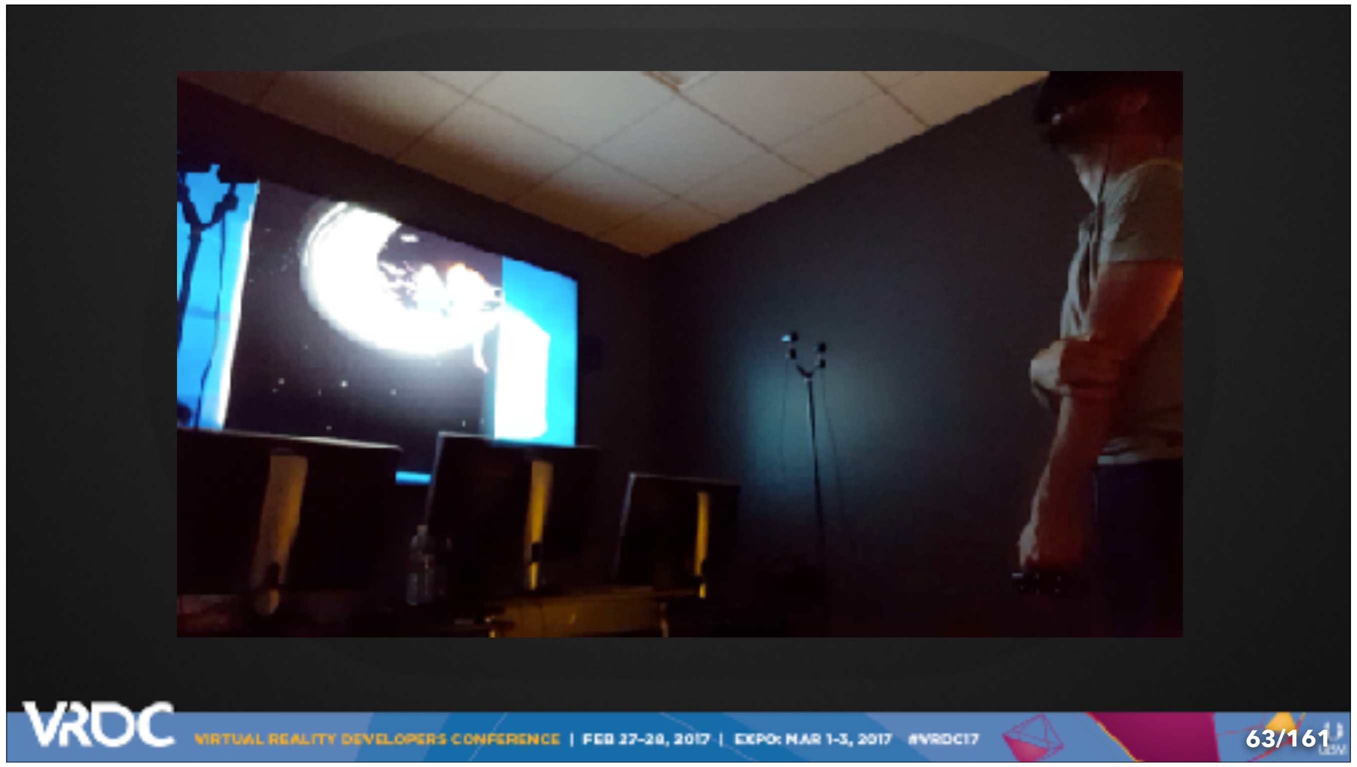
That's not the only shader trick we used either. Sometimes we needed our curves to move in very specific ways - and for this we used texture atlasses. To be clear - if you ever decide to do any of this I would suggest putting engineering resources toward a proper implementation for movement - but since we didn't have resources to spare - this is what we did.


RGB stores XYZ position
Time
Vertex IDs
VRDC VIRTUAL REALITY DEVELOPERS CONFERENCE | FEB 27-28, 2017 | EXPO: MAR 1-3, 2017 #VRDC17
61/161

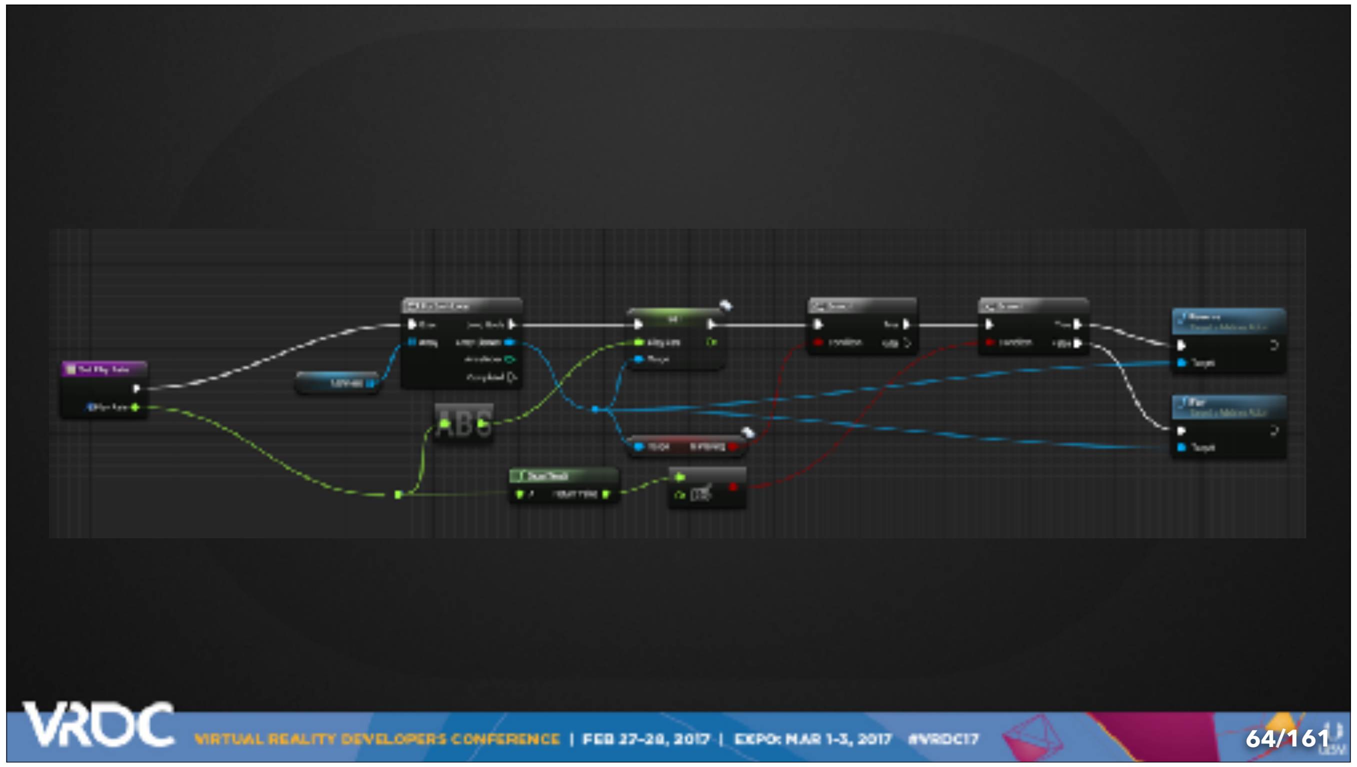
Basically every column is a vertex on the curve. Each row is a frame in time. We convert the RGB values into XYZ offsets in object space and apply that transformation. The upside was that we didn't have to do any engineering work to get this up and running. The downside is these files are HUGE and we can't do compression. We're posting a blog soon that goes into WAY more detail as there's some really interesting tricks around interpolation and implementation.


VRDC
VIRTUAL REALITY DEVELOPERS CONFERENCE | FEB 27-28, 2017 | EXPO: MAR 1-3, 2017 #VRDC17
62/161

This method is also how we got our creepy outer space dance video. Space Angelica was rigged and animated in Maya. Key poses were baked out texture maps as vertex offsets that we would linearly interpolate between. It's worth noting that this stuff is uncanny valley to the max in VR - so we toned the movement WAY down.


VRDC
VIRTUAL REALITY DEVELOPERS CONFERENCE | FEB 27-28, 2017 | EXPO: MAR 1-3, 2017 #VRDC17

[video] One last thing we found EXTREMELY useful during production is shown in this review session. Previously - we would have to re-launch the application every time we wanted to watch a sequence.
Then we added what we call “power of time” We took a rift controller and hooked up the thumbstick to fast-forward and rewind the experience. We also hooked up little laser pointers to the controllers so Saschka could point at what he was talking about.
It allowed us to efficiently review our FX, editorial, as well as compose in 360. Saschka would often watch through a sequence in a cardinal direction giving notes, then rotate 90 degrees and go through the whole sequence again - making sure that there was always something interesting to look at.


VRDC
VIRTUAL REALITY DEVELOPERS CONFERENCE | FEB 27-28, 2017 | EXPO: MAR 1-3, 2017 #VRDC17
64/161

The way we did this was quite simple - grabbing all matinees, seeing if they're playing, and then firing the play or reverse functions. Super easy and I highly suggest you use it.

TO DO
FINISH DA
VRDC
VIRTUAL REALITY DEVELOPERS CONFERENCE | FEB 27-28, 2017 | EXPO: MAR 1-3, 2017 #VRDC17
65/161

So that's the end of my song and dance - a kind of birds eye view of some of the interesting bits in Dear Angelica.

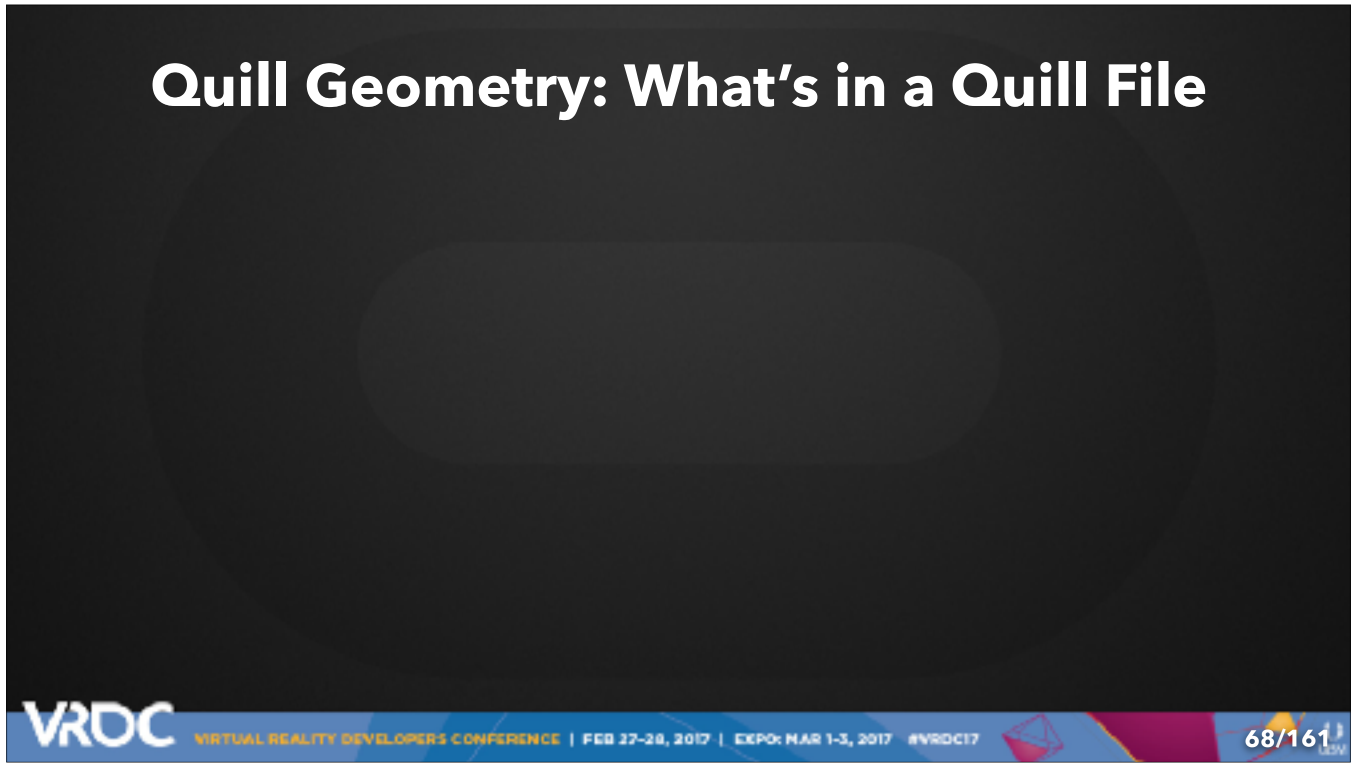
Now we'll have Ian, our Quill integration lead, hop in and tell you some of the nitty gritty details of our quill graphics pipeline and unreal integration.

Thank you Chris.

Good morning, I’m Ian Wakelin, I worked on the Quill to Unreal integration for Dear Angelica.

For this portion of the presentation, we’re going to start diving more into the technical details of the integration of Quill into Unreal Editor. I’ll provide some more detail on the Quill file format, and briefly touch on how we added a new custom geometry type into the Unreal Editor workflow. I’ll also describe the performance graphs we created to help make sure all our scenes were performing at or above 90fps.

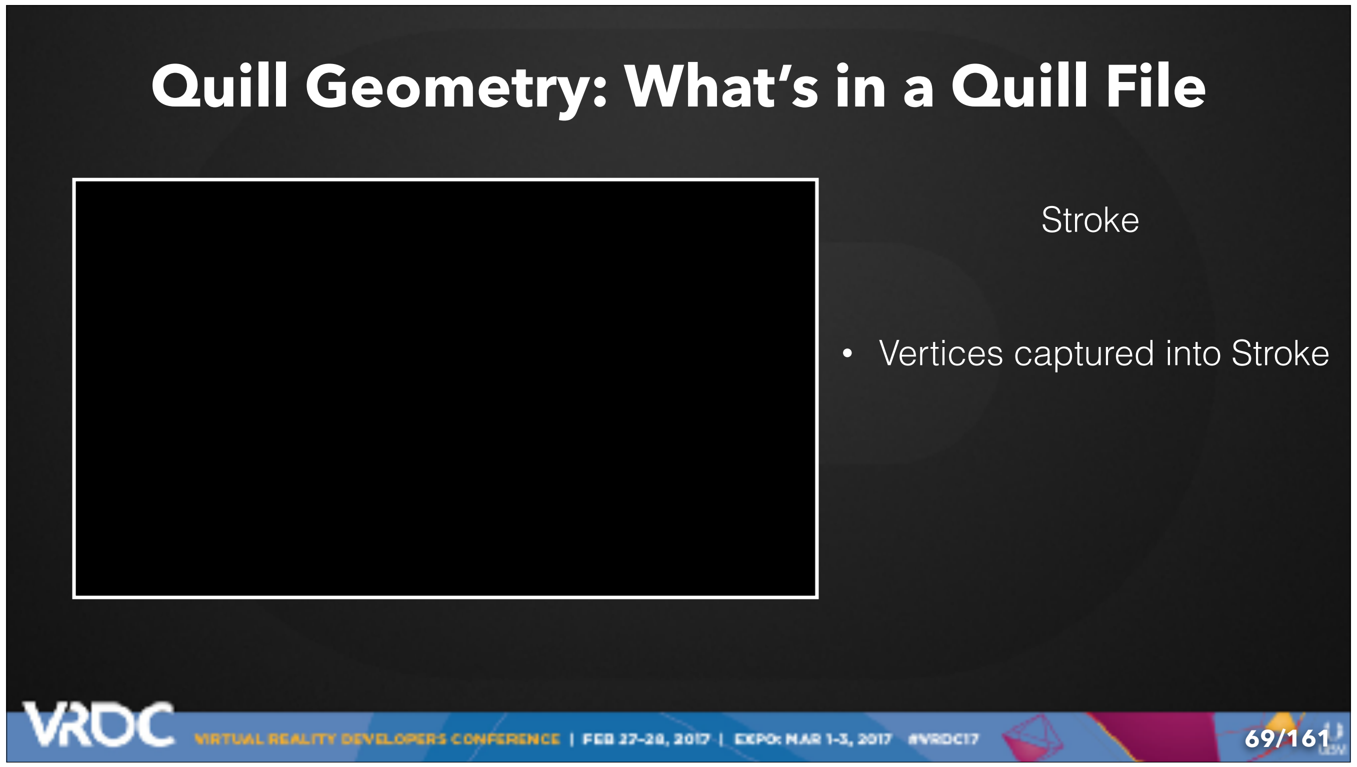
After this portion, Martin will present several techniques that he applied in DA to keep us fast and graphically beautiful.



Quill Geometry: What's in a Quill File

VIRTUAL REALITY DEVELOPERS CONFERENCE | FEB 27-28, 2017 | EXPO: MAR 1-3, 2017 #VRDC17

So first off, let's look at what makes up a Quill file.



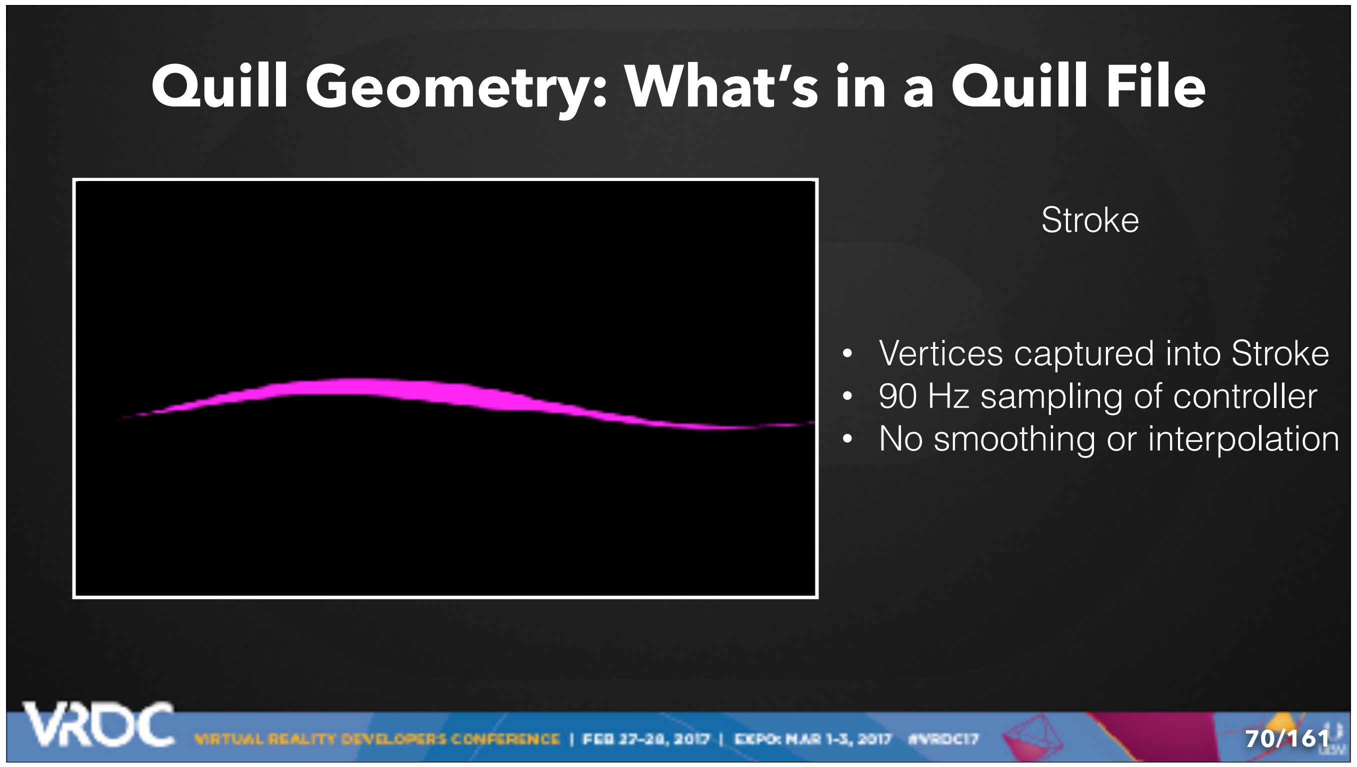
Quill Geometry: What's in a Quill File

Stroke

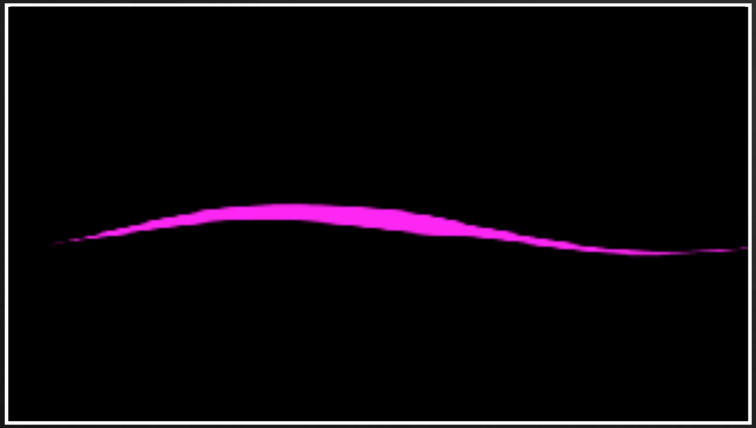
- Vertices captured into Stroke

VRDC VIRTUAL REALITY DEVELOPERS CONFERENCE | FEB 27-28, 2017 | EXPO: MAR 1-3, 2017 #VRDC17

In the Quill app, as someone uses the paint tool, we continuously capture controller transforms at 90hz and mix in tool properties like color, width and transparency to make a stroke.



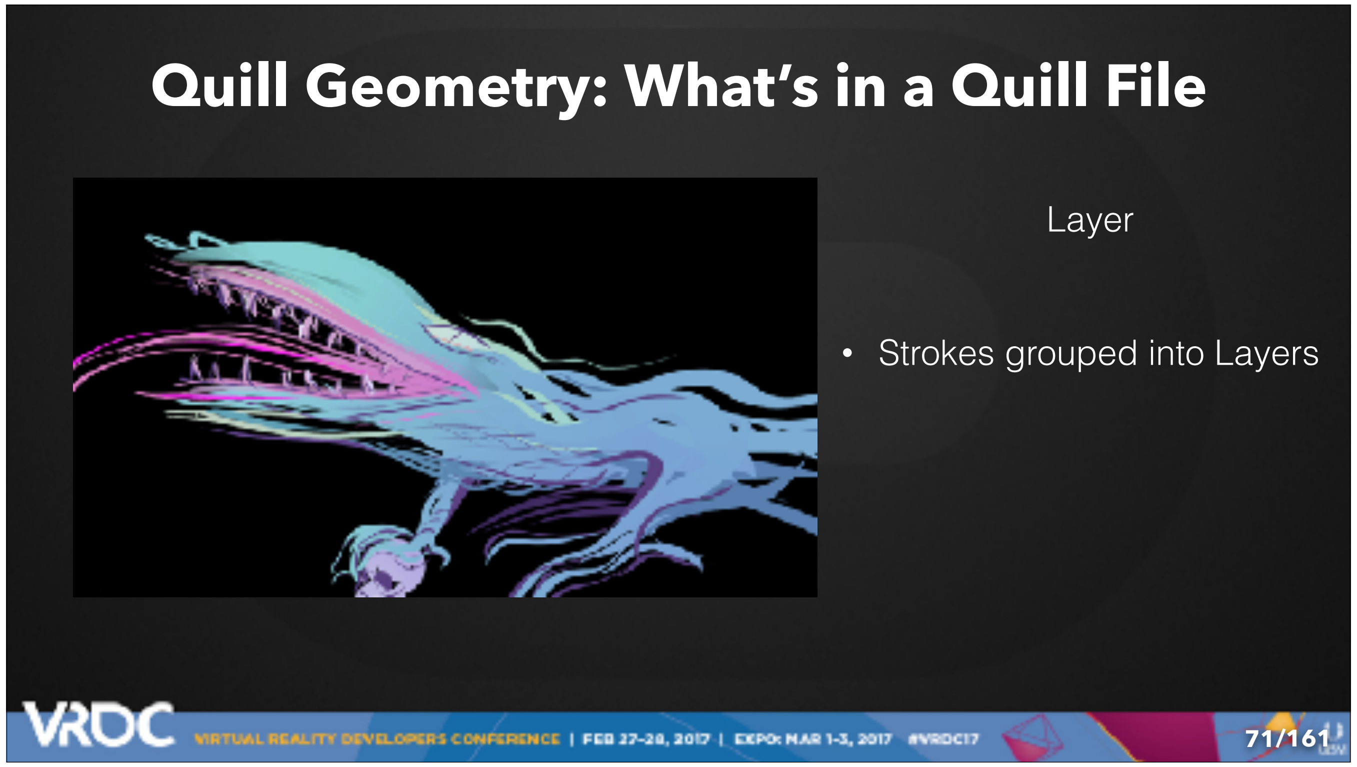
Quill Geometry: What's in a Quill File



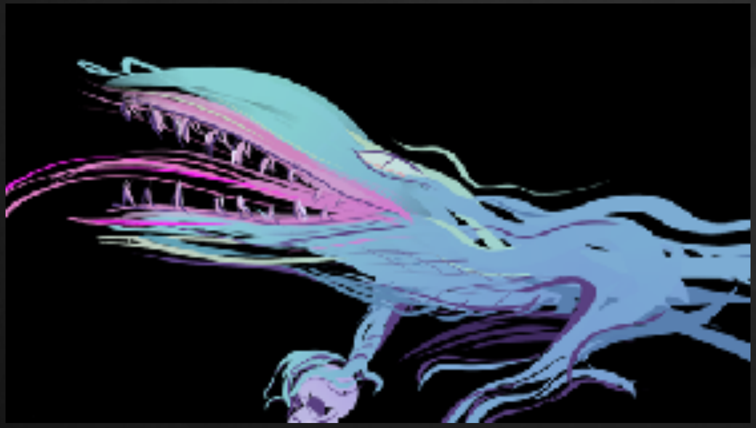
Stroke

- Vertices captured into Stroke
- 90 Hz sampling of controller
- No smoothing or interpolation

Strokes use raw transforms, no filtering is applied.



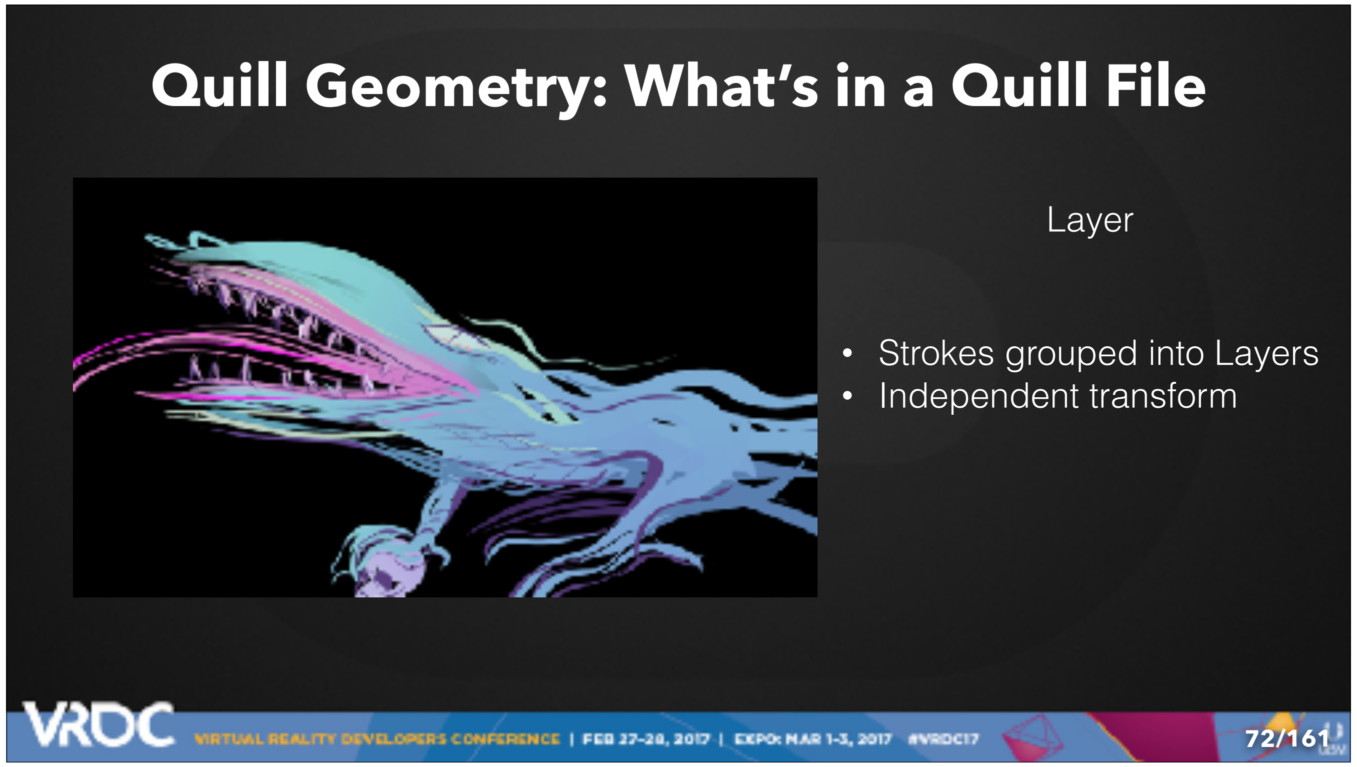
Quill Geometry: What's in a Quill File



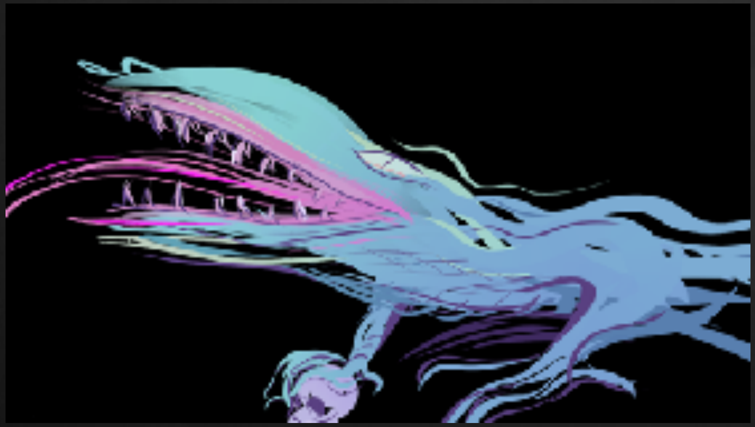
Layer

- Strokes grouped into Layers

Strokes are grouped into layers. This is more or less equivalent to an Unreal Actor.



Quill Geometry: What's in a Quill File

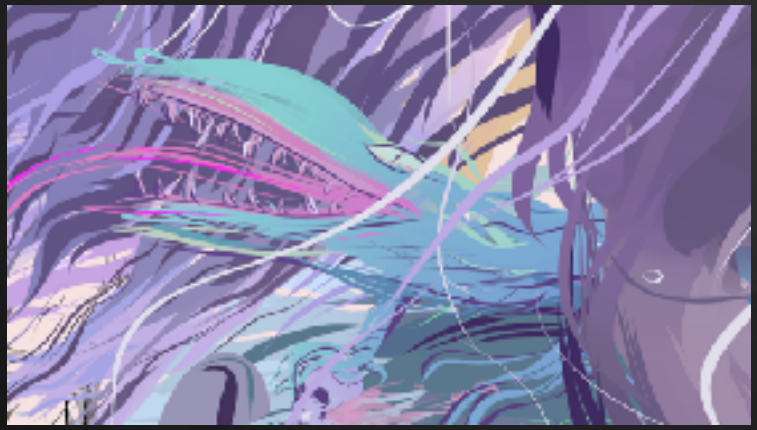


Layer

- Strokes grouped into Layers
- Independent transform

A layer can be independently transformed in the scene, but otherwise is strictly organizational.

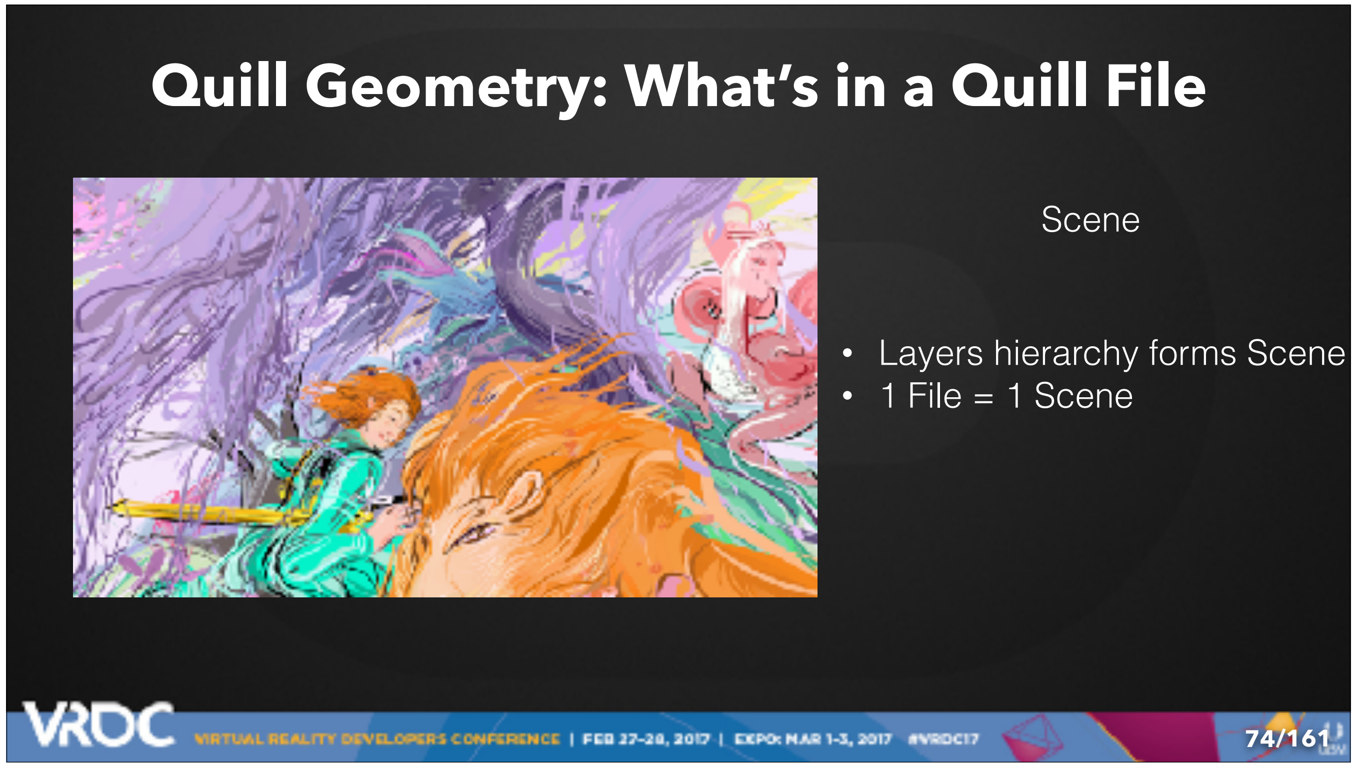
Quill Geometry: What’s in a Quill File



Scene

- Layers hierarchy forms Scene

And a scene is just a hierarchy of layers.



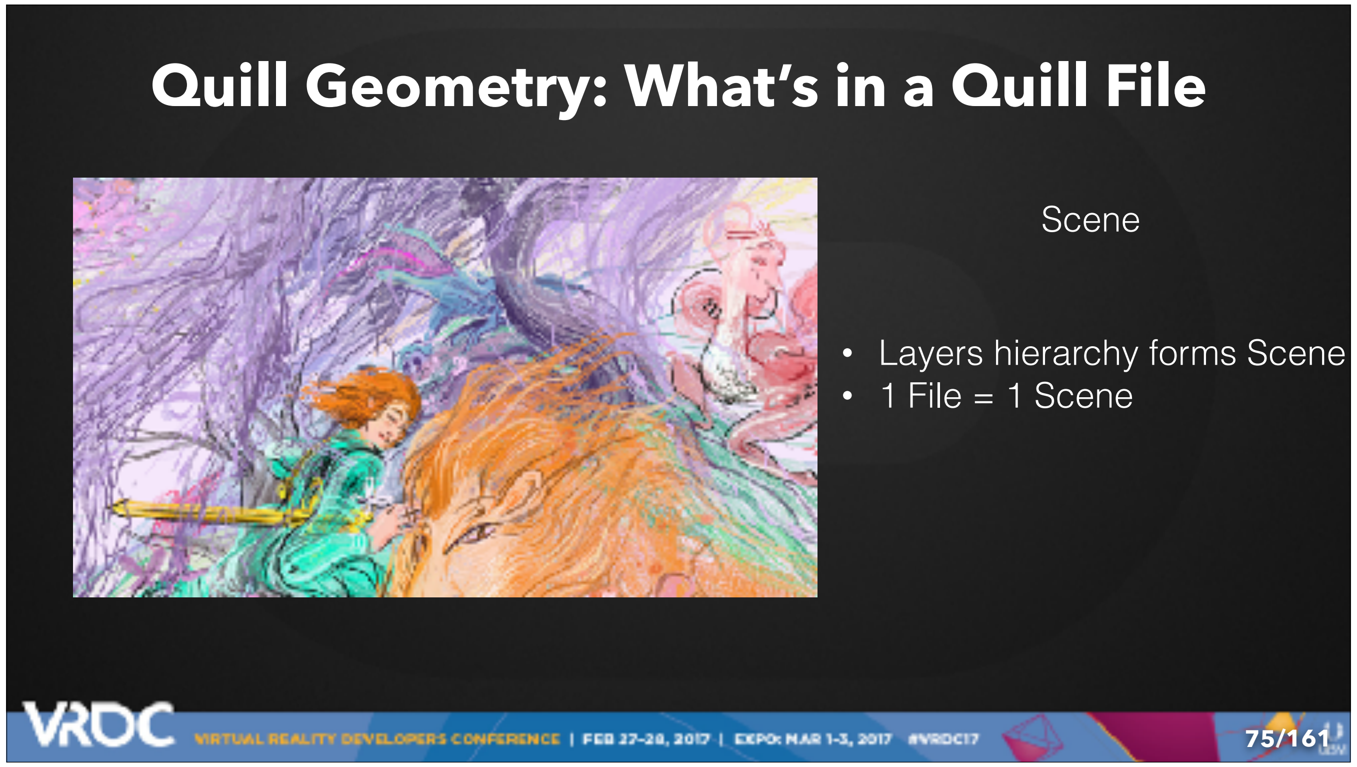
Quill Geometry: What's in a Quill File

Scene

- Layers hierarchy forms Scene
- 1 File = 1 Scene

VRDC VIRTUAL REALITY DEVELOPERS CONFERENCE | FEB 27-28, 2017 | EXPO: MAR 1-3, 2017 #VRDC17

Each Quill file contains a single scene.



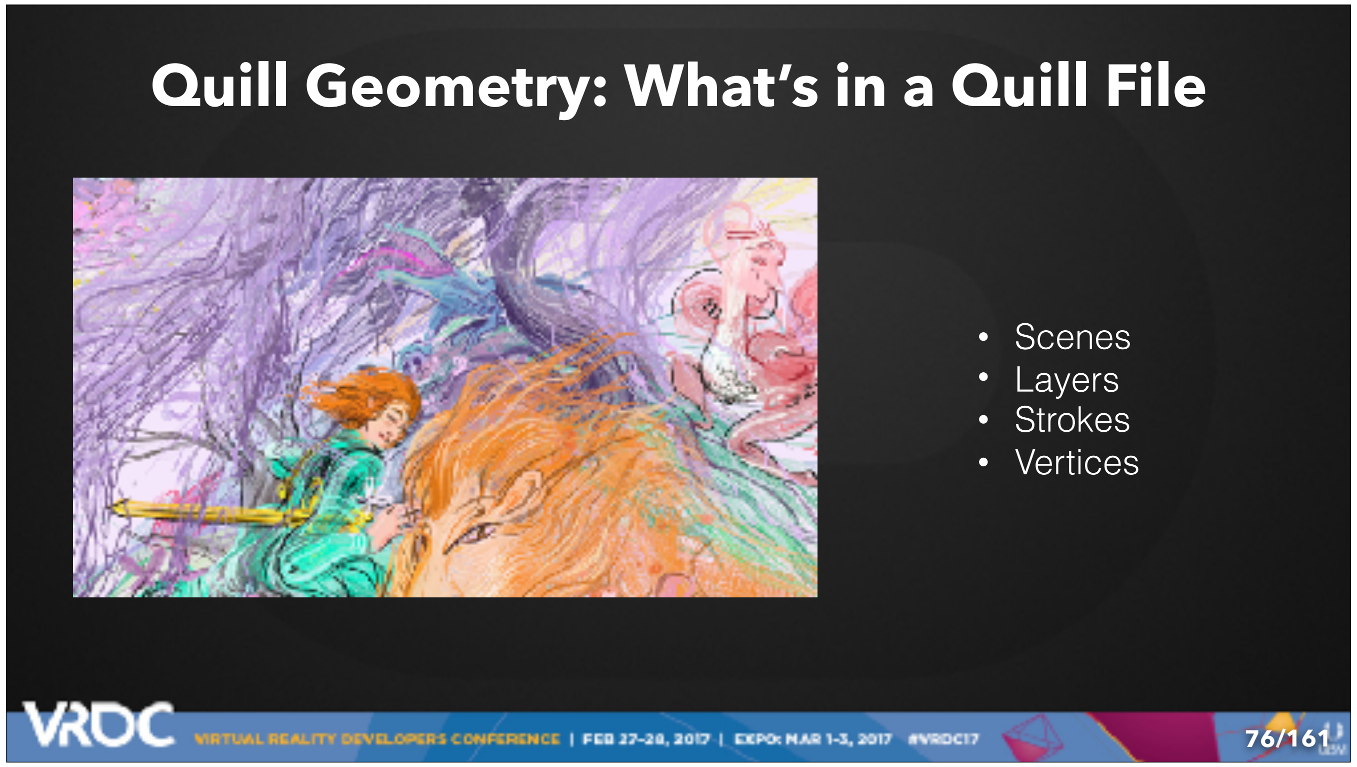
Quill Geometry: What's in a Quill File

Scene

- Layers hierarchy forms Scene
- 1 File = 1 Scene

VRDC VIRTUAL REALITY DEVELOPERS CONFERENCE | FEB 27-28, 2017 | EXPO: MAR 1-3, 2017 #VRDC17

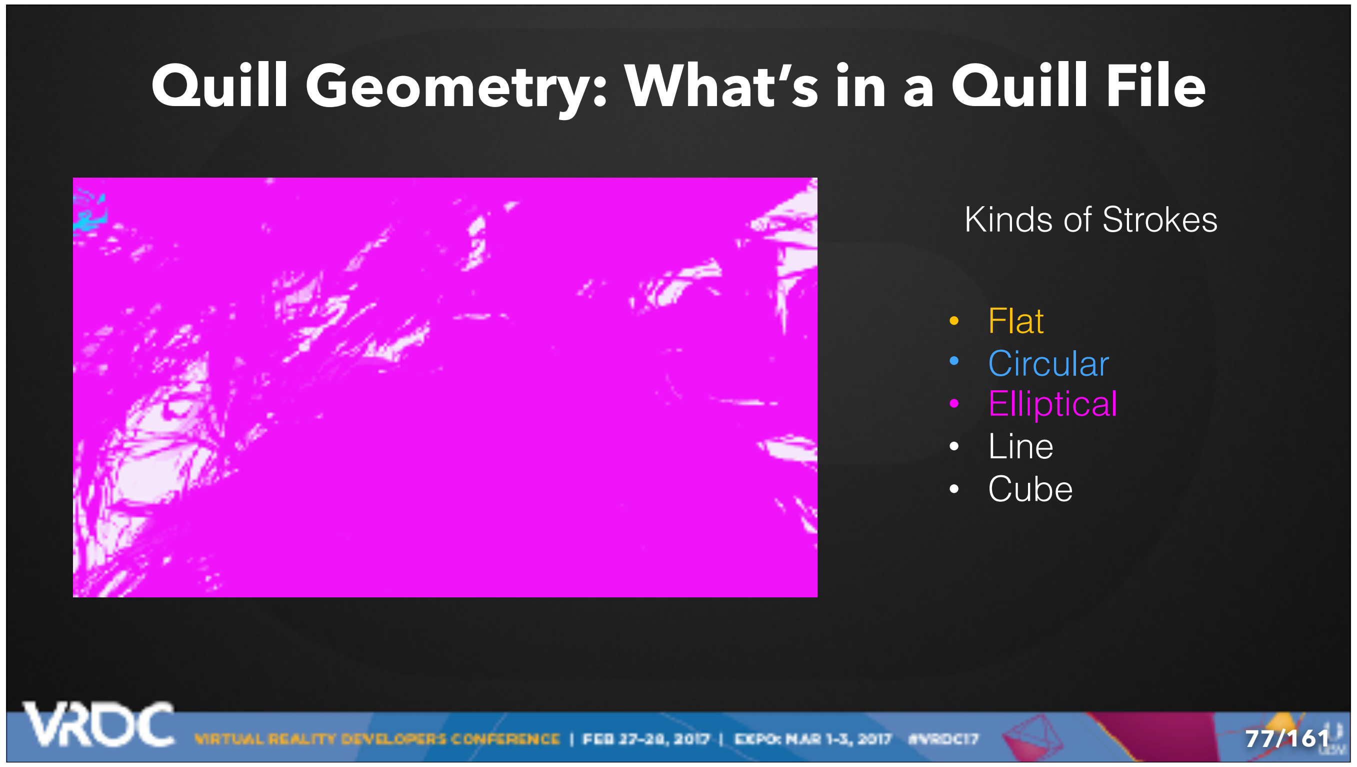
This is a wireframe of the same scene, just to give a sense of the density.



Quill Geometry: What's in a Quill File

- Scenes
- Layers
- Strokes
- Vertices

That's actually it for the core types of things used for drawing: Scenes, Layers, Strokes, and Vertices. Let's look at strokes in more detail.



Quill Geometry: What's in a Quill File

Kinds of Strokes

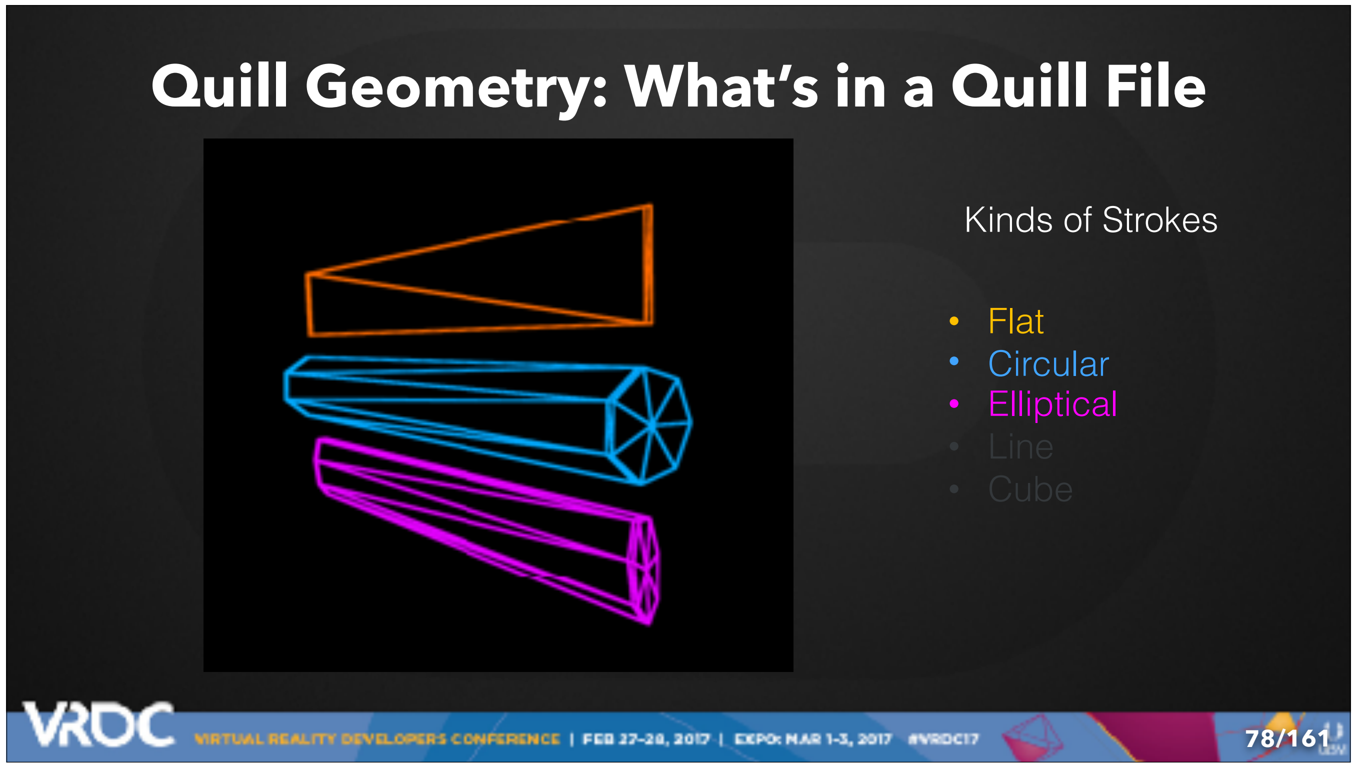
- Flat
- Circular
- Elliptical
- Line
- Cube

VRDC VIRTUAL REALITY DEVELOPERS CONFERENCE | FEB 27-28, 2017 | EXPO: MAR 1-3, 2017 #VRDC17

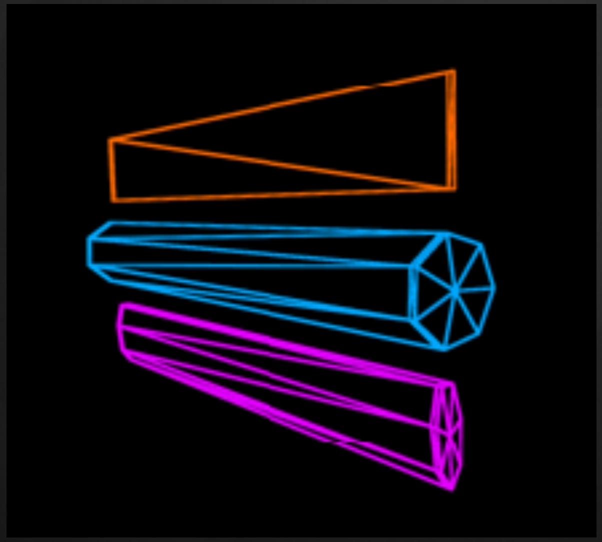
77/161

This is the same scene again, with stroke brush type visualization. We actually used 3 different stroke brush topologies in Dear Angelica, flat, circular, and elliptical.

There's a tiny bit of the circular brush in the upper left, but it's mostly the elliptical brush in this scene.



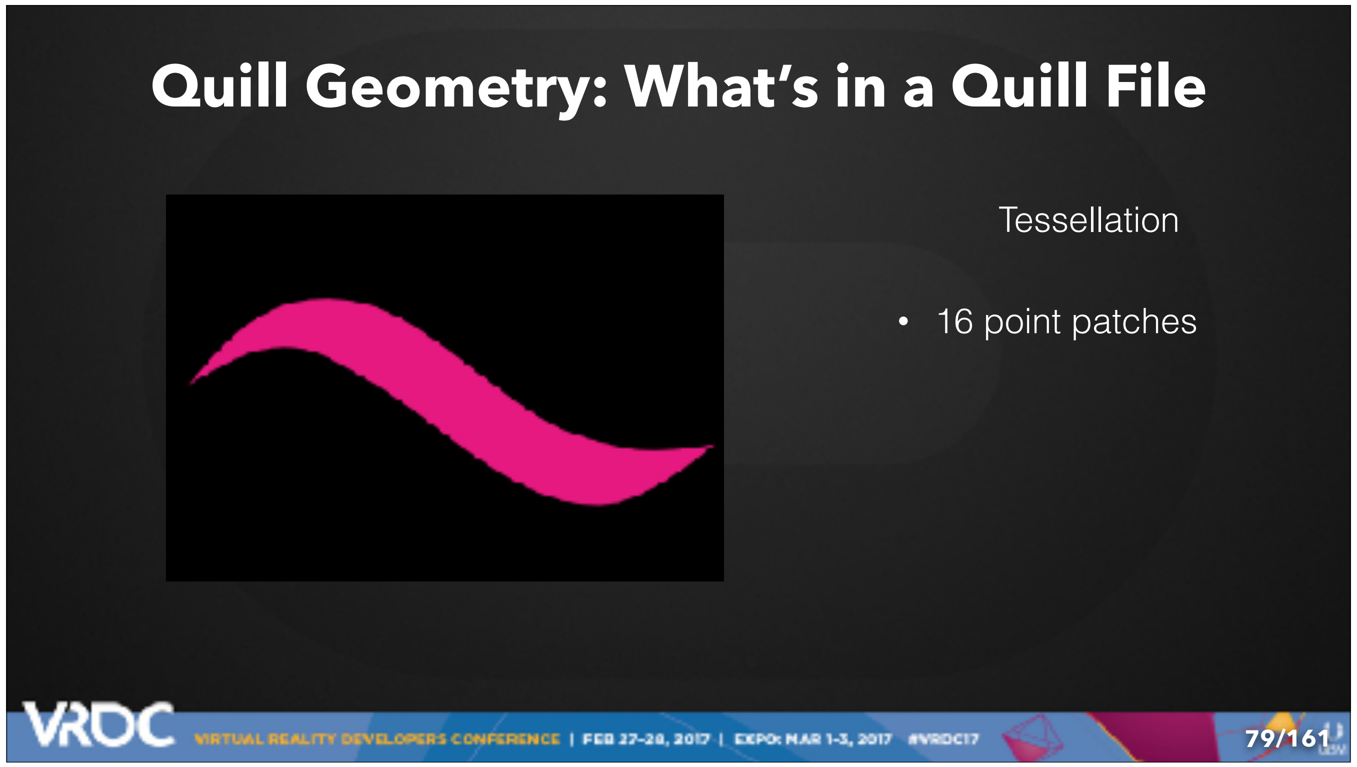
Quill Geometry: What's in a Quill File



Kinds of Strokes

- Flat
- Circular
- Elliptical
- Line
- Cube

Here's a closer look at those brush types. From top to bottom we have the flat, the circular, and the elliptical. Inigo set up the tessellation shader configuration in Quill to use fixed amplification to generate these topologies.



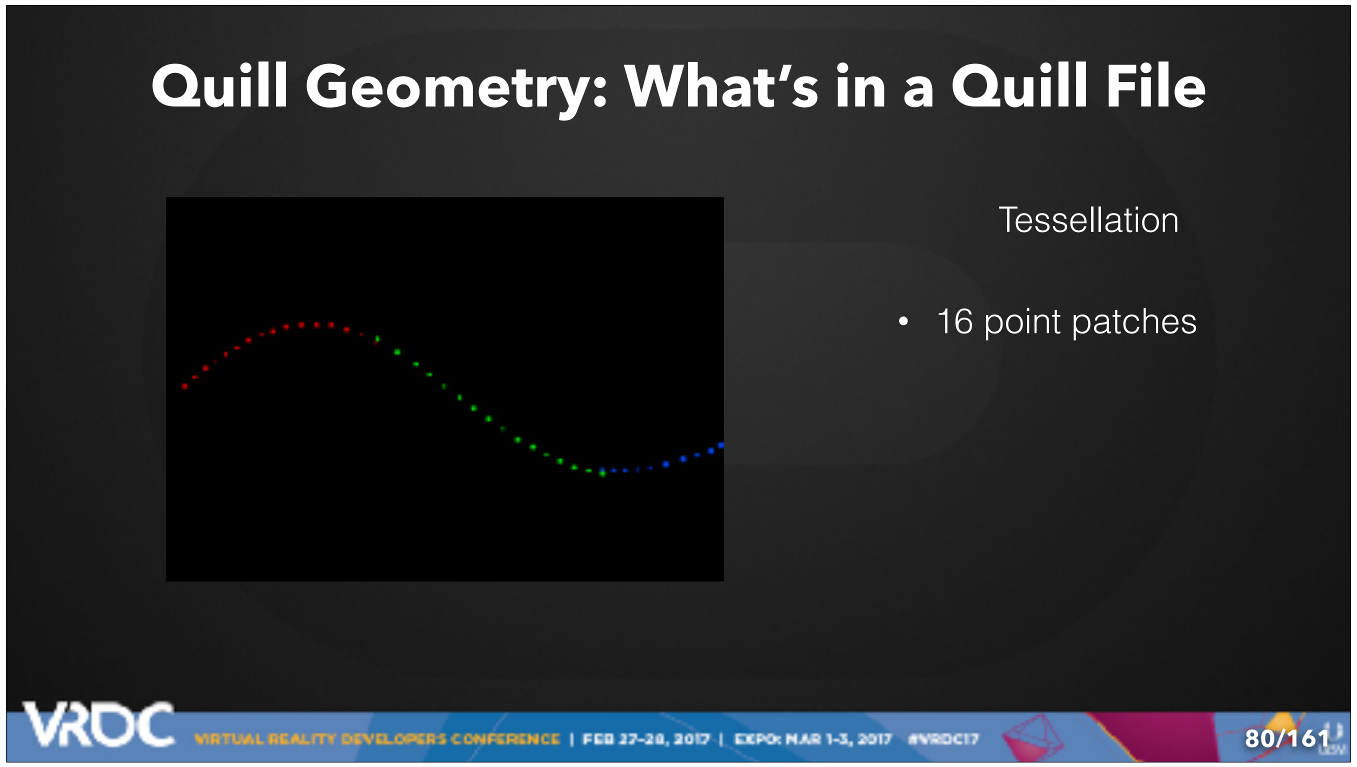
Quill Geometry: What's in a Quill File

Tessellation

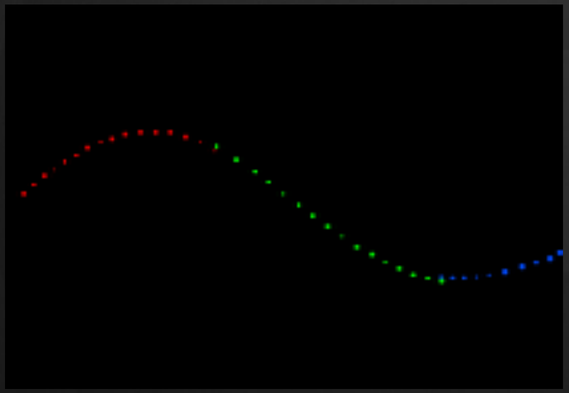
- 16 point patches

VIRTUAL REALITY DEVELOPERS CONFERENCE | FEB 27-28, 2017 | EXPO: MAR 1-3, 2017 #VRDC17

We use 16 point patch primitives. For the elliptical brush, for example, to produce a stroke like this one



Quill Geometry: What’s in a Quill File



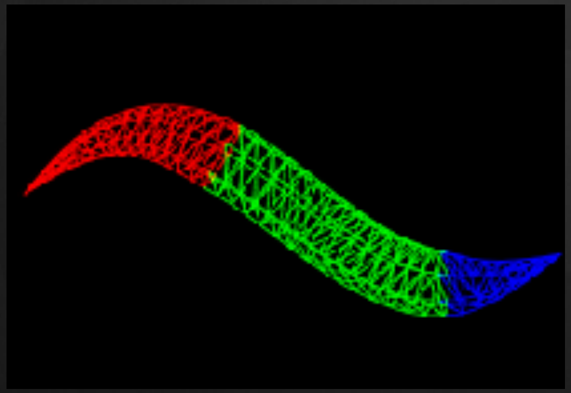
Tessellation

- 16 point patches

VRDC VIRTUAL REALITY DEVELOPERS CONFERENCE | FEB 27-28, 2017 | EXPO: MAR 1-3, 2017 #VRDC17

we take the curve points and group them into 16 point patches, with overlap. Here I’m using RGB to indicate distinct patches.

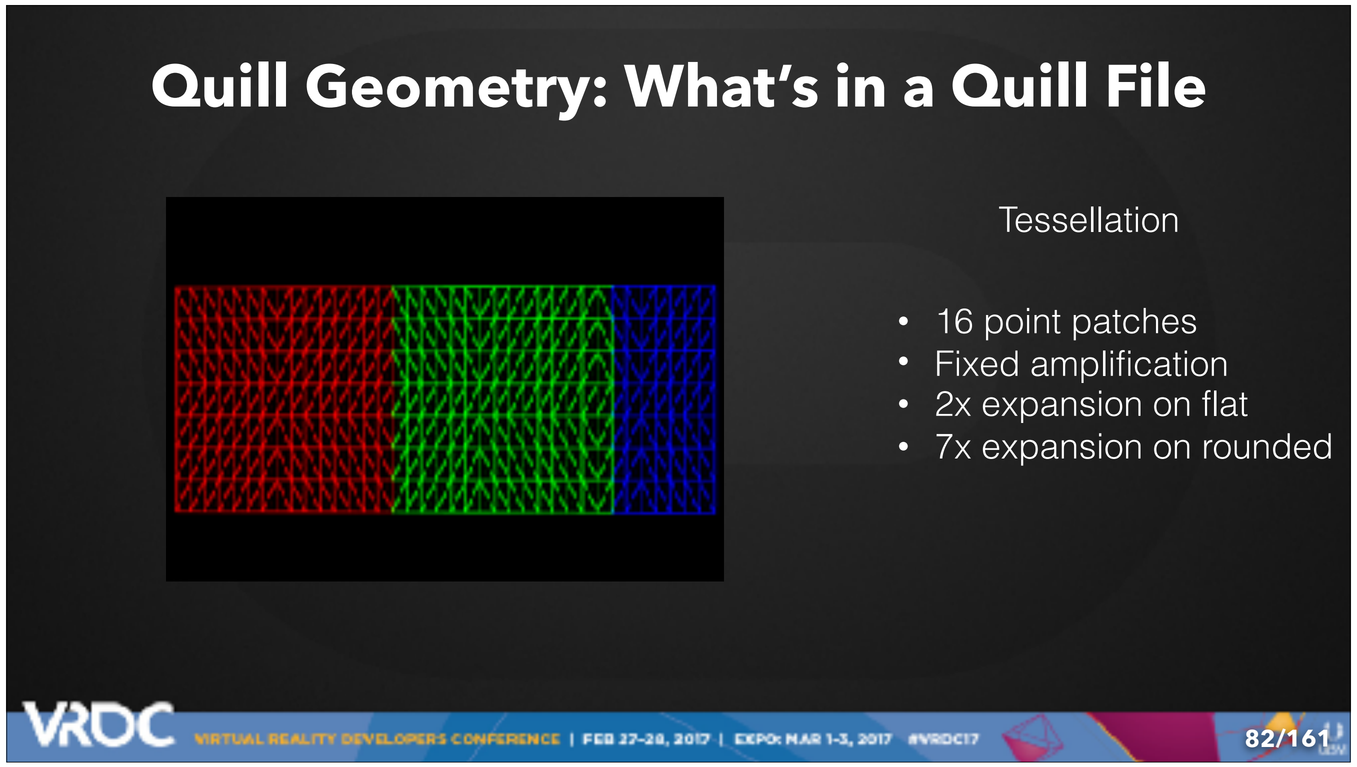
Quill Geometry: What's in a Quill File



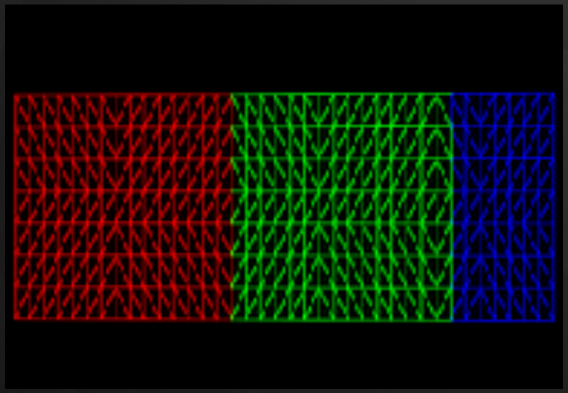
Tessellation

- 16 point patches
- Fixed amplification
- 2x expansion on flat
- 7x expansion on rounded

So this particular geometry will be drawn as 3 16-point patch primitives. The final blue patch on the right doesn't have 16 points, but is tessellated at a lower amplification to get the correct topology.



Quill Geometry: What's in a Quill File



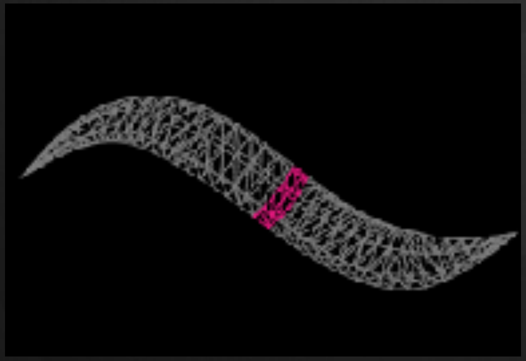
Tessellation

- 16 point patches
- Fixed amplification
- 2x expansion on flat
- 7x expansion on rounded

VRDC VIRTUAL REALITY DEVELOPERS CONFERENCE | FEB 27-28, 2017 | EXPO: MAR 1-3, 2017 #VRDC17

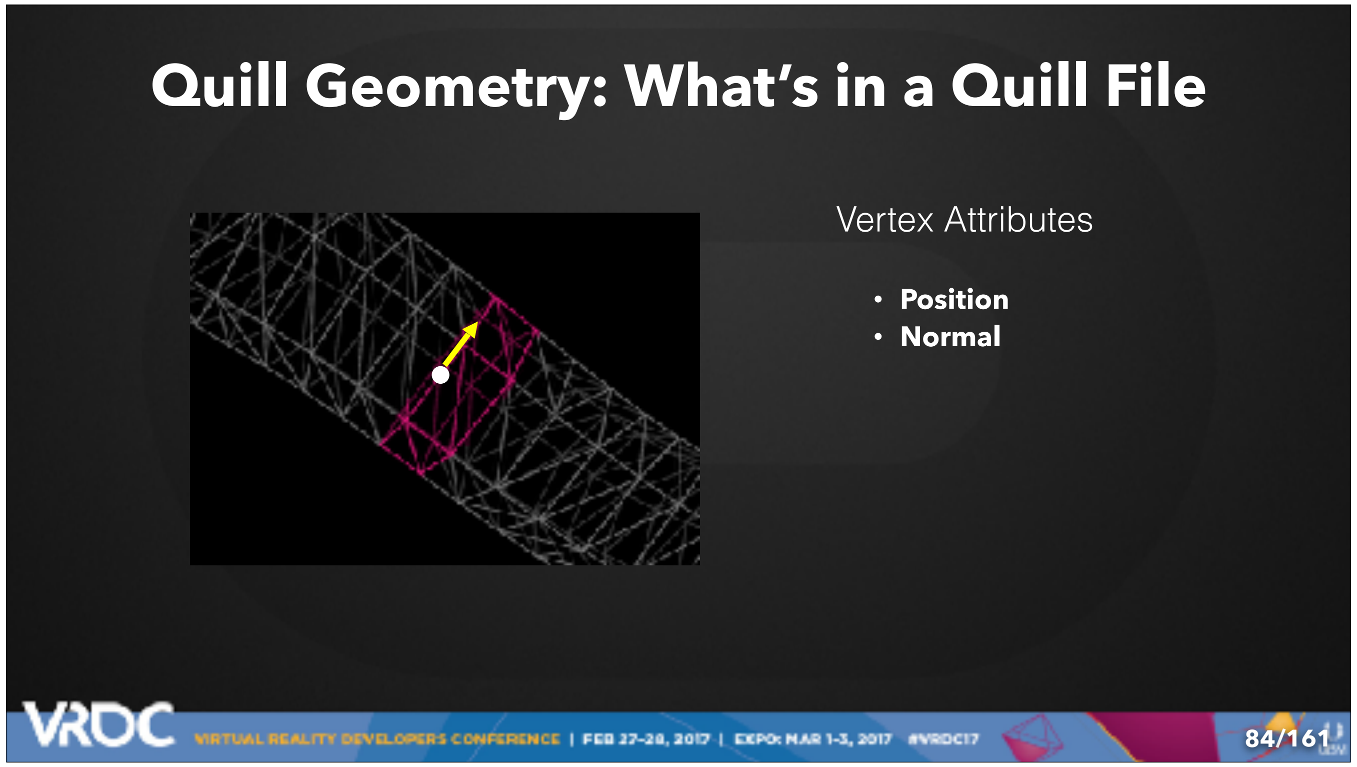
Here's that same rounded stroke unrolled, that's 3 primitives.

Quill Geometry: What’s in a Quill File

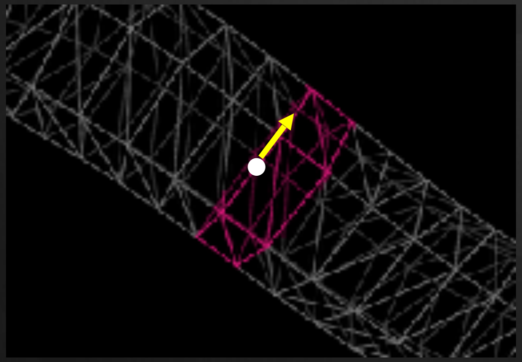


Vertex Attributes

We’ll finish the Quill file description with a look at the vertex attributes.



Quill Geometry: What’s in a Quill File

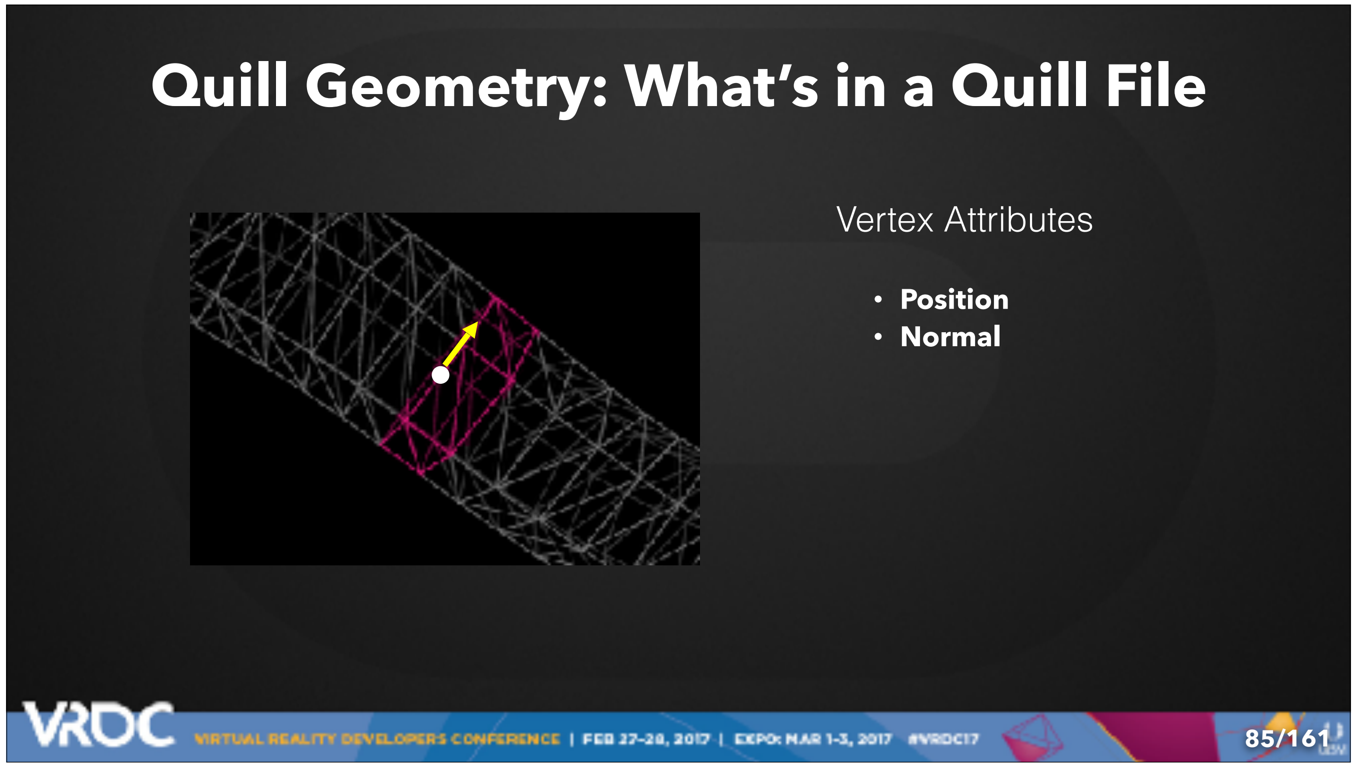


Vertex Attributes

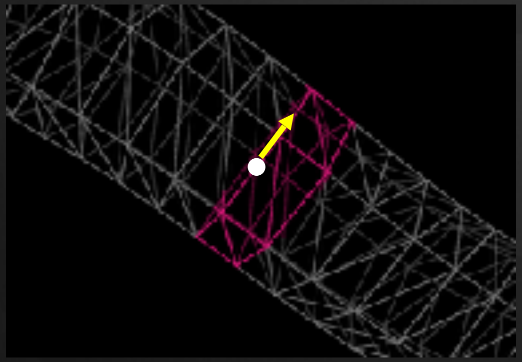
- **Position**
- **Normal**

VRDC VIRTUAL REALITY DEVELOPERS CONFERENCE | FEB 27-28, 2017 | EXPO: MAR 1-3, 2017 #VRDC17

Position and normal are both sampled from the controller and stored in layer space.



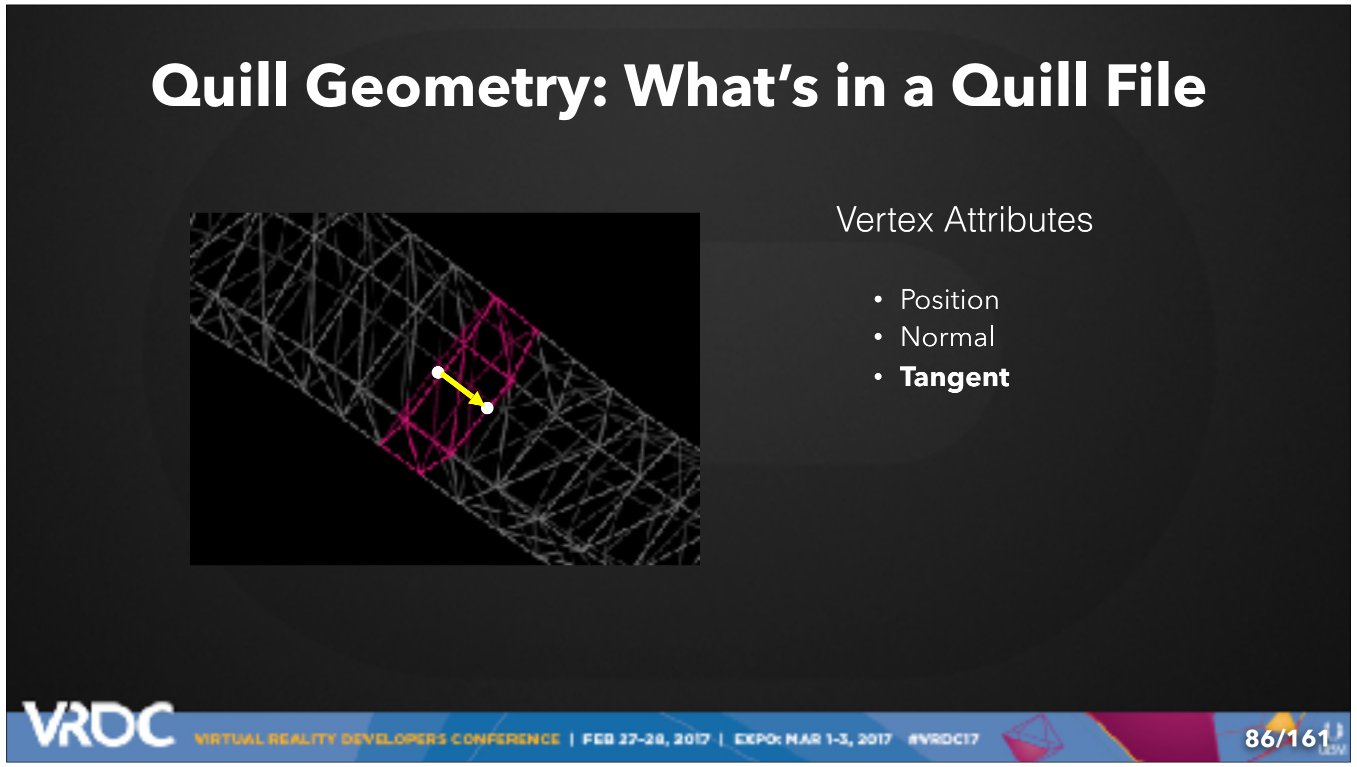
Quill Geometry: What's in a Quill File



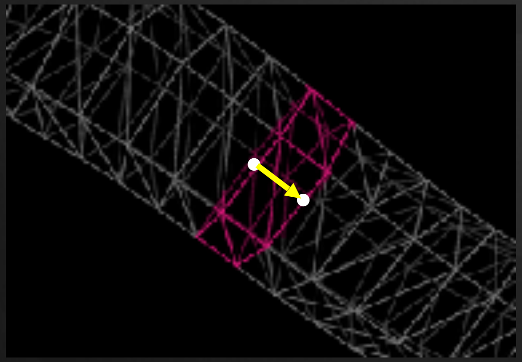
Vertex Attributes

- **Position**
- **Normal**

Normal provides a directionality to our strokes, in this case it's a major axis for the elliptical brush.



Quill Geometry: What's in a Quill File

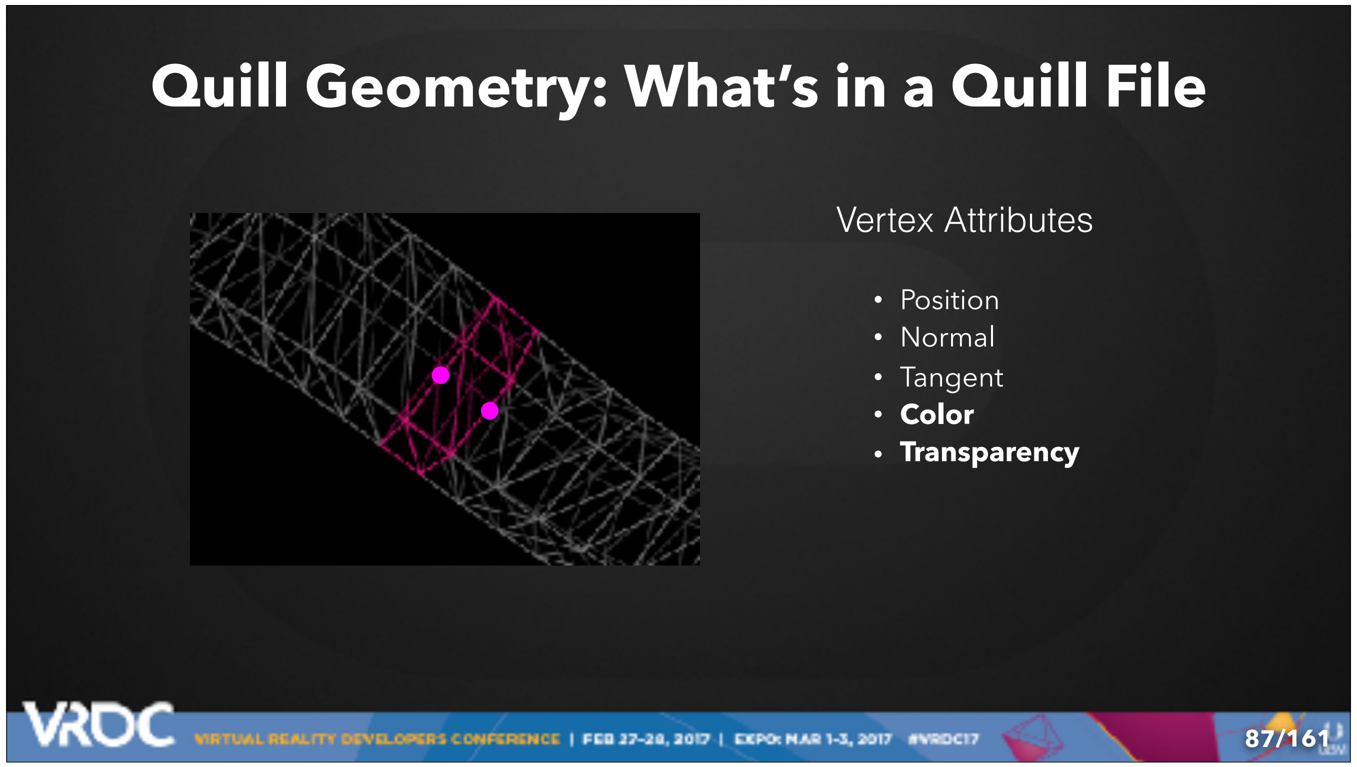


Vertex Attributes

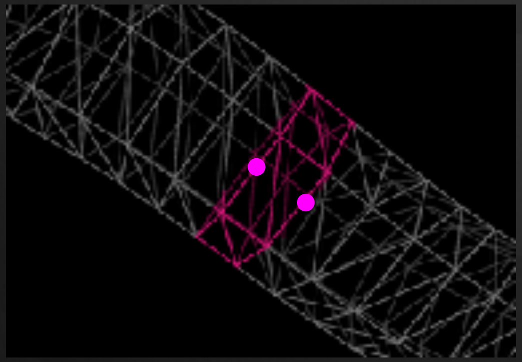
- Position
- Normal
- **Tangent**

VRDC VIRTUAL REALITY DEVELOPERS CONFERENCE | FEB 27-28, 2017 | EXPO: MAR 1-3, 2017 #VRDC17

Tangent indicates an averaged direction to the next vertex – a tangent to the overall curve at this vertex.



Quill Geometry: What’s in a Quill File

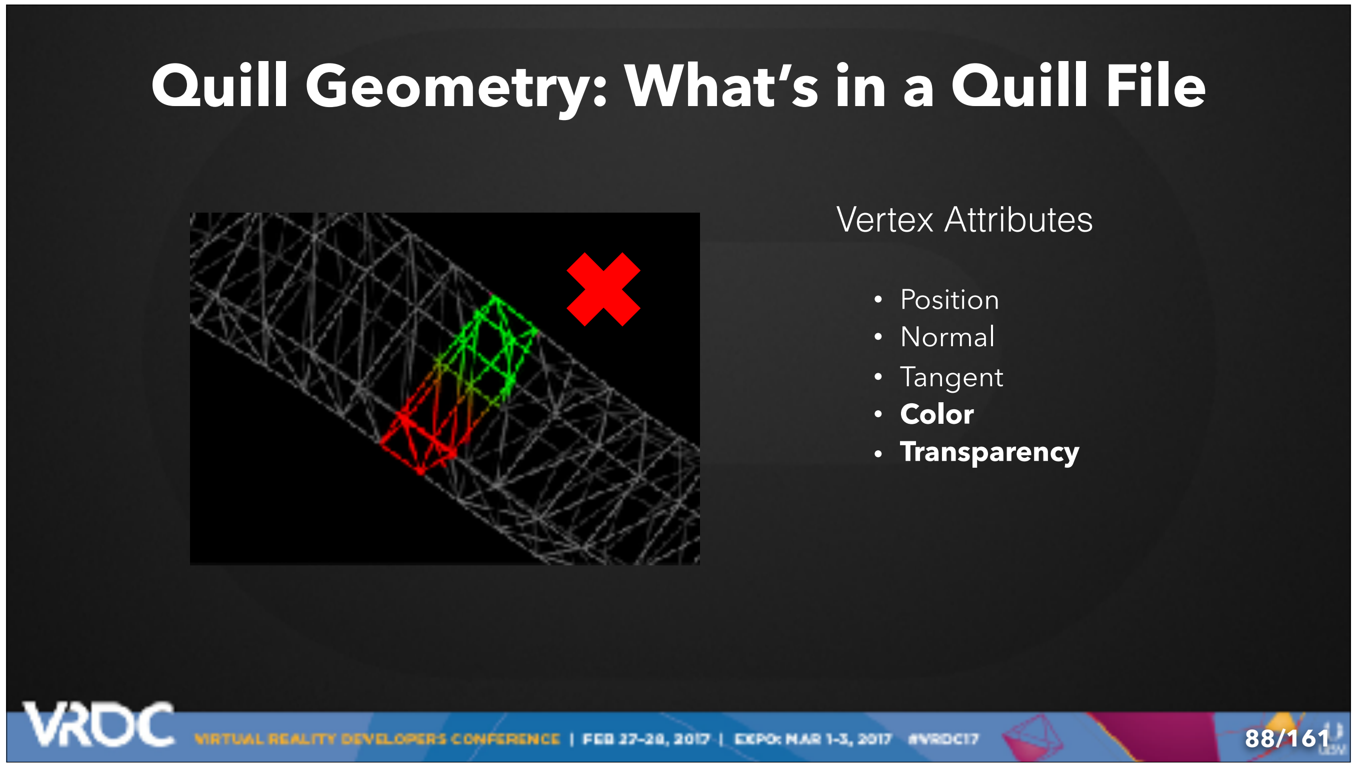


Vertex Attributes

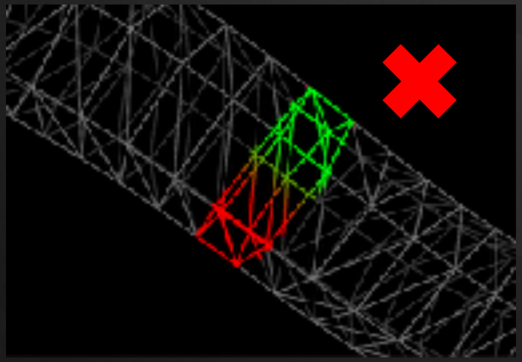
- Position
- Normal
- Tangent
- **Color**
- **Transparency**

VRDC VIRTUAL REALITY DEVELOPERS CONFERENCE | FEB 27-28, 2017 | EXPO: MAR 1-3, 2017 #VRDC17

Color and transparency also have vertex resolution.



Quill Geometry: What’s in a Quill File

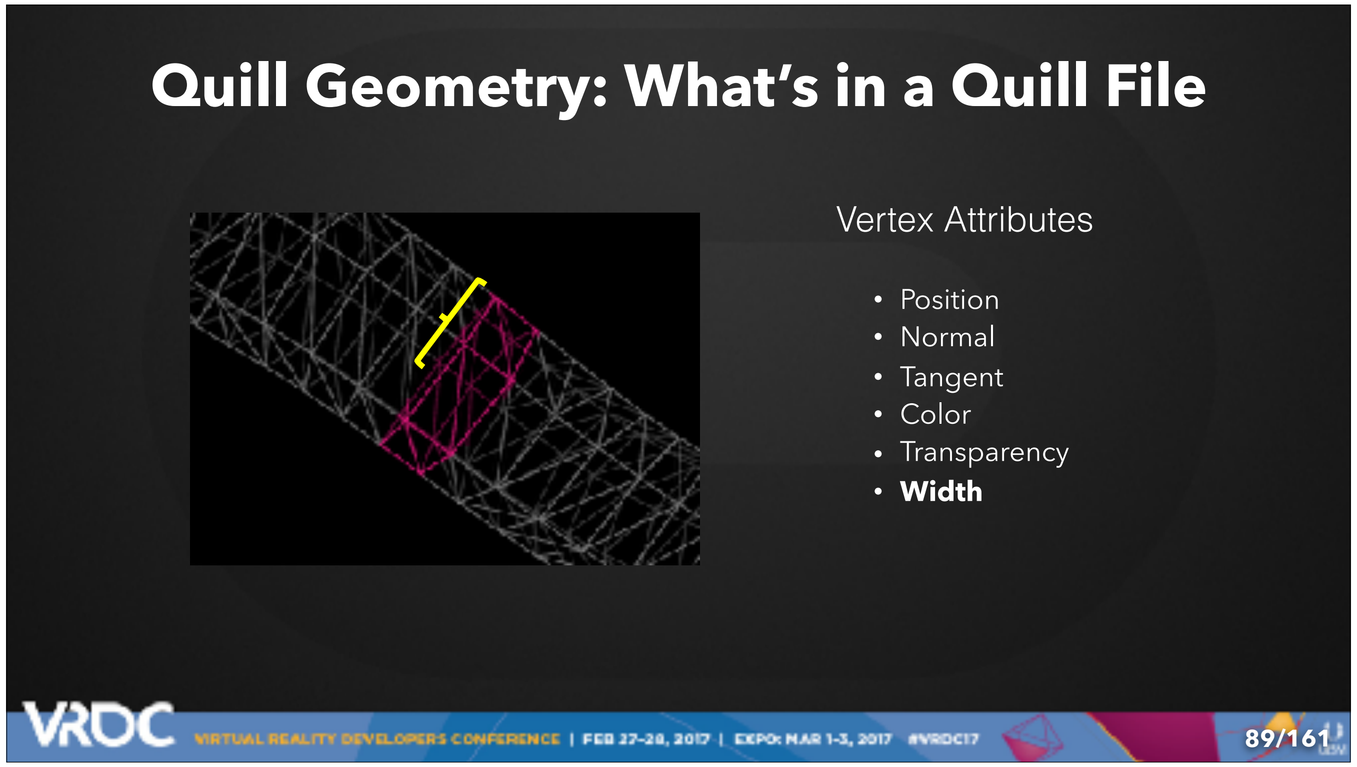


Vertex Attributes

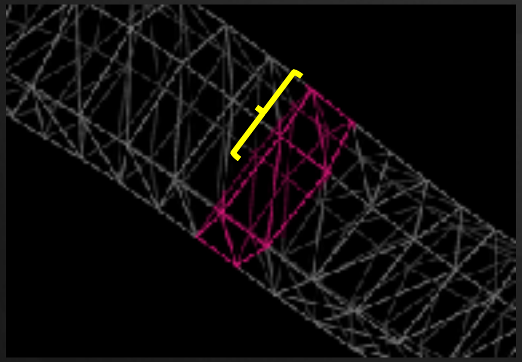
- Position
- Normal
- Tangent
- **Color**
- **Transparency**

VRDC VIRTUAL REALITY DEVELOPERS CONFERENCE | FEB 27-28, 2017 | EXPO: MAR 1-3, 2017 #VRDC17

So we couldn’t for example color this segment here green on the top and red on the bottom without using uv’s.



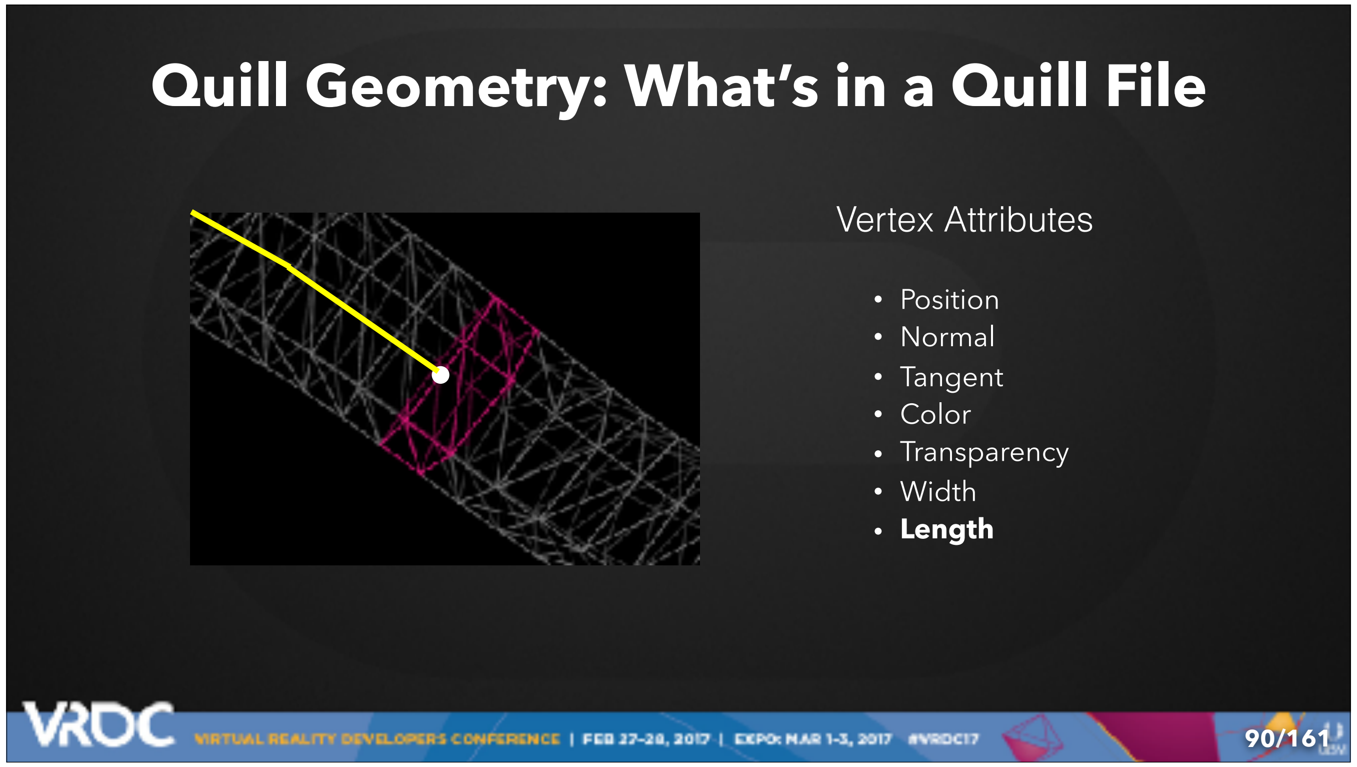
Quill Geometry: What's in a Quill File



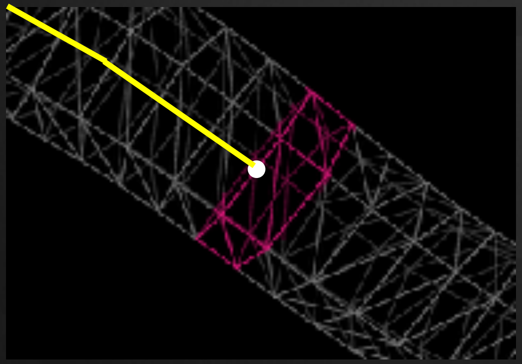
Vertex Attributes

- Position
- Normal
- Tangent
- Color
- Transparency
- **Width**

Width defines the radius of the stroke's major axis, and also has per vertex resolution.



Quill Geometry: What's in a Quill File

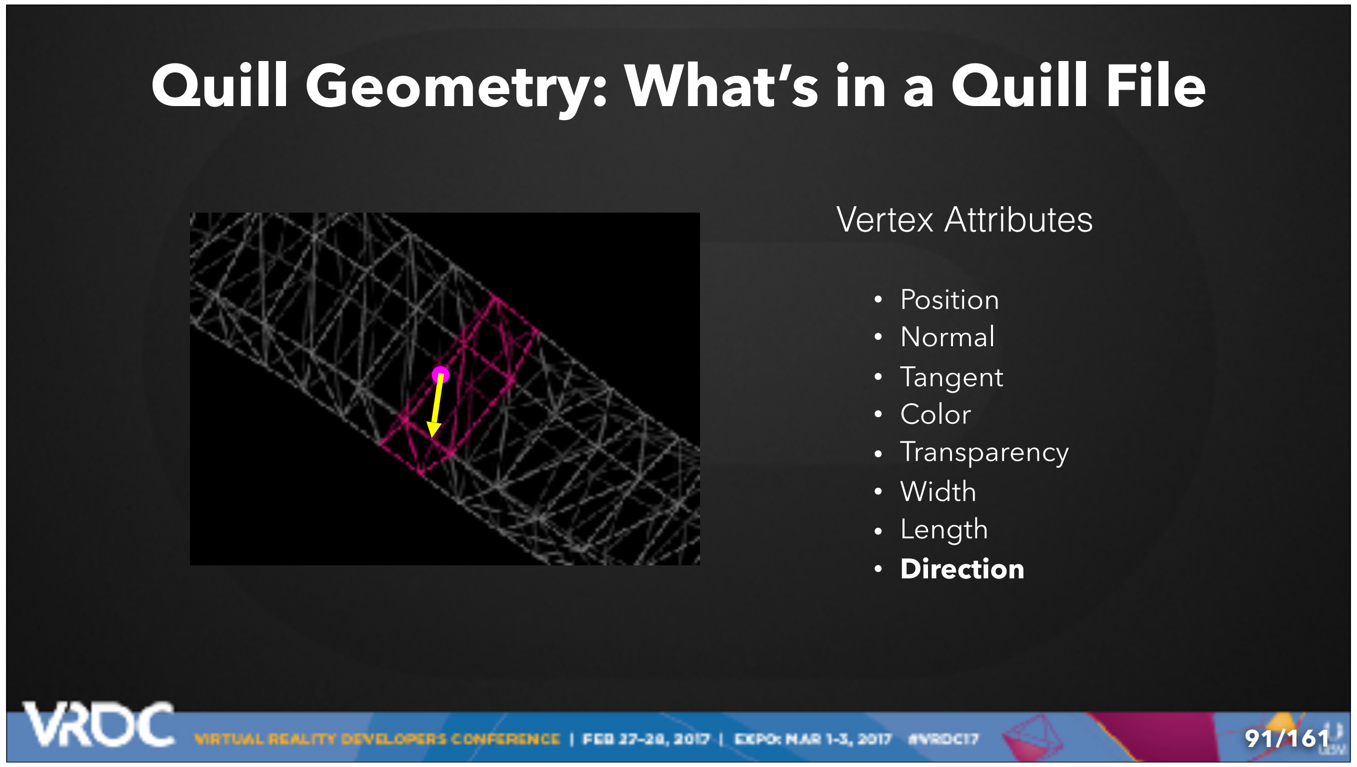


Vertex Attributes

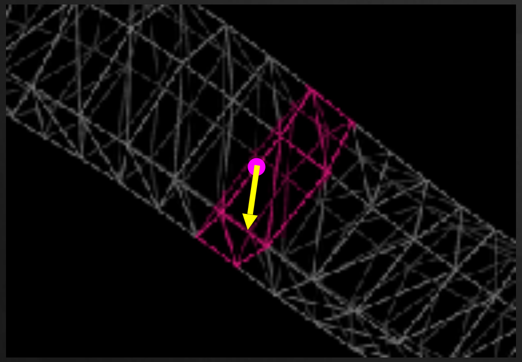
- Position
- Normal
- Tangent
- Color
- Transparency
- Width
- **Length**

VRDC VIRTUAL REALITY DEVELOPERS CONFERENCE | FEB 27-28, 2017 | EXPO: MAR 1-3, 2017 #VRDC17

Length is a computed value, it represents the local space distance along the stroke at that vertex. So if you had a 100cm stroke, a vertex towards the middle would have a length of 50cm.



Quill Geometry: What's in a Quill File

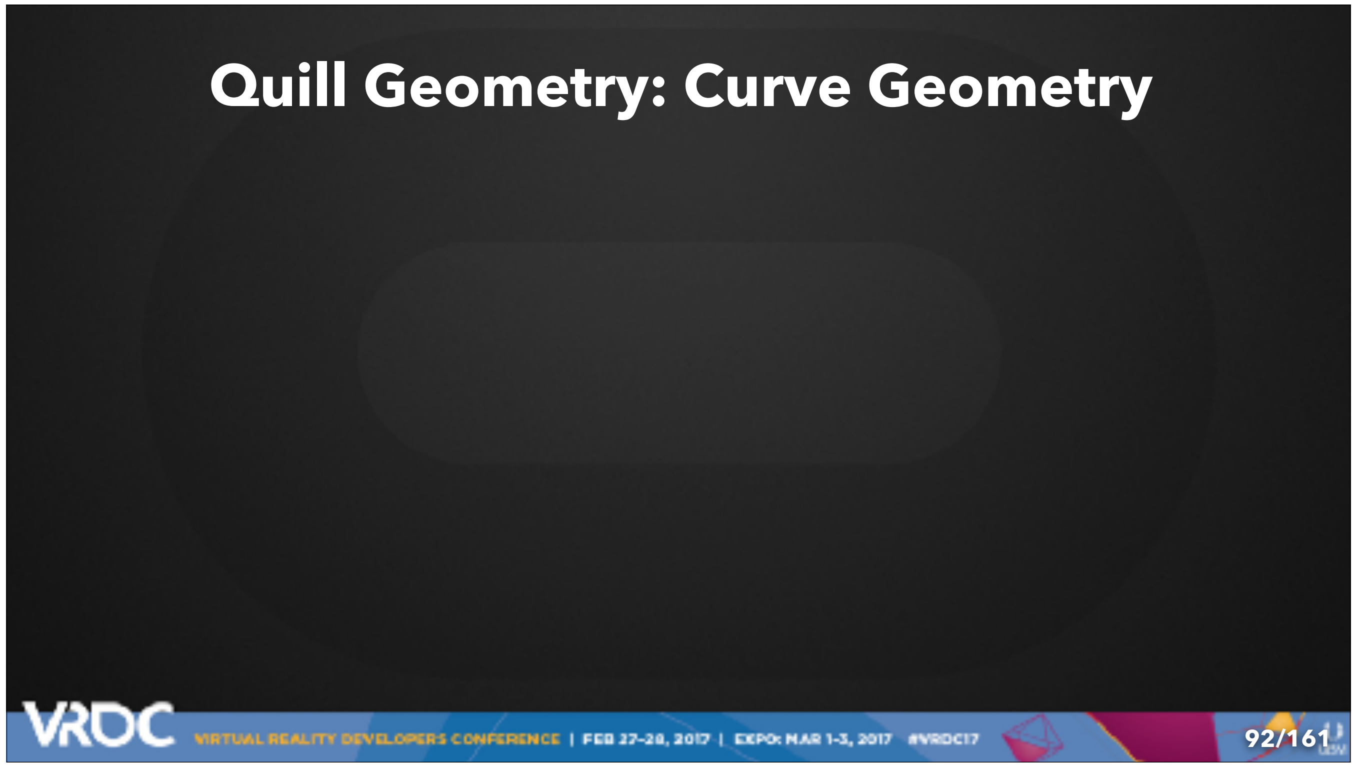


Vertex Attributes

- Position
- Normal
- Tangent
- Color
- Transparency
- Width
- Length
- **Direction**

VRDC VIRTUAL REALITY DEVELOPERS CONFERENCE | FEB 27-28, 2017 | EXPO: MAR 1-3, 2017 #VRDC17

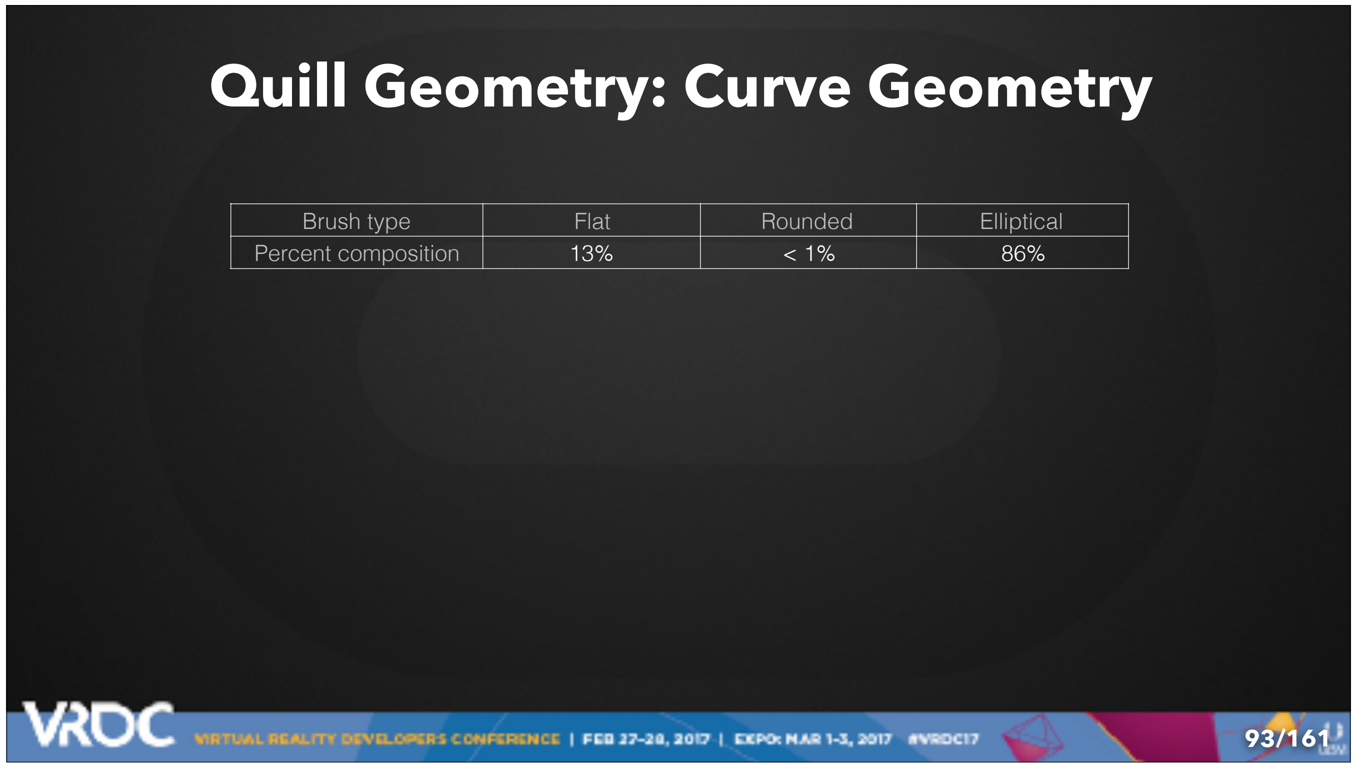
Finally, the Quill application stores the direction to the artist HMD in local space. This can be used for view dependent effects, but was not preserved in Dear Angelica (it was assimilated by the animation pipeline).



Quill Geometry: Curve Geometry

VIRTUAL REALITY DEVELOPERS CONFERENCE | FEB 27-28, 2017 | EXPO: MAR 1-3, 2017 #VRDC17

I have just a couple remarks regarding using curve geometry on Dear Angelica.

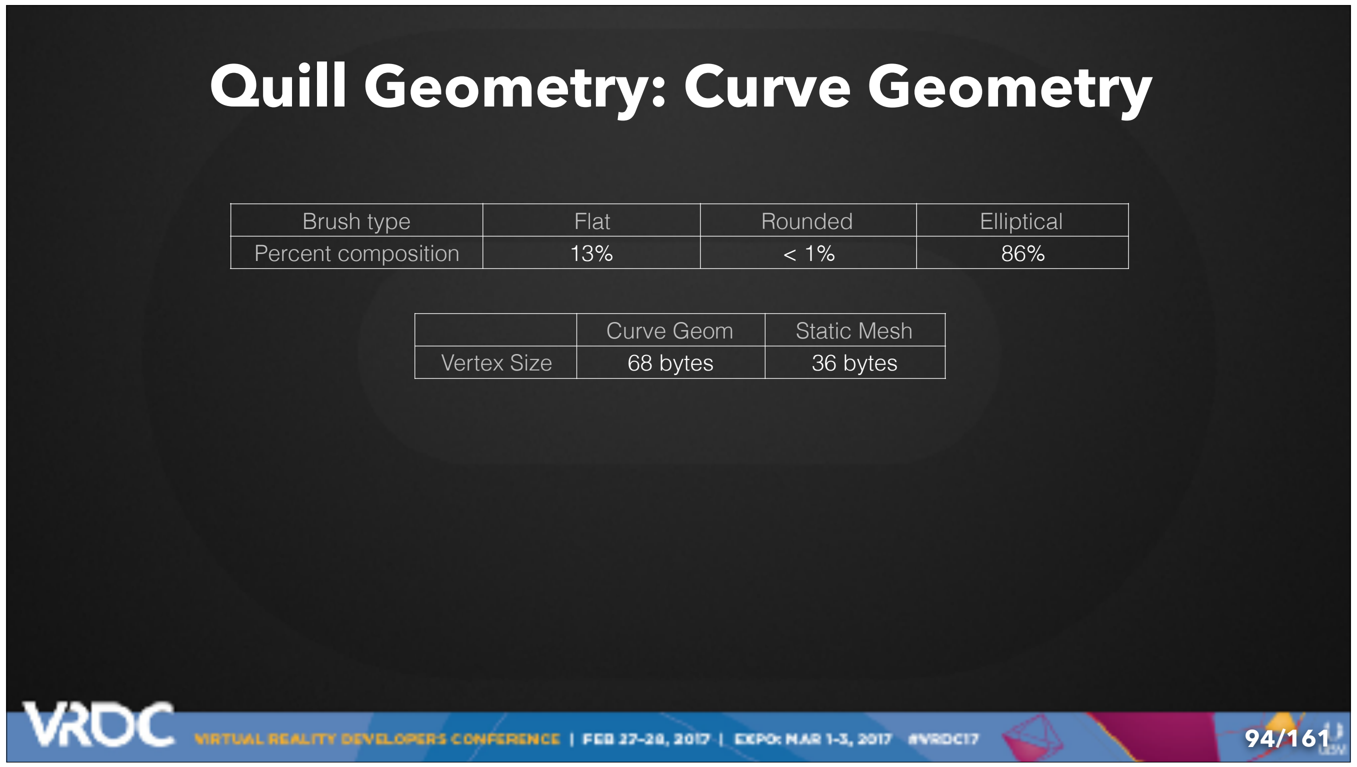


Quill Geometry: Curve Geometry

Brush type	Flat	Rounded	Elliptical
Percent composition	13%	< 1%	86%

VRDC VIRTUAL REALITY DEVELOPERS CONFERENCE | FEB 27-28, 2017 | EXPO: MAR 1-3, 2017 #VRDC17

If we tally up the various brush types used across the entire project, factor in how much they were expanded in tessellation,



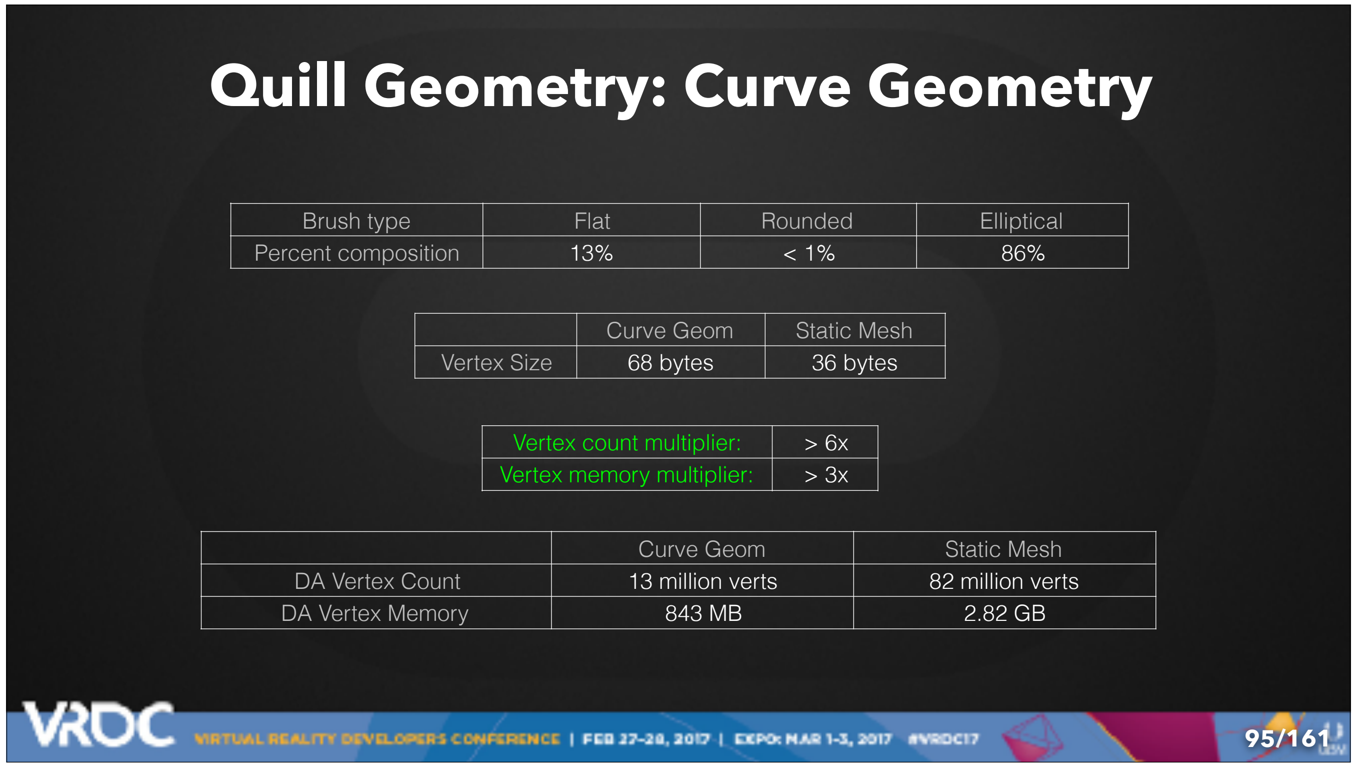
Quill Geometry: Curve Geometry

Brush type	Flat	Rounded	Elliptical
Percent composition	13%	< 1%	86%

	Curve Geom	Static Mesh
Vertex Size	68 bytes	36 bytes

VRDC VIRTUAL REALITY DEVELOPERS CONFERENCE | FEB 27-28, 2017 | EXPO: MAR 1-3, 2017 #VRDC17

and taking into account our larger vertex size compared to what we would have needed on static meshes, we cut our memory requirements by a factor of almost 3. And that's just for geometry memory, not including textures.



Quill Geometry: Curve Geometry

Brush type	Flat	Rounded	Elliptical
Percent composition	13%	< 1%	86%

	Curve Geom	Static Mesh
Vertex Size	68 bytes	36 bytes

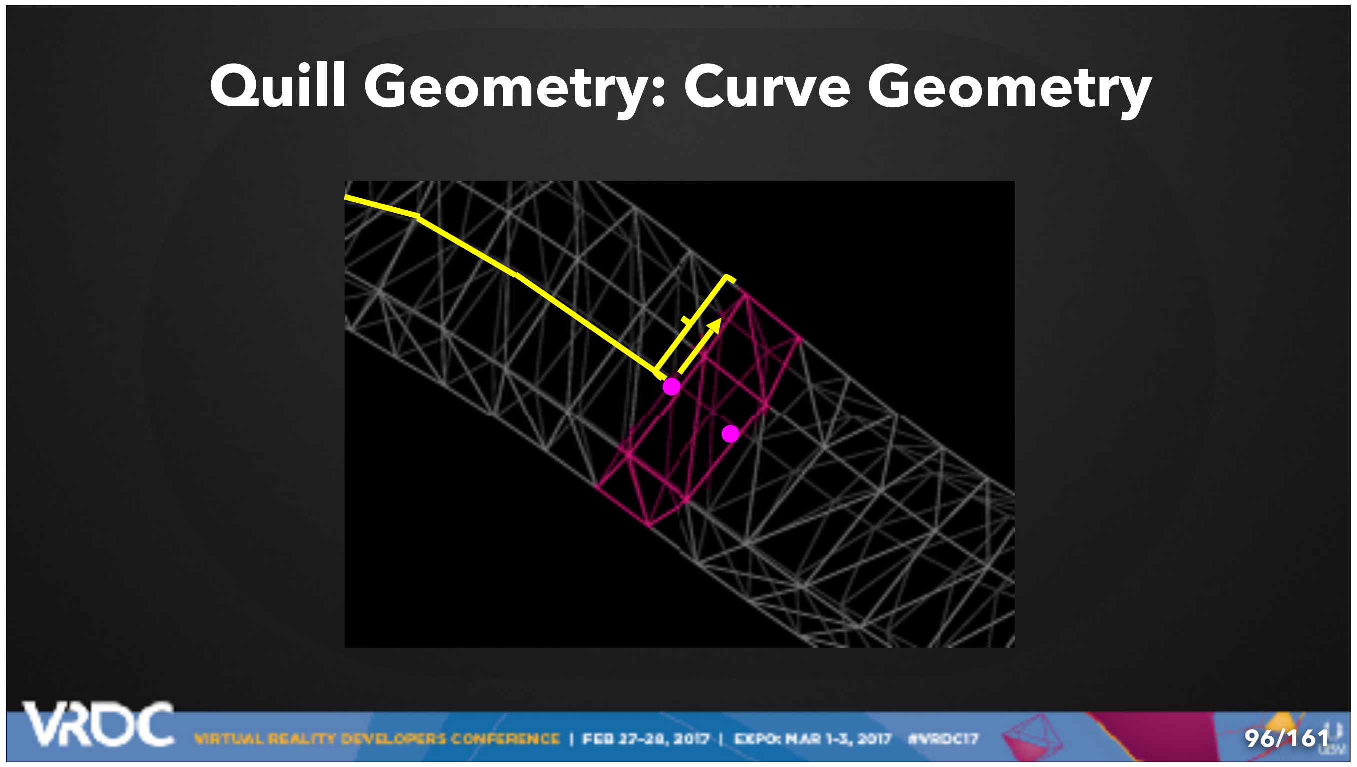
Vertex count multiplier:	> 6x
Vertex memory multiplier:	> 3x

	Curve Geom	Static Mesh
DA Vertex Count	13 million verts	82 million verts
DA Vertex Memory	843 MB	2.82 GB

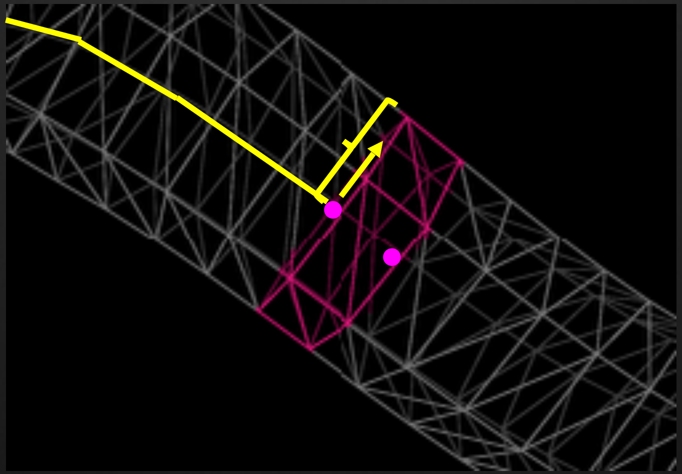
VRDC VIRTUAL REALITY DEVELOPERS CONFERENCE | FEB 27-28, 2017 | EXPO: MAR 1-3, 2017 #VRDC17

The whole project came in at less than 1GB for the geometry data, whereas it would have been on the order of 3GB had we used static meshes. When you add in position textures used to do animations, it quickly gets prohibitively expensive.

…

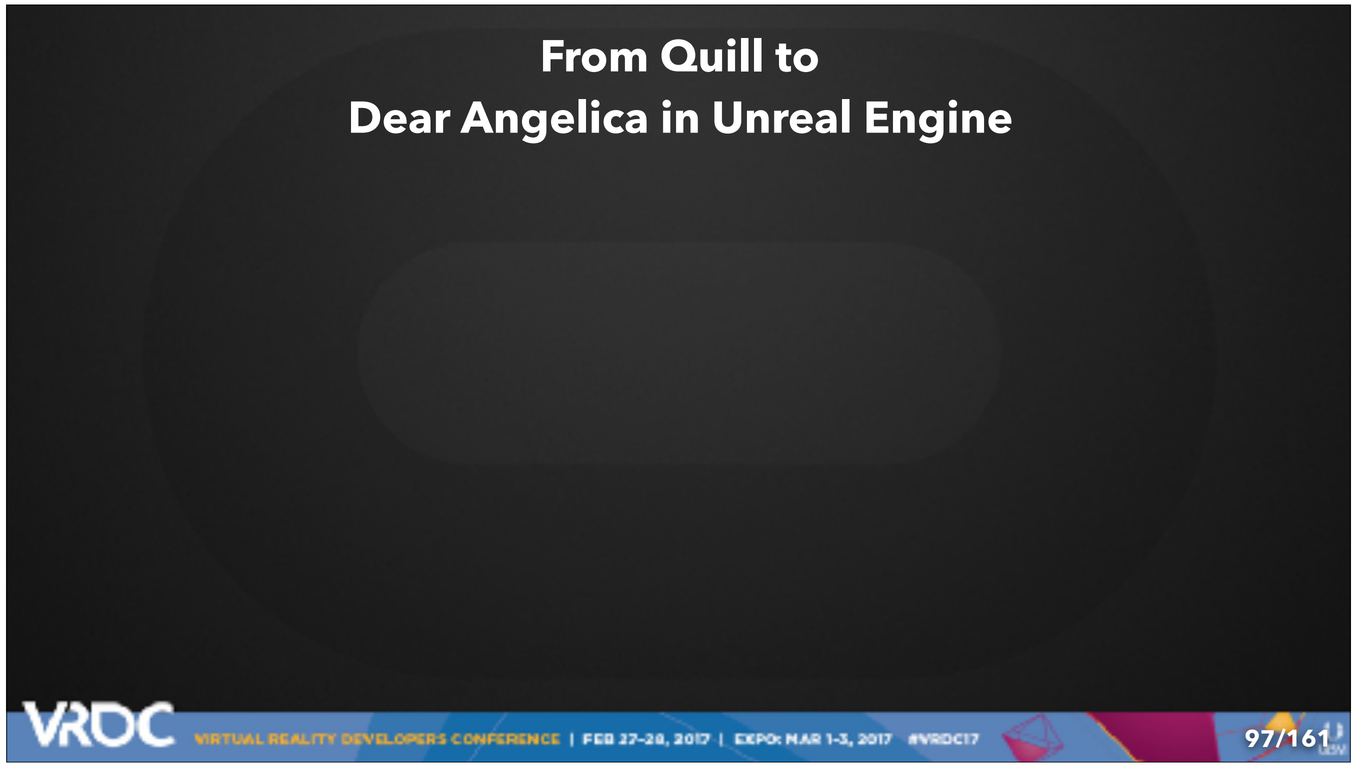


Quill Geometry: Curve Geometry



On a different note, it's also worth mentioning that when you are working with vertices in the precise way they were authored, as curve geometry, you're working in the art's natural domain, so effects can target the tessellation properties. At early stages in the project when we were using static meshes, we had to jump through extra hoops to bring in the source domain.

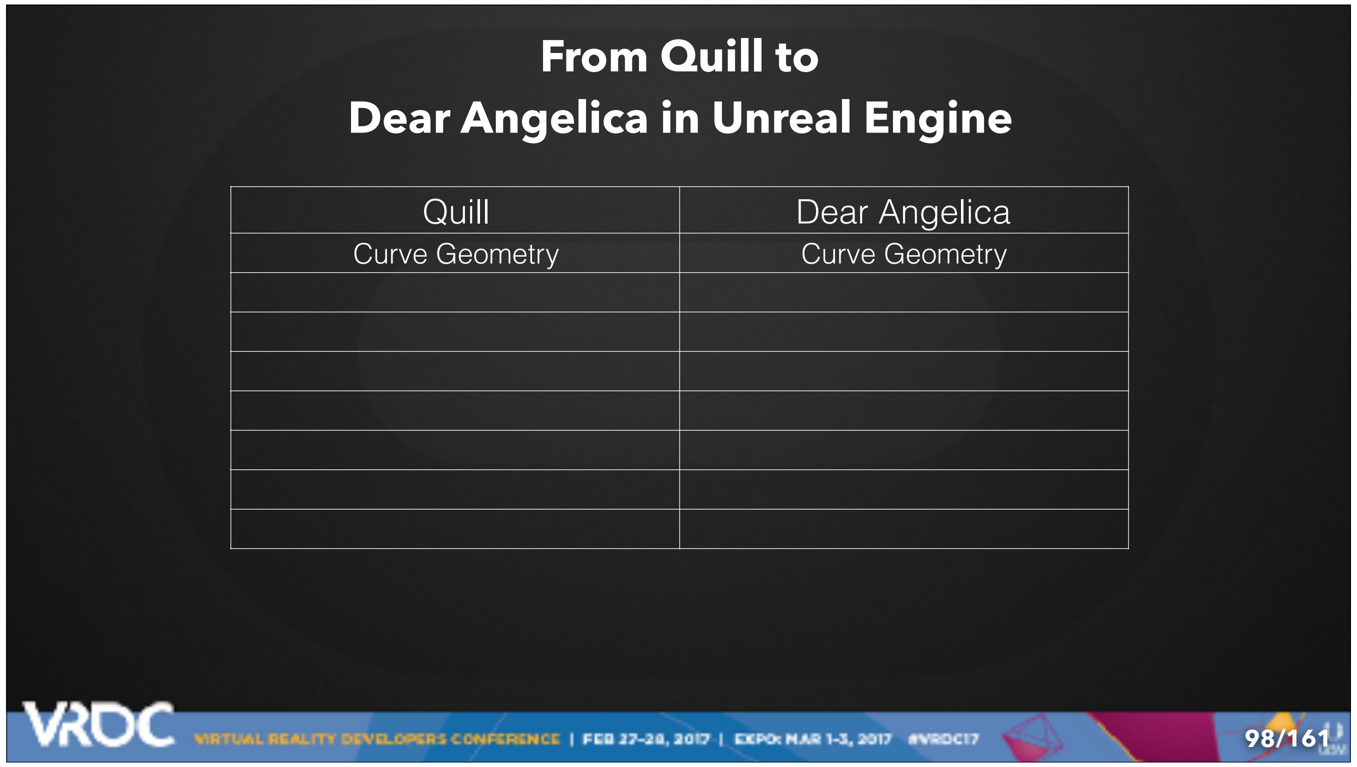
So this wraps up specifics about the geometry and the techniques for drawing it.



From Quill to Dear Angelica in Unreal Engine

VRDC VIRTUAL REALITY DEVELOPERS CONFERENCE | FEB 27-28, 2017 | EXPO: MAR 1-3, 2017 #VRDC17

Here I just wanted to quickly compare and contrast drawing Dear Angelica geometry in the native application vs using it in Unreal Engine. This was an interesting port in that we were taking geometry out of its ideal environment, so it's helpful to take inventory of where we might expect things to perform differently.

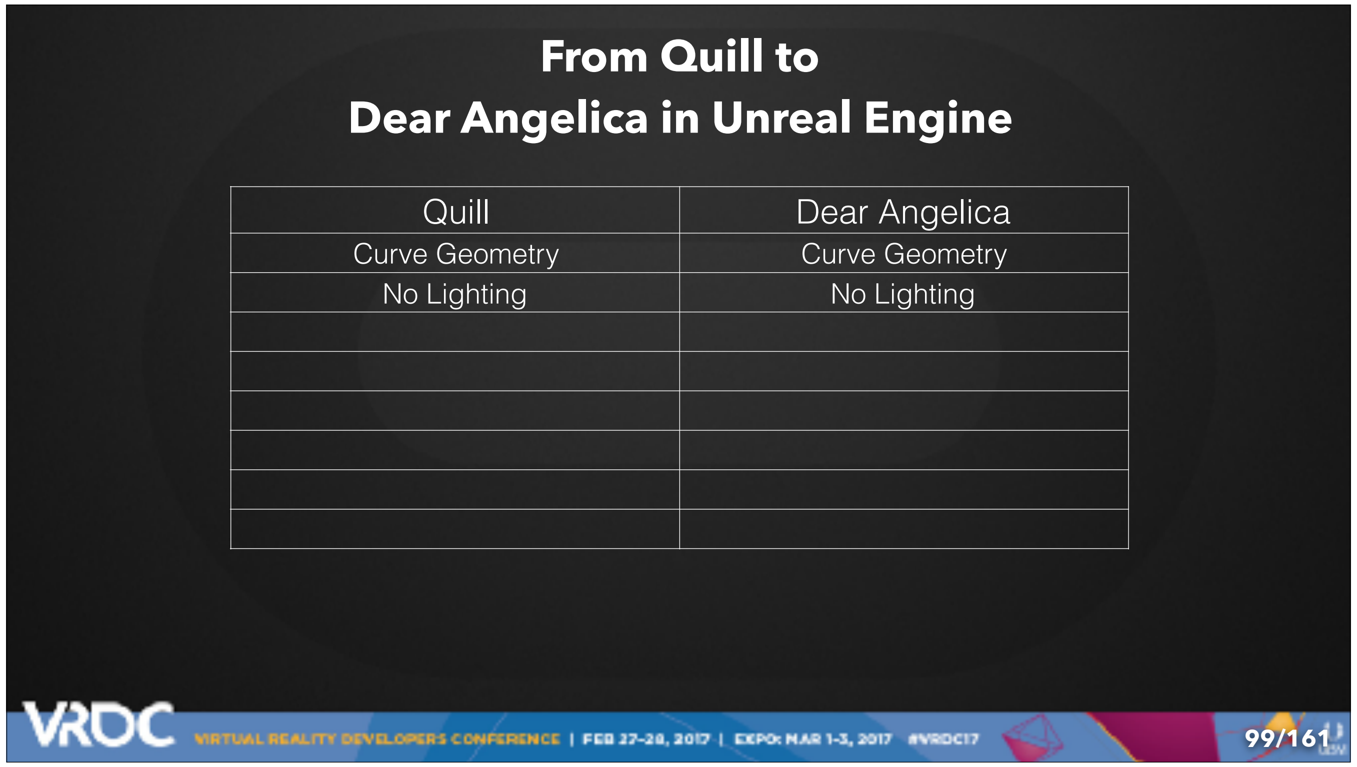


From Quill to Dear Angelica in Unreal Engine

Quill	Dear Angelica
Curve Geometry	Curve Geometry

VRDC VIRTUAL REALITY DEVELOPERS CONFERENCE | FEB 27-28, 2017 | EXPO: MAR 1-3, 2017 #VRDC17

They both used curve geometry.

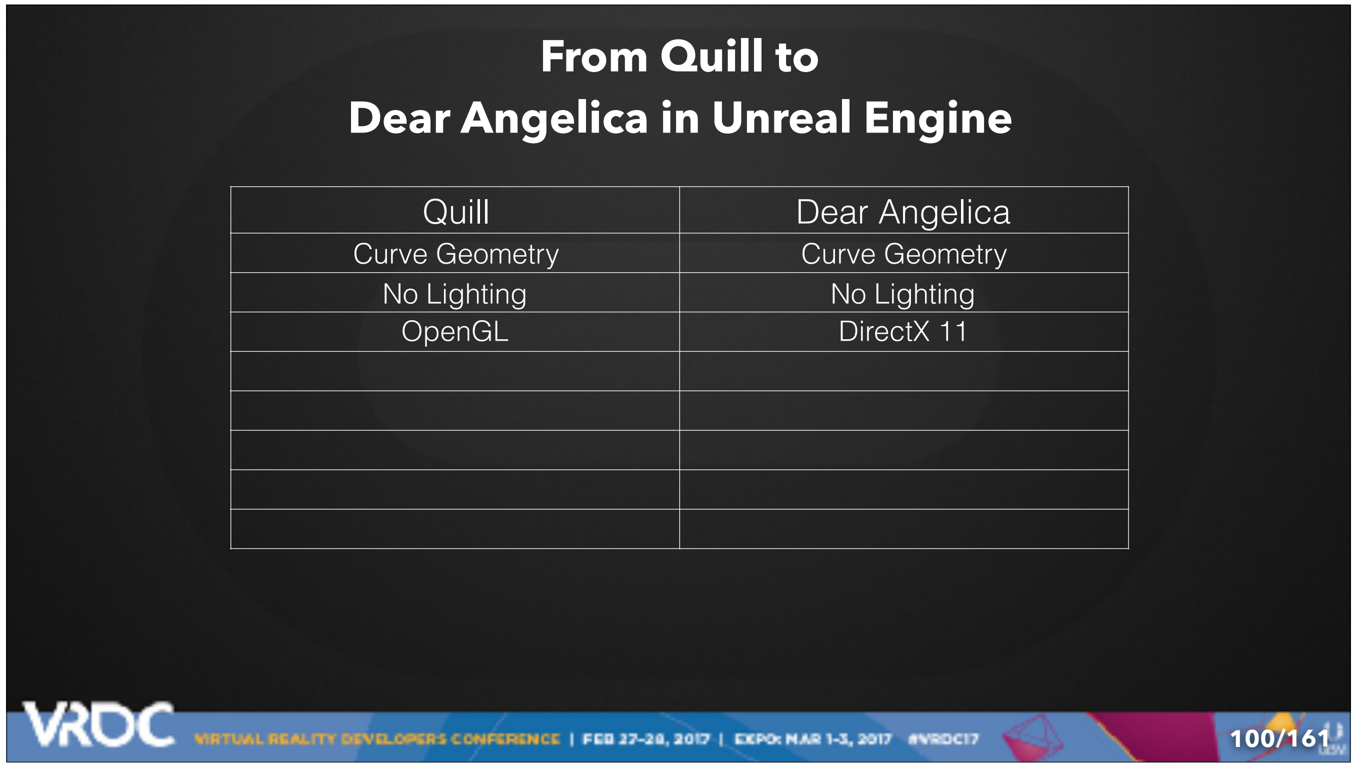


From Quill to Dear Angelica in Unreal Engine

Quill	Dear Angelica
Curve Geometry	Curve Geometry
No Lighting	No Lighting

VRDC VIRTUAL REALITY DEVELOPERS CONFERENCE | FEB 27-28, 2017 | EXPO: MAR 1-3, 2017 #VRDC17

They both were unlit.

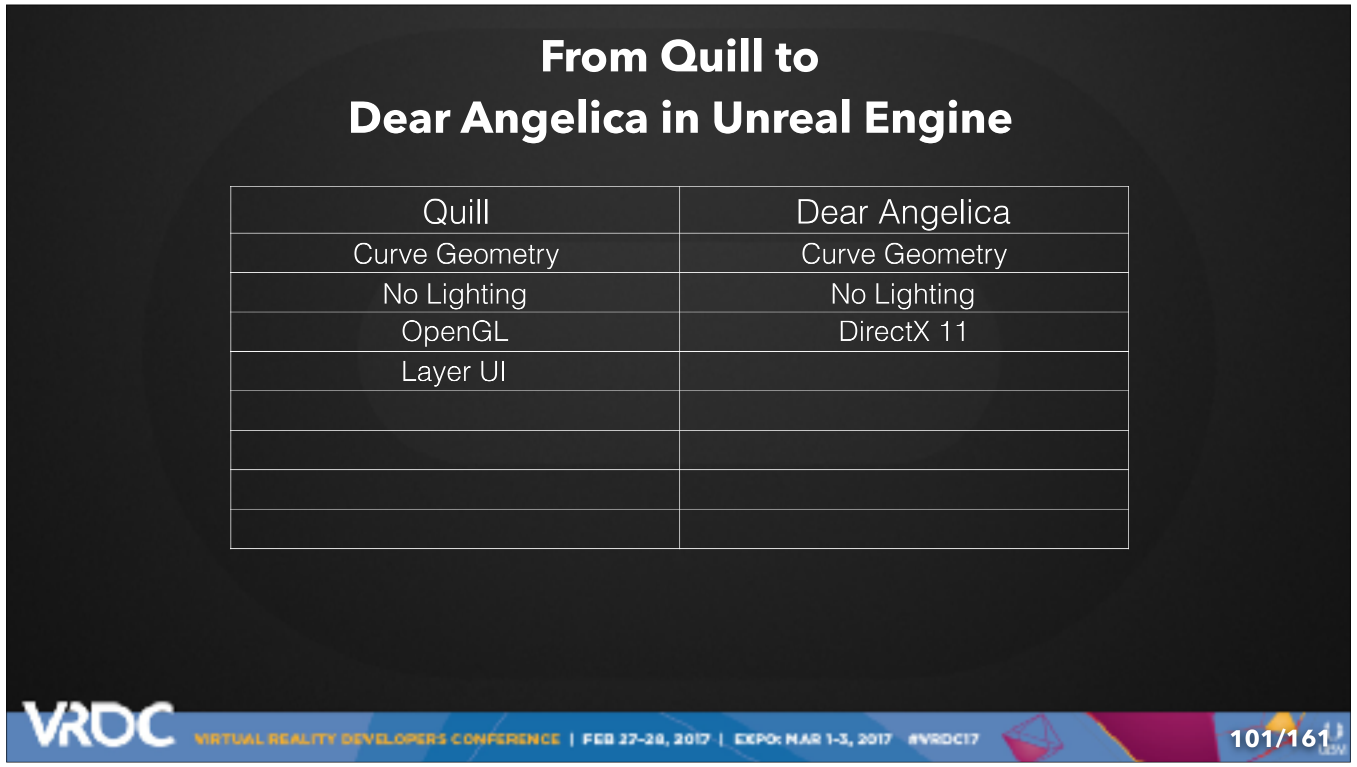


From Quill to Dear Angelica in Unreal Engine

Quill	Dear Angelica
Curve Geometry	Curve Geometry
No Lighting	No Lighting
OpenGL	DirectX 11

VRDC VIRTUAL REALITY DEVELOPERS CONFERENCE | FEB 27-28, 2017 | EXPO: MAR 1-3, 2017 #VRDC17

They used different API's, no problem, especially since we weren't relying on using any of the relatively new stereo extensions.

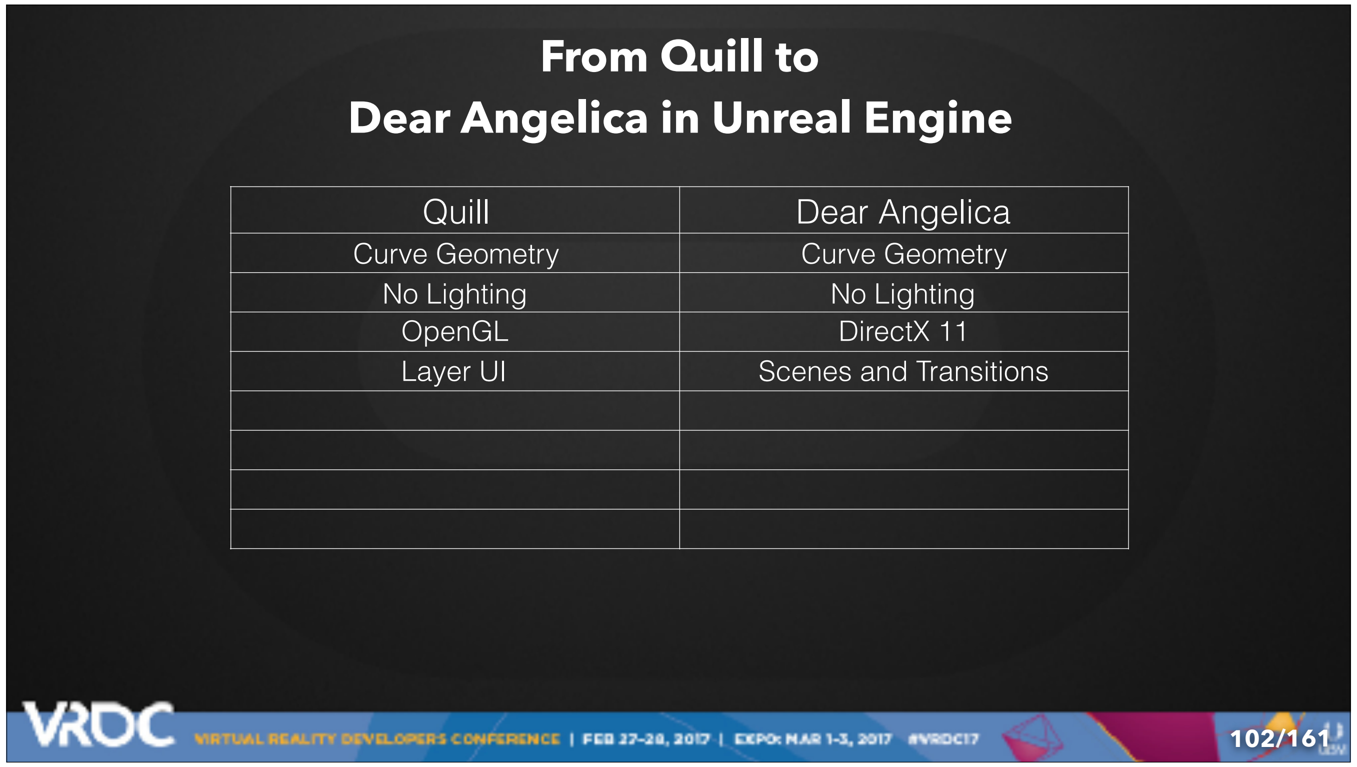


From Quill to Dear Angelica in Unreal Engine

Quill	Dear Angelica
Curve Geometry	Curve Geometry
No Lighting	No Lighting
OpenGL	DirectX 11
Layer UI	

VRDC VIRTUAL REALITY DEVELOPERS CONFERENCE | FEB 27-28, 2017 | EXPO: MAR 1-3, 2017 #VRDC17

Quill allowed turning on and off individual layers, generally used for isolating just a few things to work on at a time, or having a few different versions to flip between, but it also meant you might be looking at just a part of the whole scene.

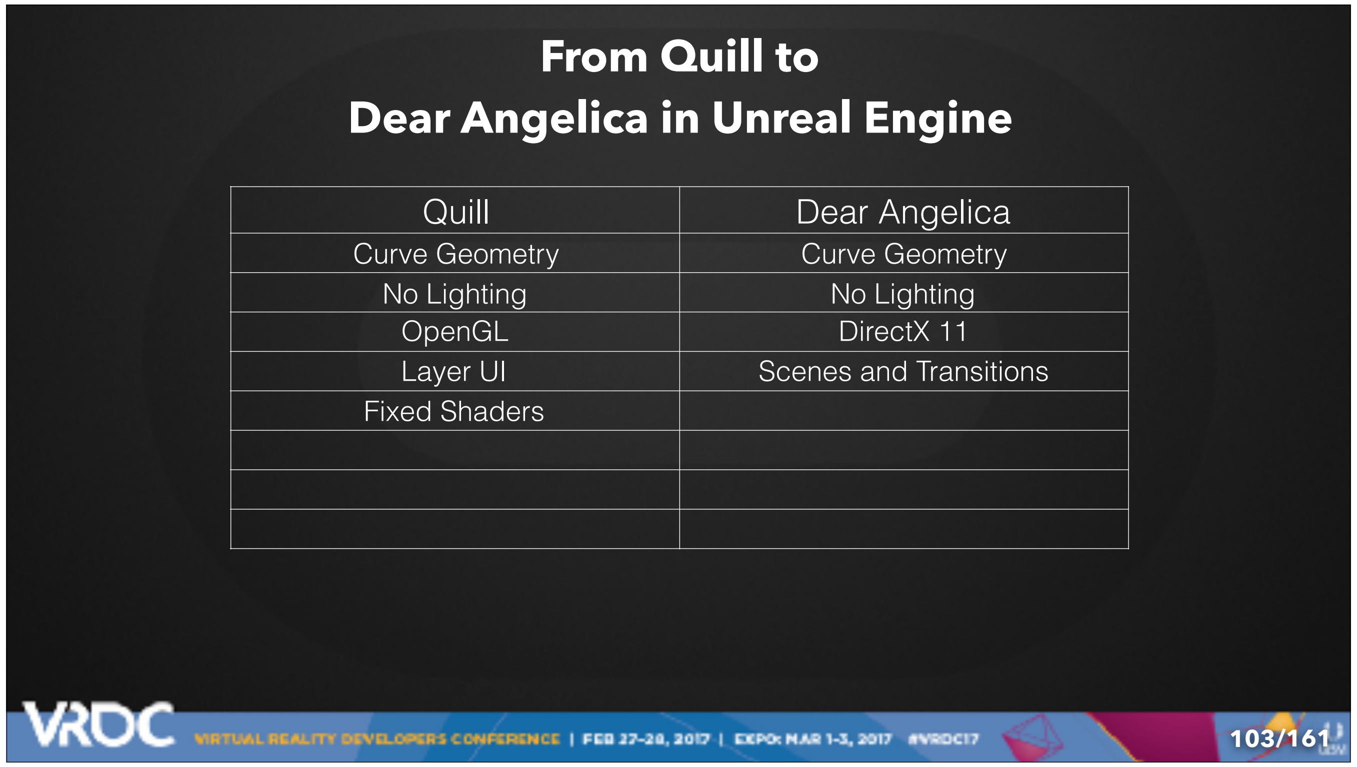


From Quill to Dear Angelica in Unreal Engine

Quill	Dear Angelica
Curve Geometry	Curve Geometry
No Lighting	No Lighting
OpenGL	DirectX 11
Layer UI	Scenes and Transitions

VRDC VIRTUAL REALITY DEVELOPERS CONFERENCE | FEB 27-28, 2017 | EXPO: MAR 1-3, 2017 #VRDC17

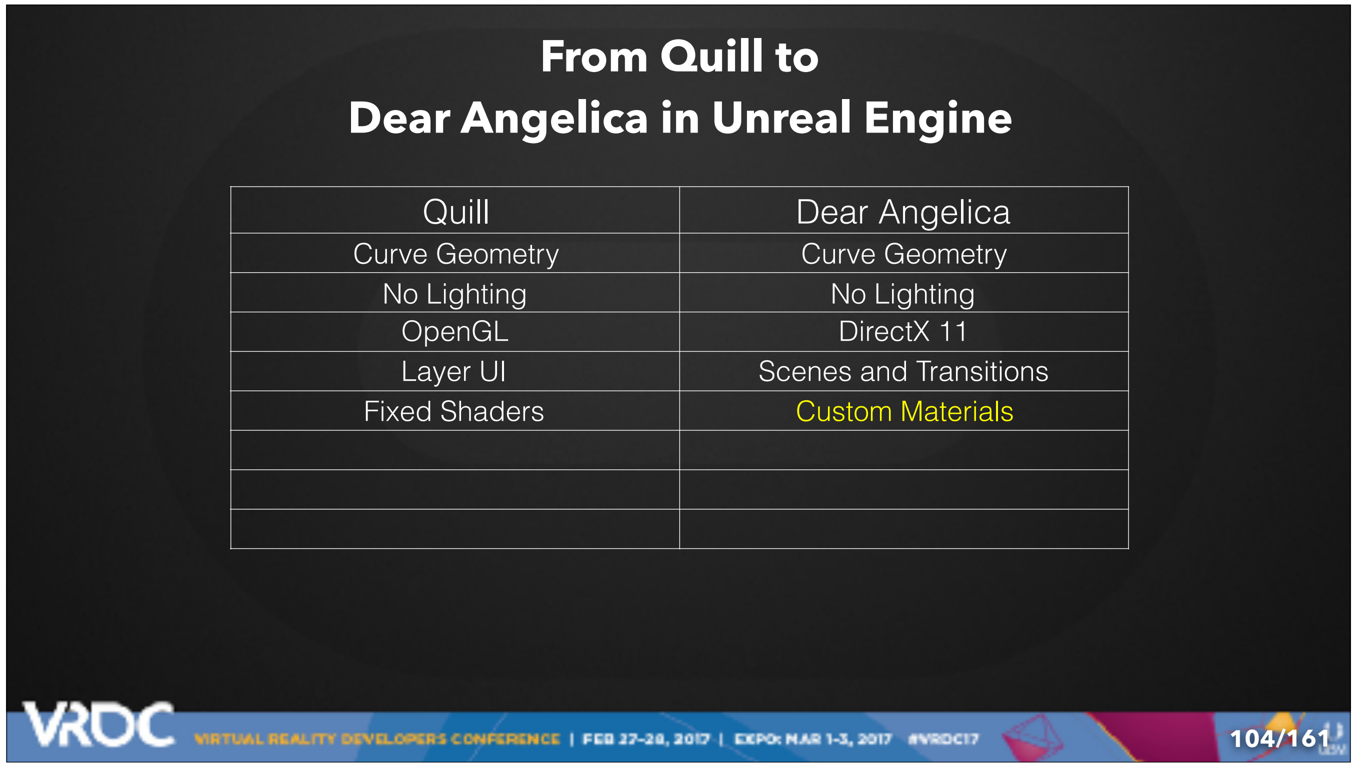
In Dear Angelica on the other hand, we would be drawing in entire scenes. In addition, we might have some cases of temporary heavy geometry use as the old scene transitioned out and the new one came in. But actually that wasn’t a big issue, since in DA scenes typically used the more sparse bedroom area as a hub.



From Quill to Dear Angelica in Unreal Engine

Quill	Dear Angelica
Curve Geometry	Curve Geometry
No Lighting	No Lighting
OpenGL	DirectX 11
Layer UI	Scenes and Transitions
Fixed Shaders	

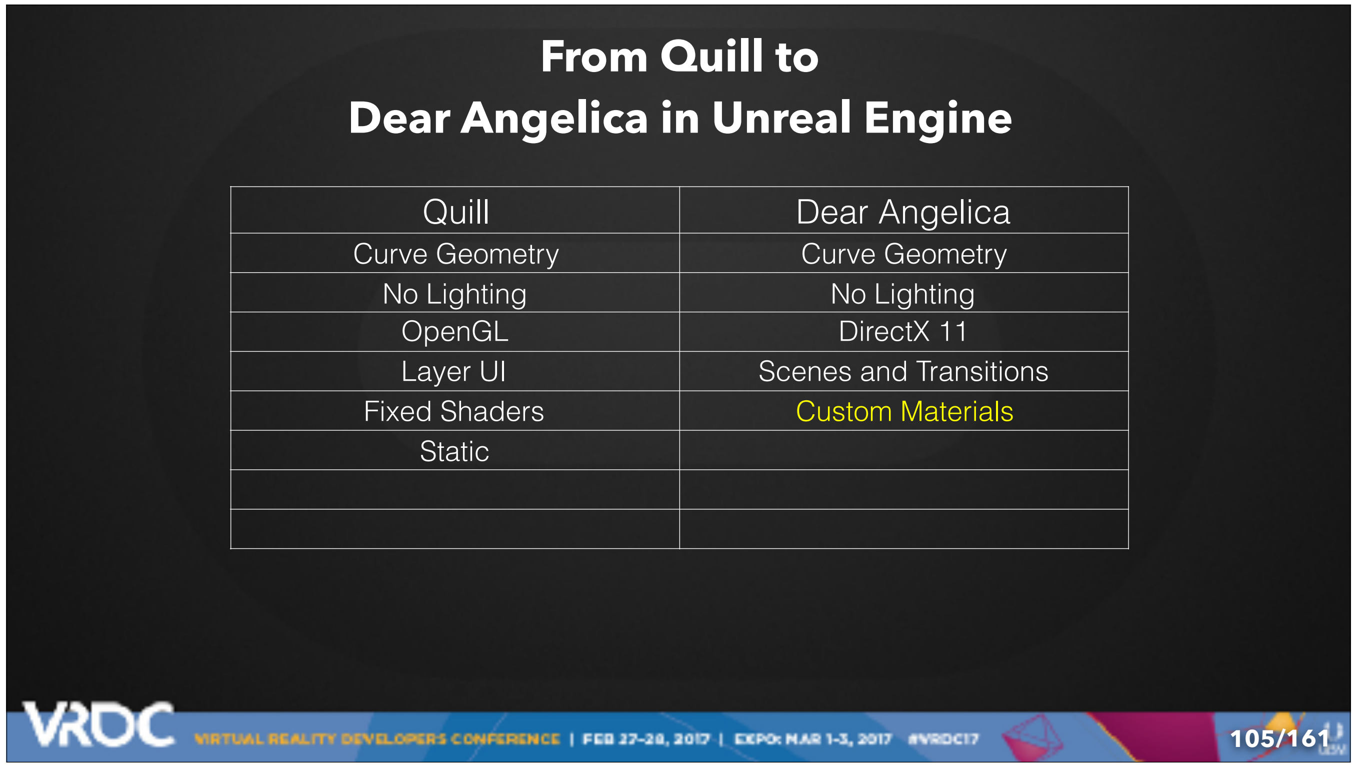
Quill used very constant techniques for producing strokes, with very few effects.



From Quill to Dear Angelica in Unreal Engine

Quill	Dear Angelica
Curve Geometry	Curve Geometry
No Lighting	No Lighting
OpenGL	DirectX 11
Layer UI	Scenes and Transitions
Fixed Shaders	Custom Materials

In DA we were expecting complex material expressions, so that would be something to keep an eye on.

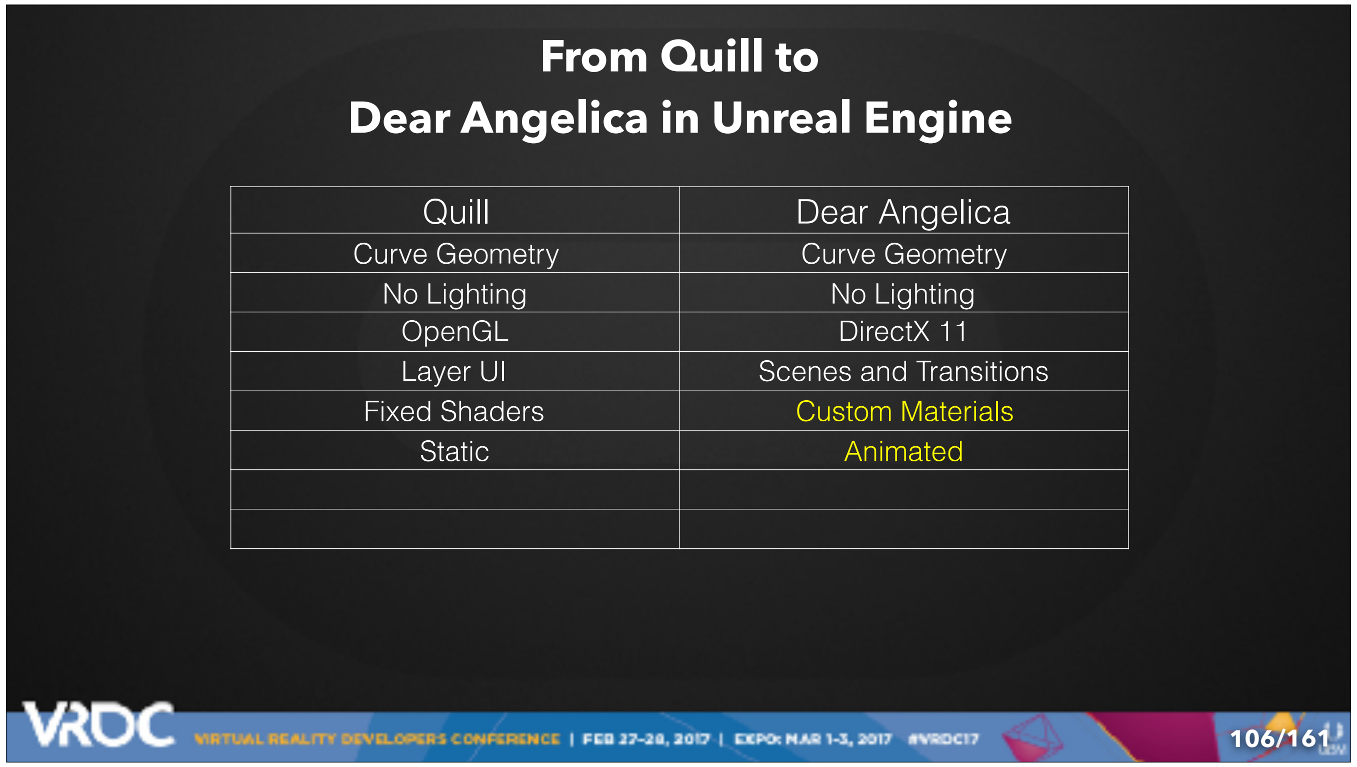


From Quill to Dear Angelica in Unreal Engine

Quill	Dear Angelica
Curve Geometry	Curve Geometry
No Lighting	No Lighting
OpenGL	DirectX 11
Layer UI	Scenes and Transitions
Fixed Shaders	Custom Materials
Static	

VRDC VIRTUAL REALITY DEVELOPERS CONFERENCE | FEB 27-28, 2017 | EXPO: MAR 1-3, 2017 #VRDC17

Again, very few effects in Quill…

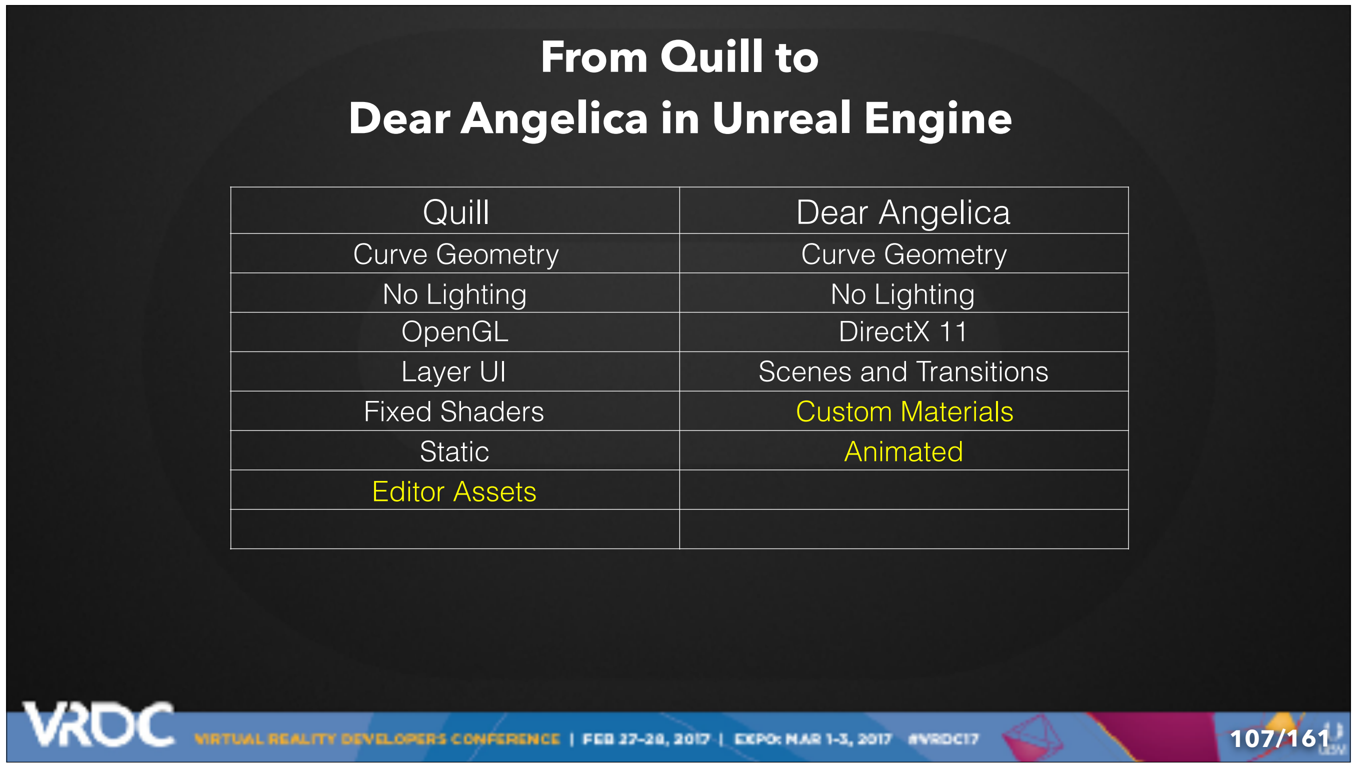


From Quill to Dear Angelica in Unreal Engine

Quill	Dear Angelica
Curve Geometry	Curve Geometry
No Lighting	No Lighting
OpenGL	DirectX 11
Layer UI	Scenes and Transitions
Fixed Shaders	Custom Materials
Static	Animated

VRDC VIRTUAL REALITY DEVELOPERS CONFERENCE | FEB 27-28, 2017 | EXPO: MAR 1-3, 2017 #VRDC17

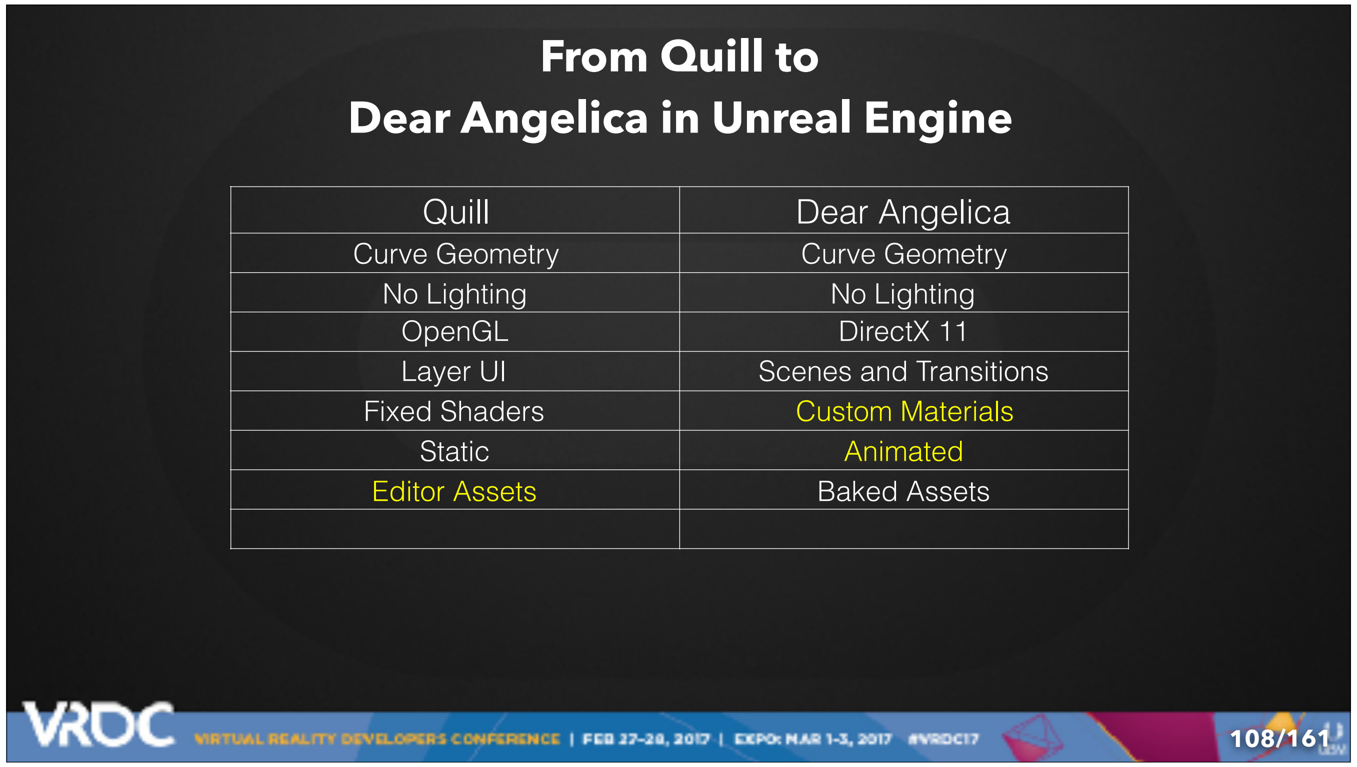
But in DA we'd be dealing with animated geometry, generally accomplished via the position textures coming out of Houdini, as Chris described earlier. So lots of large textures.



From Quill to Dear Angelica in Unreal Engine

Quill	Dear Angelica
Curve Geometry	Curve Geometry
No Lighting	No Lighting
OpenGL	DirectX 11
Layer UI	Scenes and Transitions
Fixed Shaders	Custom Materials
Static	Animated
Editor Assets	

One area we might hope to get some performance back would be in Quill's support for an editing paradigm

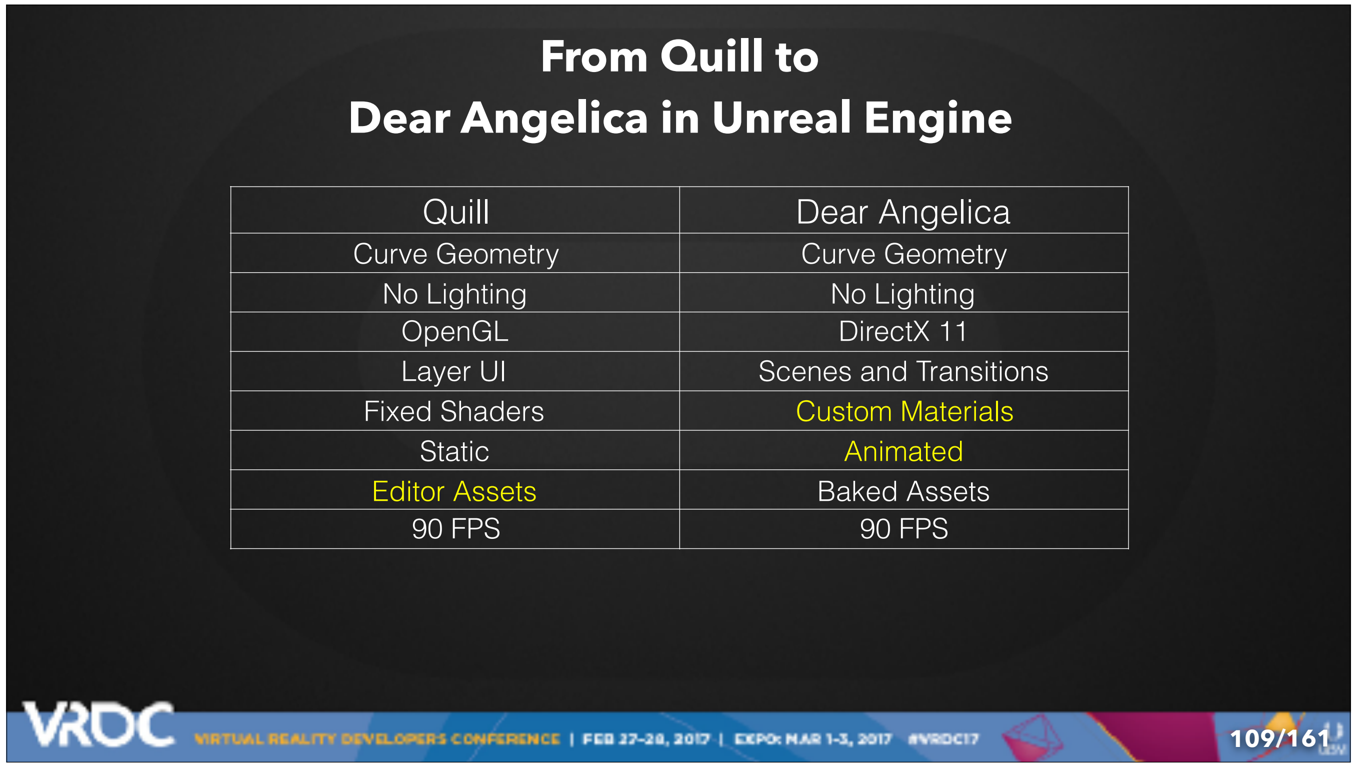


From Quill to Dear Angelica in Unreal Engine

Quill	Dear Angelica
Curve Geometry	Curve Geometry
No Lighting	No Lighting
OpenGL	DirectX 11
Layer UI	Scenes and Transitions
Fixed Shaders	Custom Materials
Static	Animated
Editor Assets	Baked Assets

VRDC VIRTUAL REALITY DEVELOPERS CONFERENCE | FEB 27-28, 2017 | EXPO: MAR 1-3, 2017 #VRDC17

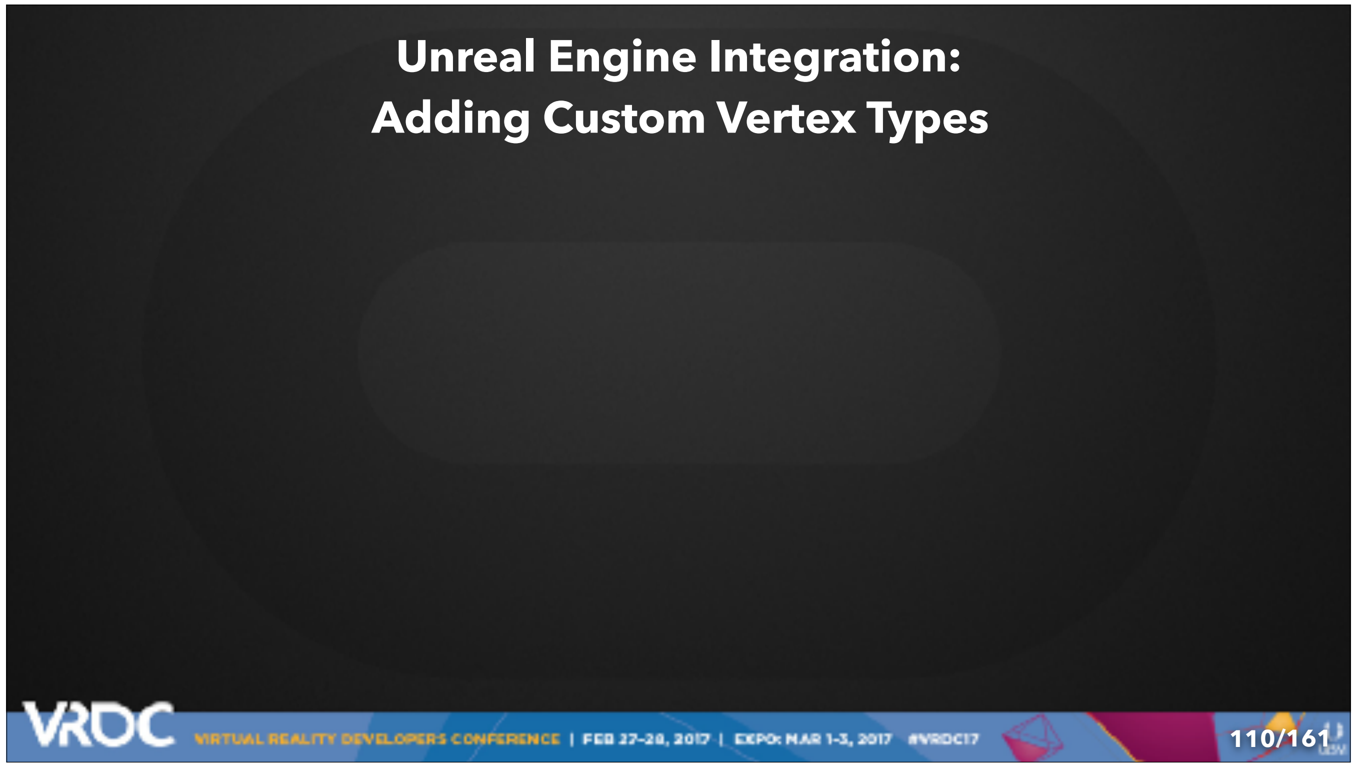
whereas we could do offline processing on our GPU layouts to get for example nice culling. Martin will talk more about this shortly.



From Quill to Dear Angelica in Unreal Engine

Quill	Dear Angelica
Curve Geometry	Curve Geometry
No Lighting	No Lighting
OpenGL	DirectX 11
Layer UI	Scenes and Transitions
Fixed Shaders	Custom Materials
Static	Animated
Editor Assets	Baked Assets
90 FPS	90 FPS

All in all though, one thing that was fairly surprising to me, is that by the end of production and before we went into finalling, we weren't that far off base in terms of performance. Especially considering these scenes were created on a more powerful machine than even rec spec. And I think a lot of this is owing to the fact that Wesley was creating this environment in the HMD at 90Hz, so scene complexity perhaps had a tendency to self-select.



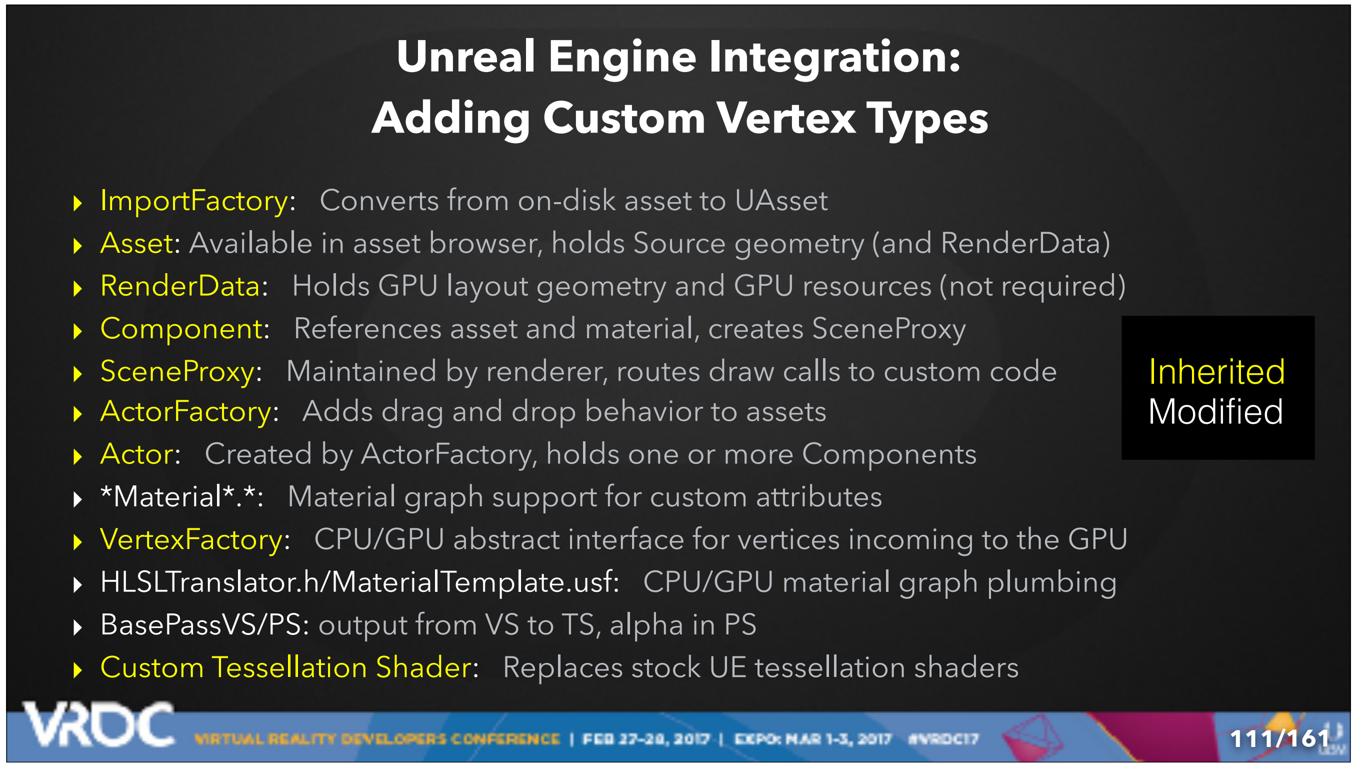
Unreal Engine Integration: Adding Custom Vertex Types

VRDC VIRTUAL REALITY DEVELOPERS CONFERENCE | FEB 27-28, 2017 | EXPO: MAR 1-3, 2017 #VRDC17

Ok so switching into Unreal Engine land now.

At the time of starting DA, it was hard to find good examples of adding custom geometry pipelines to Unreal. Stock Unreal of course supports a few different vertex types, but it supports many pathways and options for those types, making it tough to isolate what is essential for just getting something up and running.

So, the main purpose of this next slide is to itemize in 1 place all the files and classes you would need to add or modify to support custom geometry types in Unreal.



Unreal Engine Integration: Adding Custom Vertex Types

- ImportFactory: Converts from on-disk asset to UAsset
- Asset: Available in asset browser, holds Source geometry (and RenderData)
- RenderData: Holds GPU layout geometry and GPU resources (not required)
- Component: References asset and material, creates SceneProxy
- SceneProxy: Maintained by renderer, routes draw calls to custom code
- ActorFactory: Adds drag and drop behavior to assets
- Actor: Created by ActorFactory, holds one or more Components
- *Material*.*: Material graph support for custom attributes
- VertexFactory: CPU/GPU abstract interface for vertices incoming to the GPU
- HLSLTranslator.h/MaterialTemplate.usf: CPU/GPU material graph plumbing
- BasePassVS/PS: output from VS to TS, alpha in PS
- Custom Tessellation Shader: Replaces stock UE tessellation shaders

Inherited
Modified

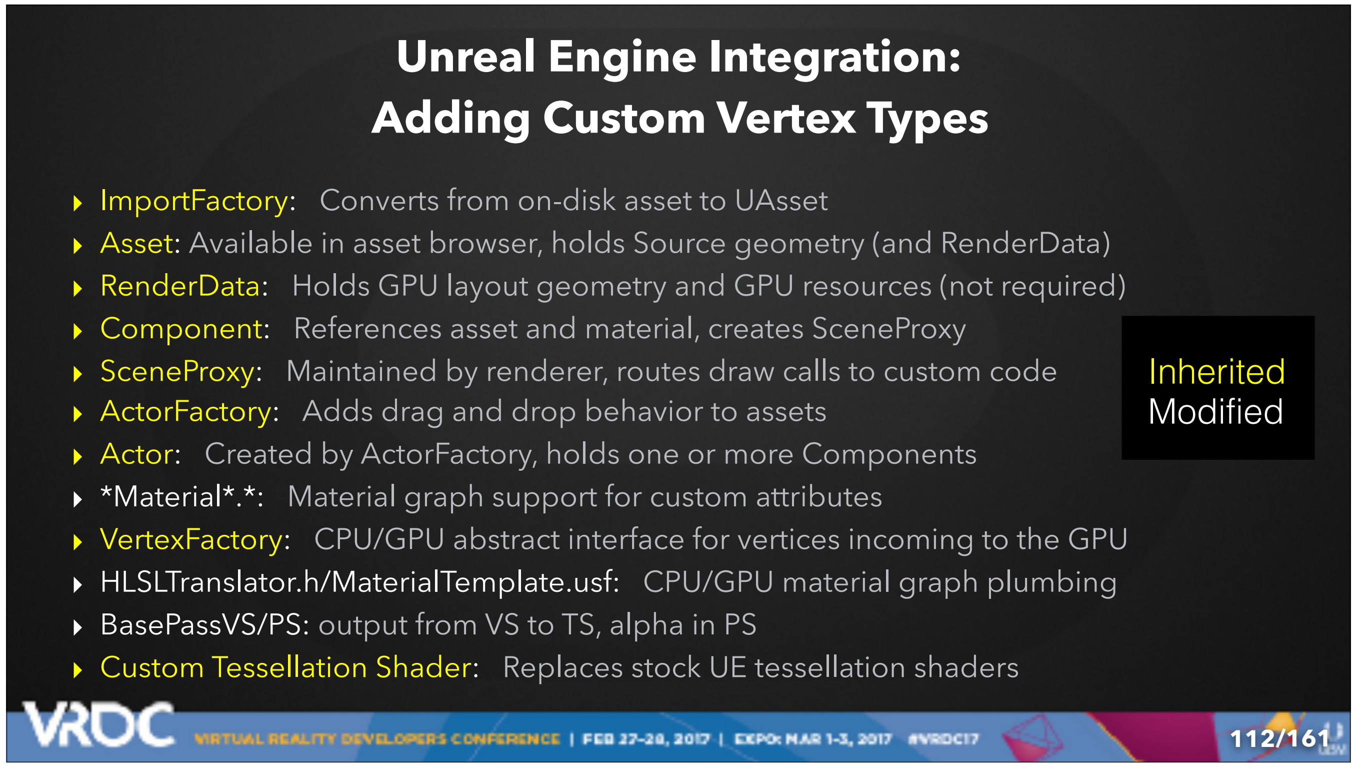
VRDC VIRTUAL REALITY DEVELOPERS CONFERENCE | FEB 27-28, 2017 | EXPO: MAR 1-3, 2017 #VRDC17

This slide is just a reference slide, I'm not going into line by line detail here.

…

The organization here, from top to bottom, we're moving from files on disk to pixels on screen.
The lines in yellow indicate that we implemented or extended an interface, so these were systems which were intended to be extended.
The lines in white indicate direct modifications to engine code. For the most part these changes were in the material system.

Don't worry about the details listed on this slide, it's just for reference.



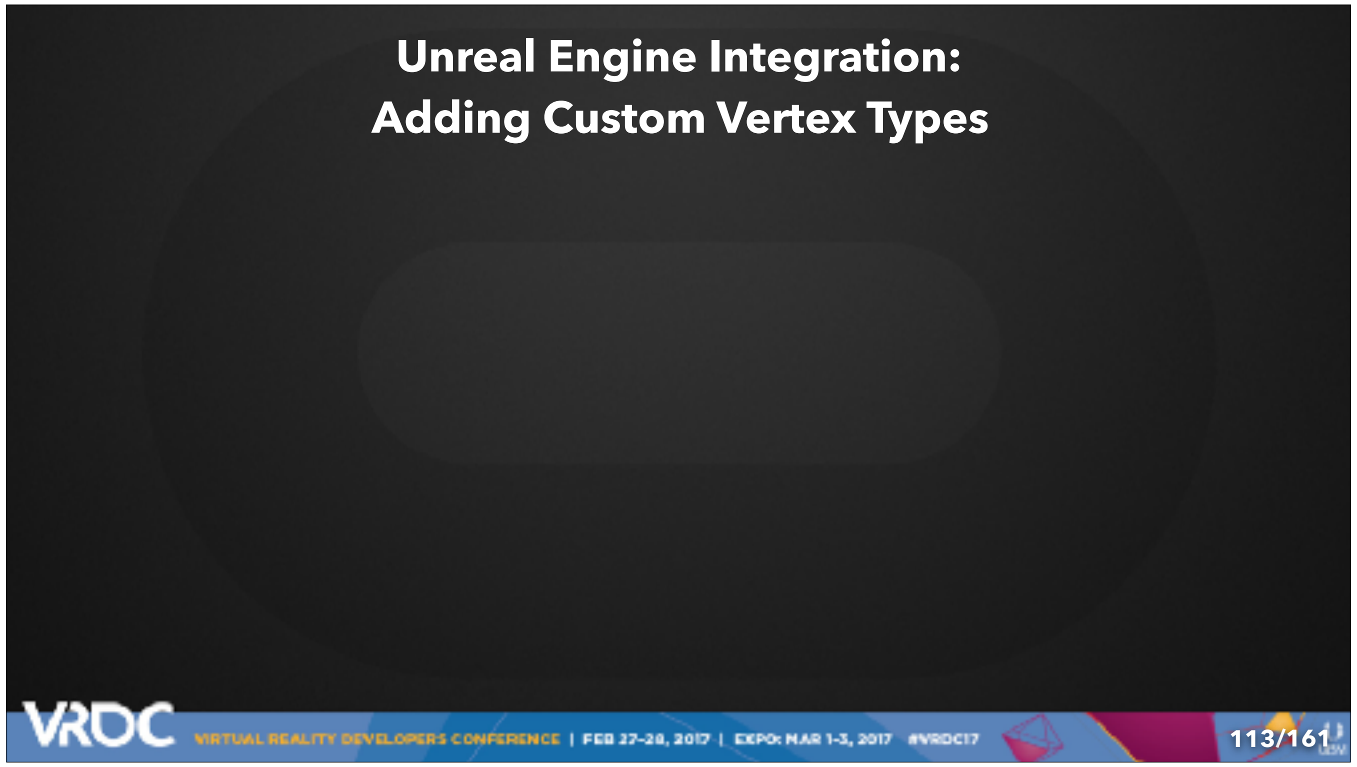
Unreal Engine Integration: Adding Custom Vertex Types

- ImportFactory: Converts from on-disk asset to UAsset
- Asset: Available in asset browser, holds Source geometry (and RenderData)
- RenderData: Holds GPU layout geometry and GPU resources (not required)
- Component: References asset and material, creates SceneProxy
- SceneProxy: Maintained by renderer, routes draw calls to custom code
- ActorFactory: Adds drag and drop behavior to assets
- Actor: Created by ActorFactory, holds one or more Components
- *Material*.*: Material graph support for custom attributes
- VertexFactory: CPU/GPU abstract interface for vertices incoming to the GPU
- HLSLTranslator.h/MaterialTemplate.usf: CPU/GPU material graph plumbing
- BasePassVS/PS: output from VS to TS, alpha in PS
- Custom Tessellation Shader: Replaces stock UE tessellation shaders

Inherited
Modified

VRDC VIRTUAL REALITY DEVELOPERS CONFERENCE | FEB 27-28, 2017 | EXPO: MAR 1-3, 2017 #VRDC17

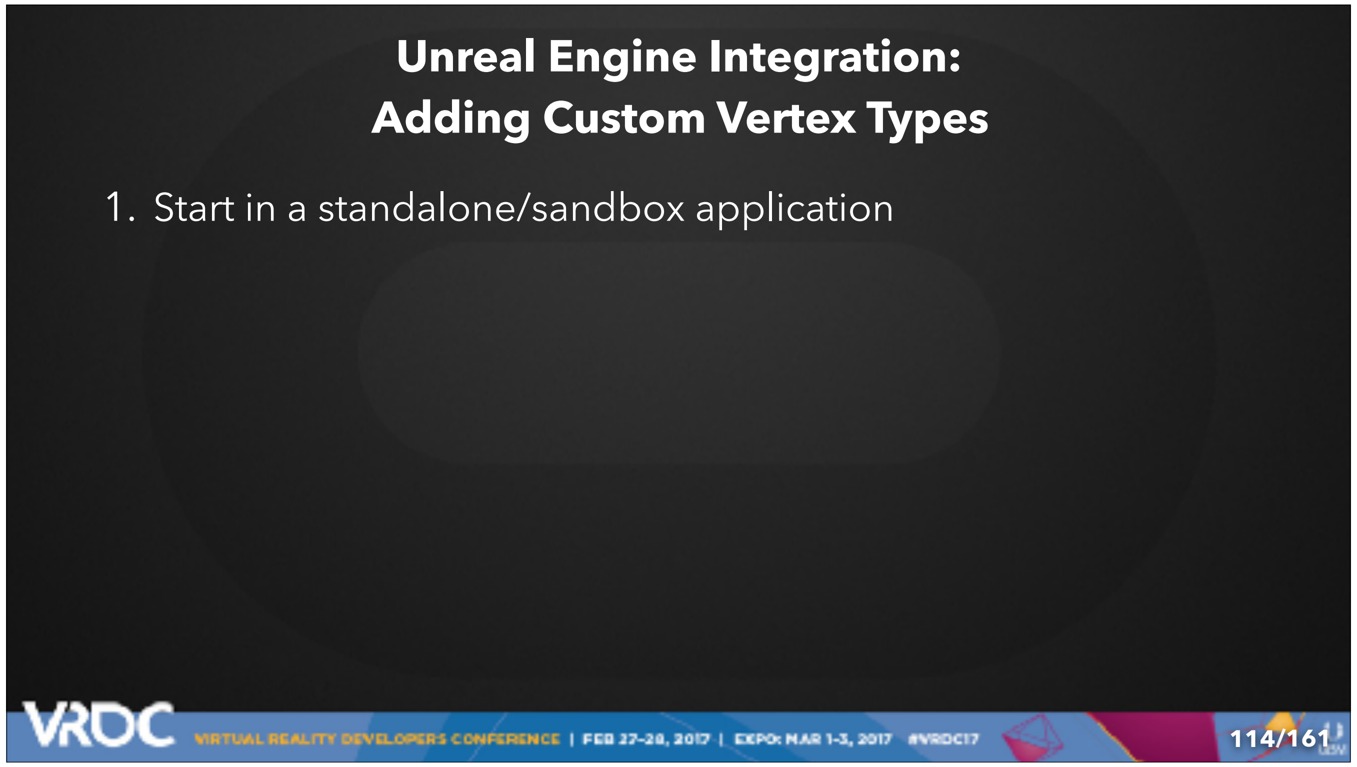
This slide is also a high level documentation for the integration which we're working on making available in github. We continue to use the integration internally, so all the work is done, we just need to finalize a few details. More info on that at the end.



Unreal Engine Integration: Adding Custom Vertex Types

VIRTUAL REALITY DEVELOPERS CONFERENCE | FEB 27-28, 2017 | EXPO: MAR 1-3, 2017 #VRDC17

Before moving on to the final section, I wanted to outline a few things we did (or didn't do) during the integration which might be helpful:



Unreal Engine Integration: Adding Custom Vertex Types

1. Start in a standalone/sandbox application

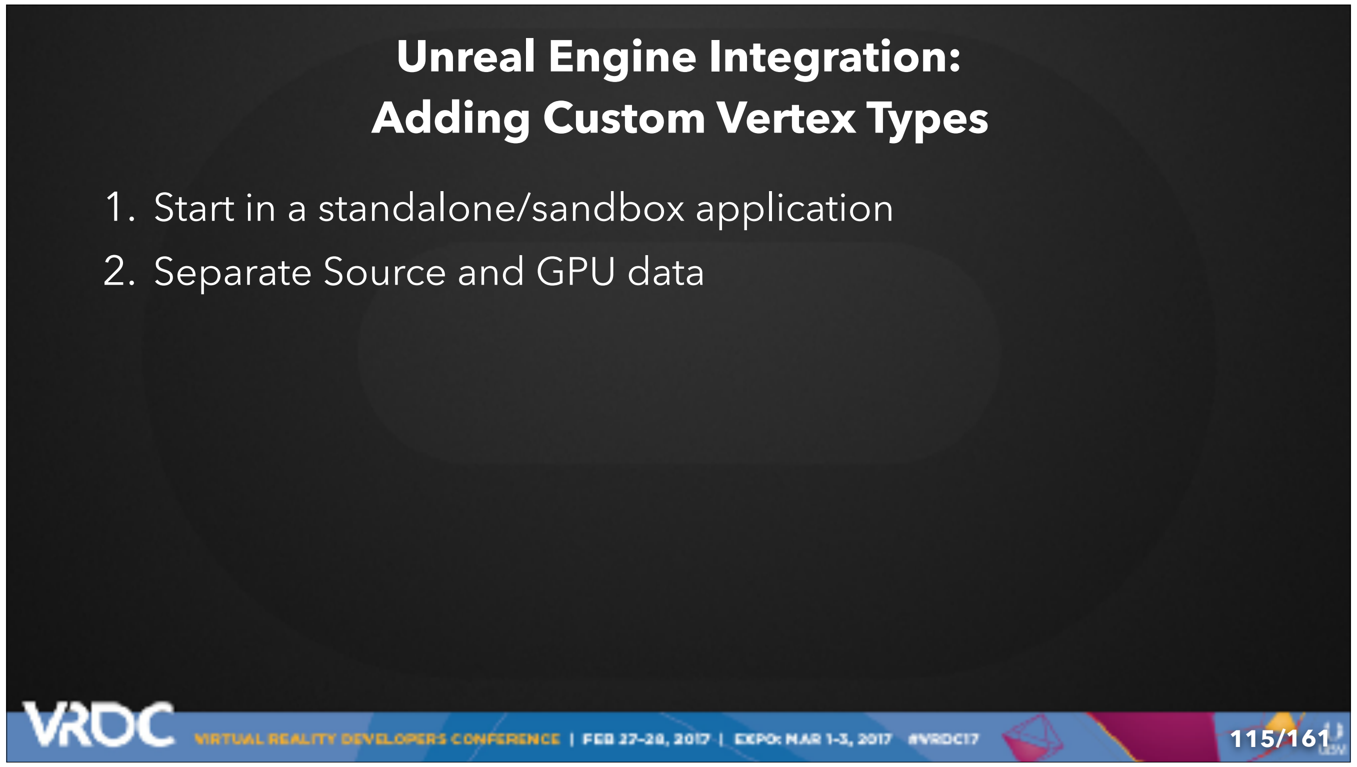
VRDC VIRTUAL REALITY DEVELOPERS CONFERENCE | FEB 27-28, 2017 | EXPO: MAR 1-3, 2017 #VRDC17

One thing I wish we had done right from the start was set up a standalone application for quick iteration. We did have a working reference implementation in the Quill app itself, but it was in OpenGL and was managing data a little differently on account of the very detailed editing paradigm.

If we had spent the time to get a DirectX implementation working in a tiny application, and then just dropped that into UE, we could have isolated technique problems from plumbing problems.

Especially if you're modifying MaterialTemplate.usf, VertexFactory.usf, or the BasePass shaders, it's helpful to minimize the attempts there.

And especially with VR, understanding how vertex size and GPU pipeline and stereo rendering techniques intersect, it's handy to be able to iterate on those combinations quickly. Of course when you bring those techniques back into UE you may still get different results, but it will help you isolate the differences.

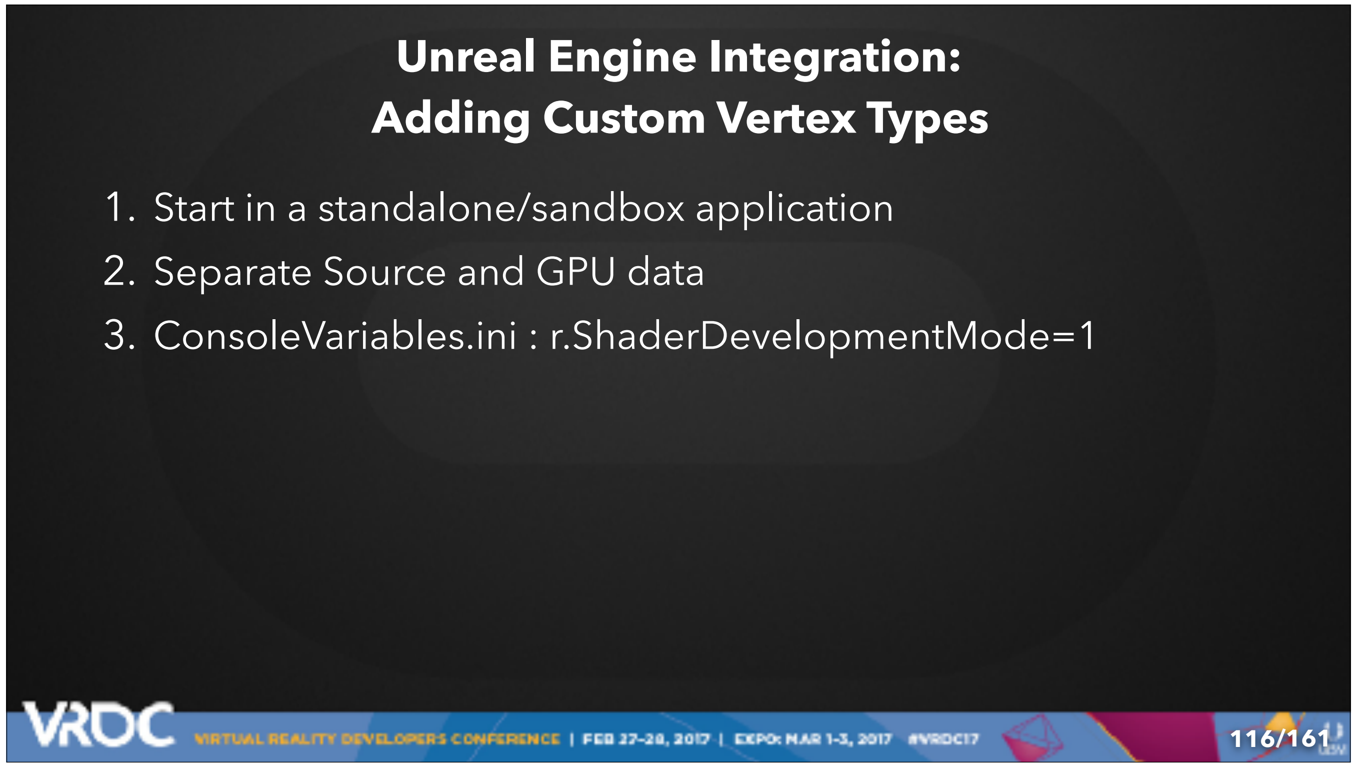


Unreal Engine Integration: Adding Custom Vertex Types

1. Start in a standalone/sandbox application
2. Separate Source and GPU data

VRDC VIRTUAL REALITY DEVELOPERS CONFERENCE | FEB 27-28, 2017 | EXPO: MAR 1-3, 2017 #VRDC17

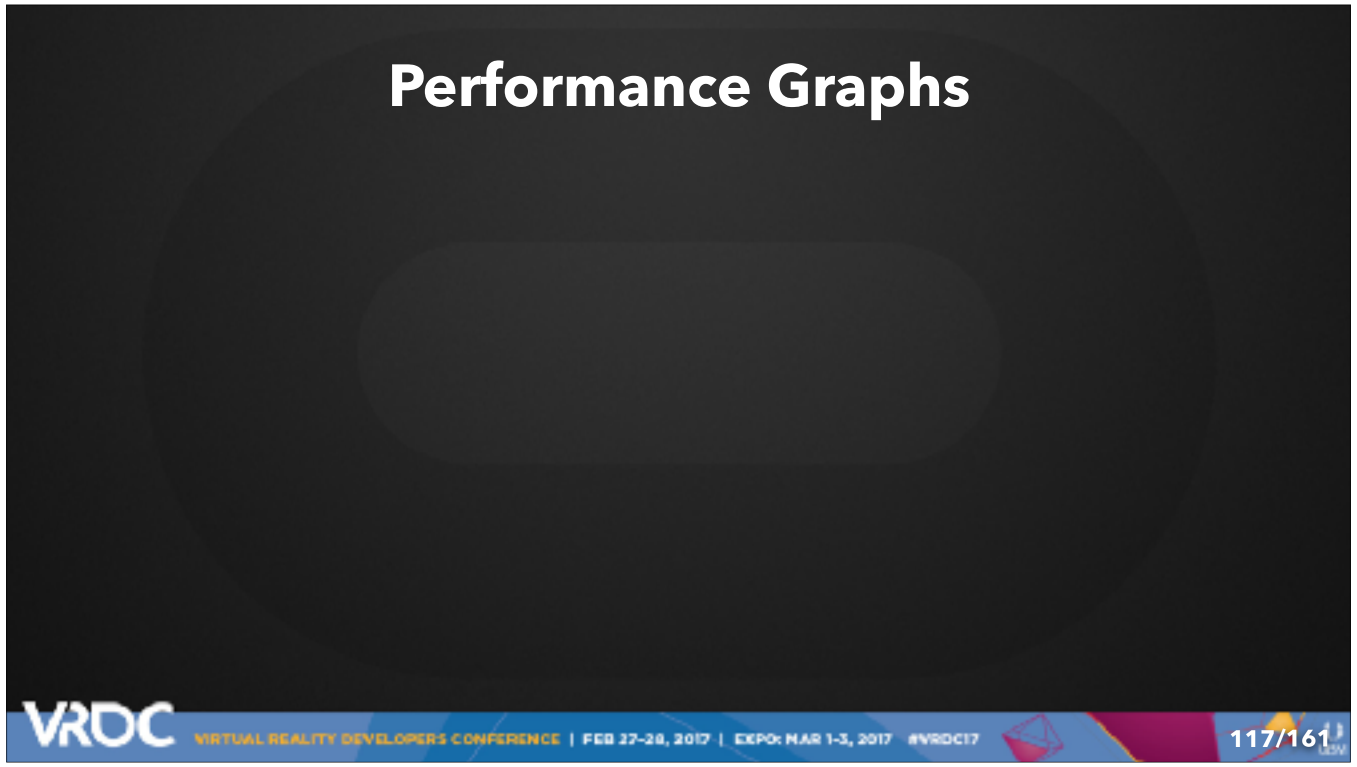
Separate source and GPU data. This is a pattern implemented in StaticMesh, which keeps the source vertices around, which we adopted as well. Again, as we were experimenting with different shader stages and vertex sizes, having a different representation for a source vertex and a gpu vertex built in was super helpful. We were able to switch from geometry shaders to tessellation shaders to hardware instancing and back in a pretty safe manner because we had a clear separation of our data structures here.



Unreal Engine Integration: Adding Custom Vertex Types

1. Start in a standalone/sandbox application
2. Separate Source and GPU data
3. ConsoleVariables.ini : r.ShaderDevelopmentMode=1

And lastly, ConsoleVariables.ini shader debugging mode is incredibly helpful to make sure the right values are getting connected to the correct spots. This option is probably common knowledge, but imagine trying to diagnose material problems BEFORE knowing about it, and then finding it, and being so grateful. That's why it's #3.



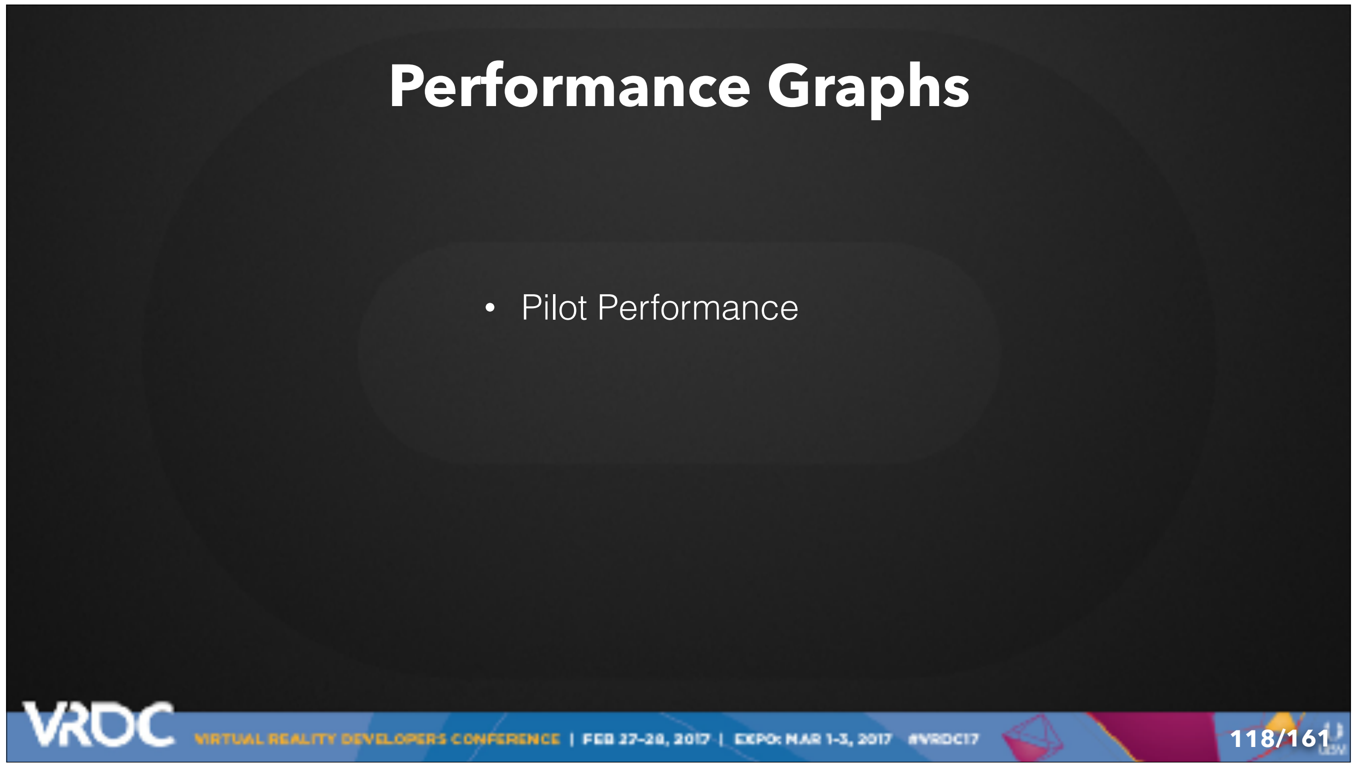
Performance Graphs

VRDC VIRTUAL REALITY DEVELOPERS CONFERENCE | FEB 27-28, 2017 | EXPO: MAR 1-3, 2017 #VRDC17

117/161

Dear Angelica was an ambitious project on a short schedule, and when you add to that a new geometry type that needs to draw at 90FPS or better, you want to be confident about shipping.

We put together a simple system that really gave us some peace of mind in identifying issues clearly and quickly. The key parts were:

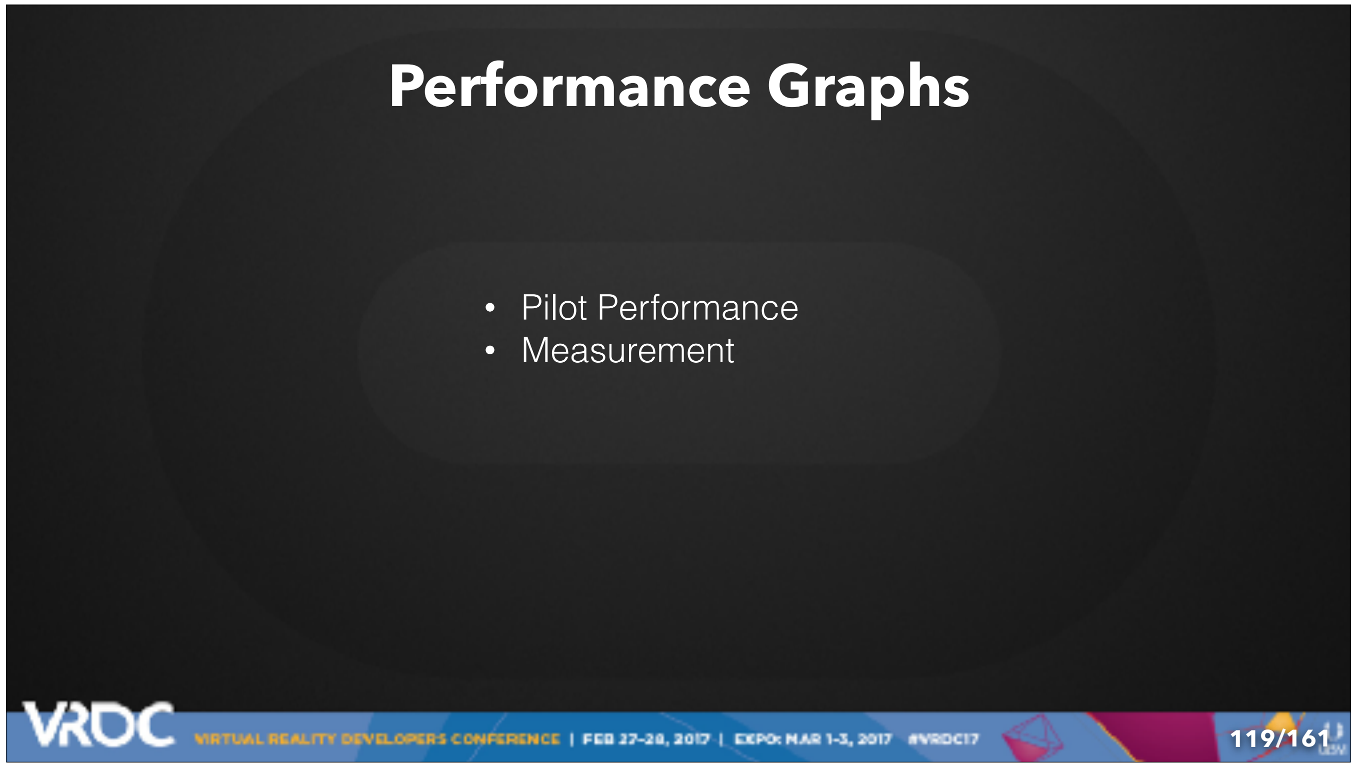


Performance Graphs

- Pilot Performance

VIRTUAL REALITY DEVELOPERS CONFERENCE | FEB 27-28, 2017 | EXPO: MAR 1-3, 2017 #VRDC17

Pilot performance, correct simulation of an actual person in the headset.

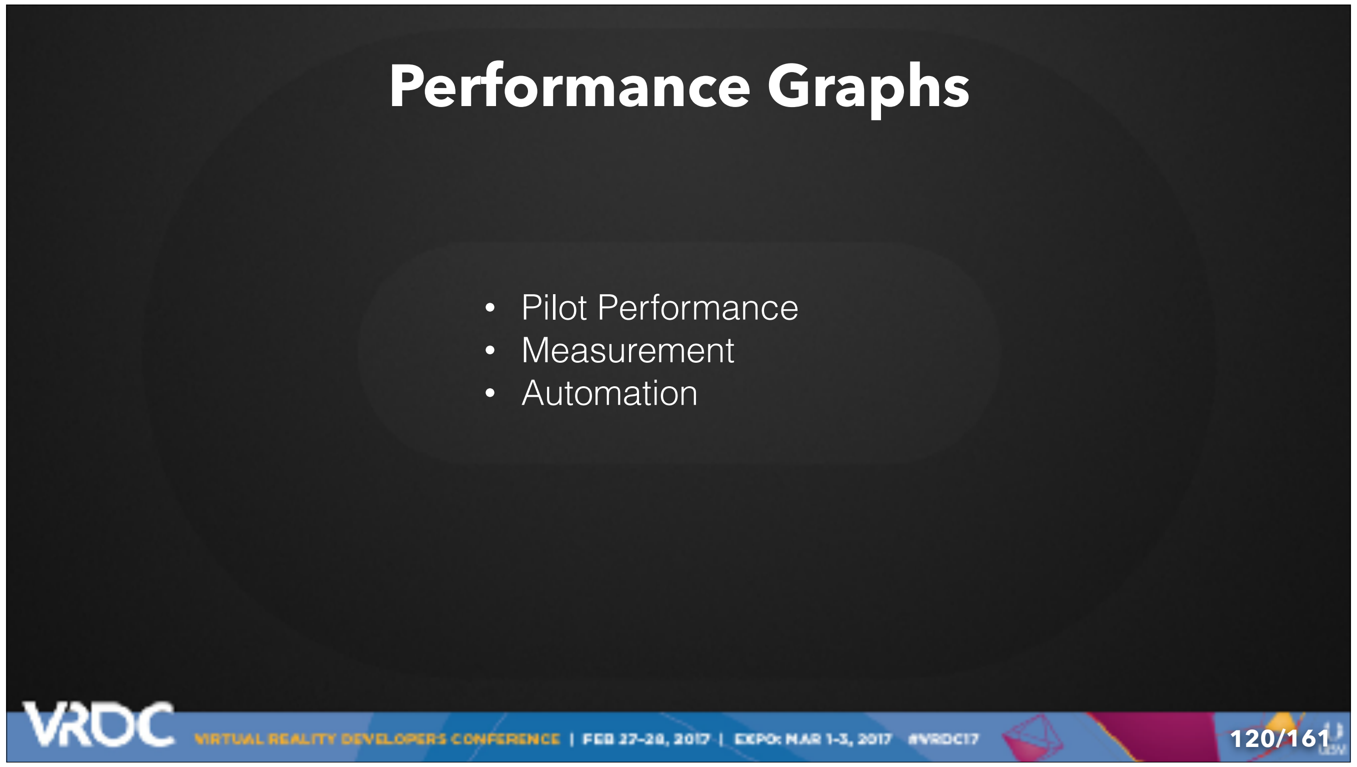


Performance Graphs

- Pilot Performance
- Measurement

VRDC VIRTUAL REALITY DEVELOPERS CONFERENCE | FEB 27-28, 2017 | EXPO: MAR 1-3, 2017 #VRDC17

Measuring not just FPS but actionable values.



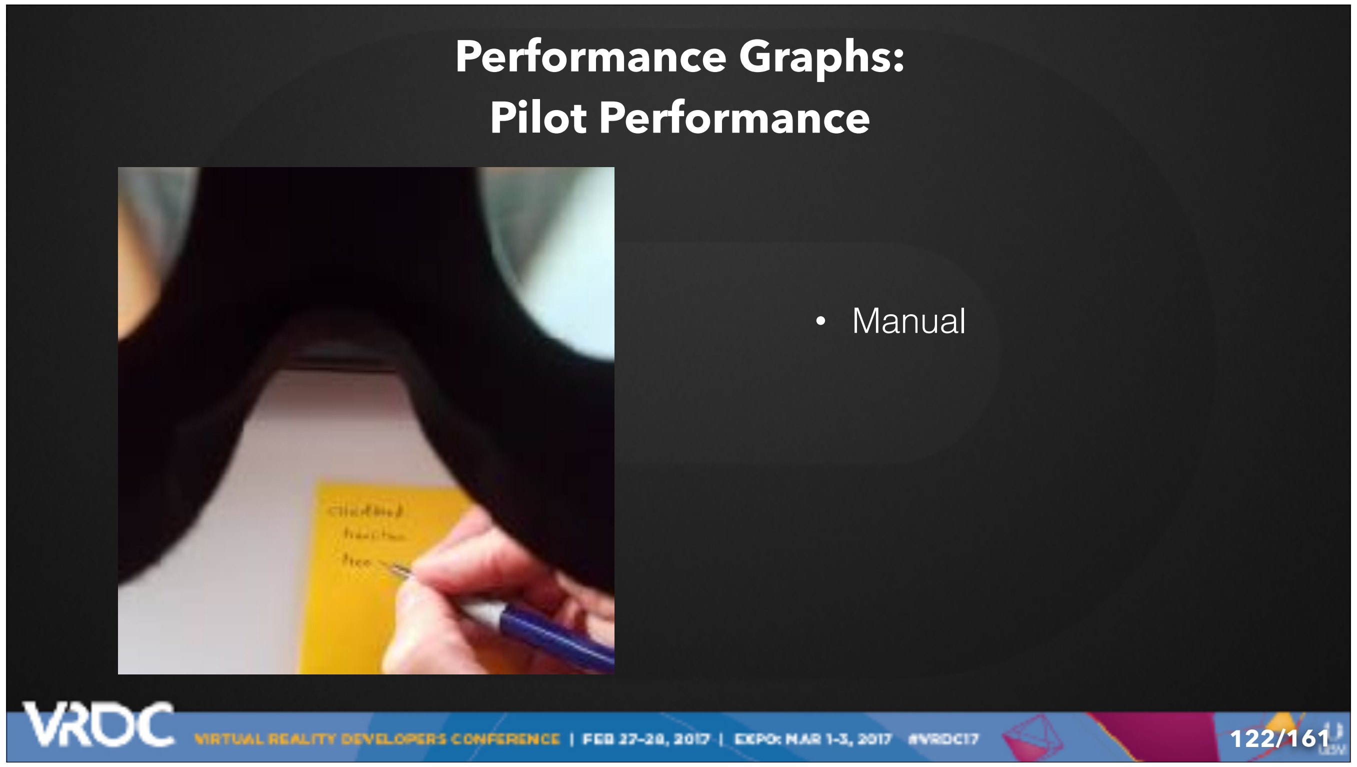
Performance Graphs

- Pilot Performance
- Measurement
- Automation

And being able to do the above very easily.

Performance Graphs: Pilot Performance

When we first had Quill assets drawing in Unreal,

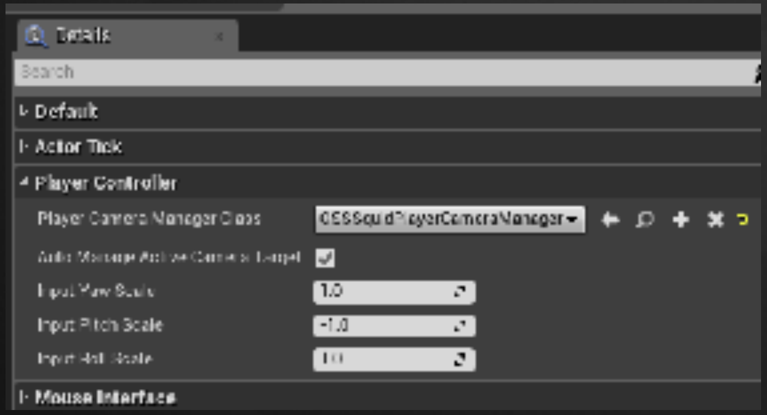


Performance Graphs: Pilot Performance

- Manual

We just tested everything manually. Play through the whole thing, try to look for problem areas, peek out of the HMD gills and make a note. But also super error prone. I would think I had isolated a problem spot, but when I would go back over it again the characteristics are very different, because I had my seat a few feet further back.

Performance Graphs: Pilot Performance



Details

Search

- Default
- Actor Tick
- Player Controller

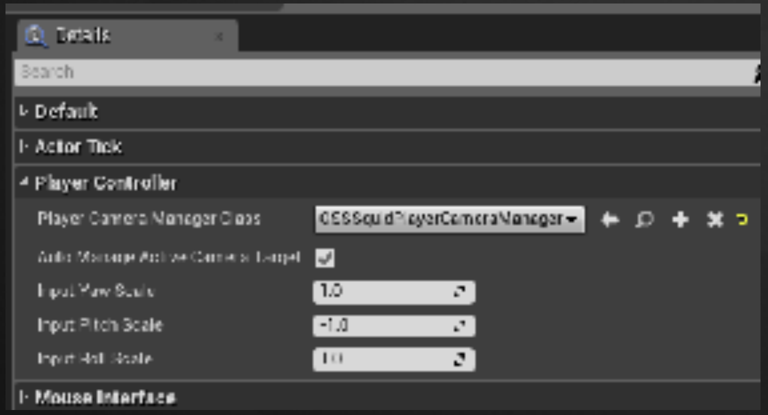
Player Camera Manager Class	OSSSquidPlayerCameraManager
Auto Manage Active Camera Target	✓
Input Yaw Scale	1.0
Input Pitch Scale	-1.0
Input Roll Scale	1.0

- Mouse Interface

- Manual
- Record/Play HMD Transforms

Enter SQUID. Max Planck wrote a camera manager replacement for Unreal, called SQUID, which can record and play back the transform of the HMD.

Performance Graphs:
Pilot Performance



Details

Search

Default

Actor Tick

Player Controller

Player Camera Manager Class: 0SSSquidPlayerCameraManager

Auto Manage Active Camera Target: ✓

Input Yaw Scale: 1.0

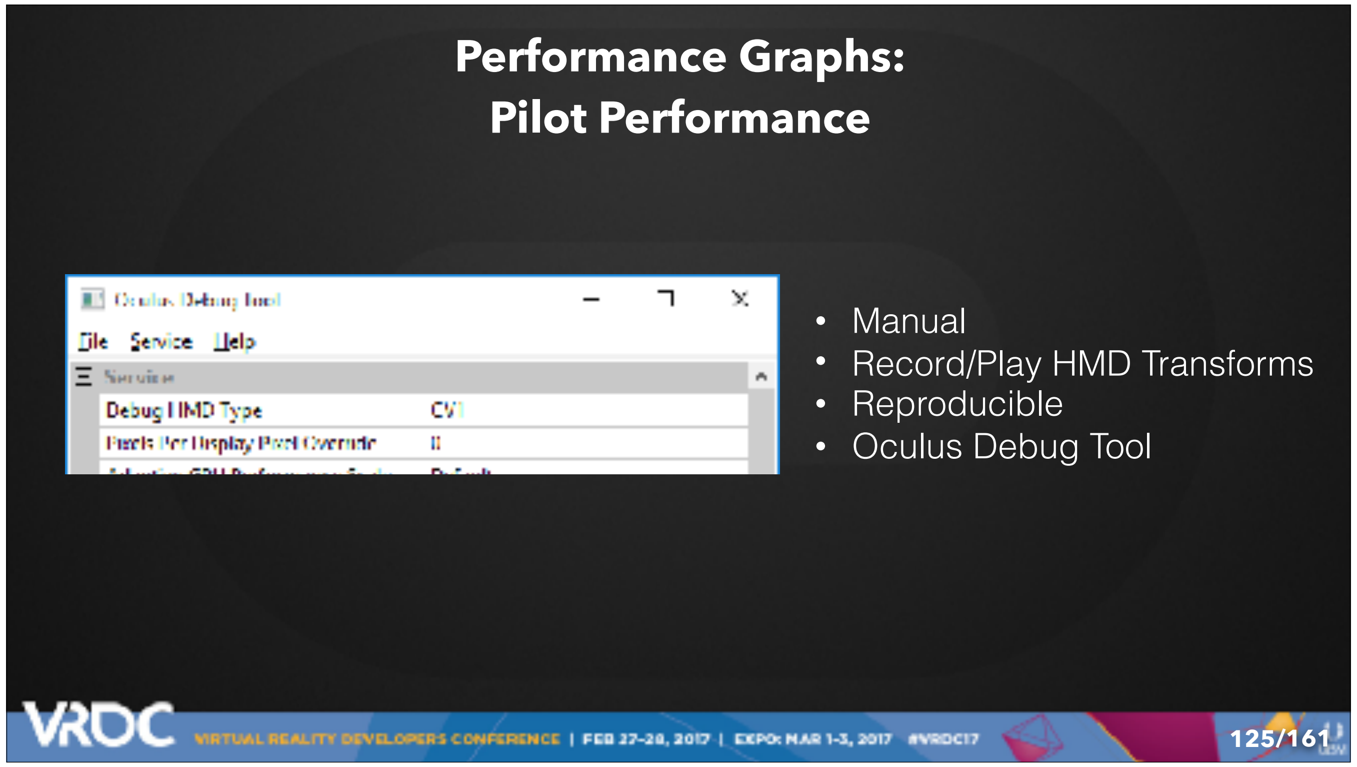
Input Pitch Scale: -1.0

Input Roll Scale: 1.0

Mouse Interface

- Manual
- Record/Play HMD Transforms
- Reproducible

This was a simple concept, but actually huge. Since DA was essentially a linear experience, we could have very close to identical playthroughs of the entire piece, which was about 12 minutes long.



Performance Graphs: Pilot Performance

Oculus Debug Tool

File Service Help

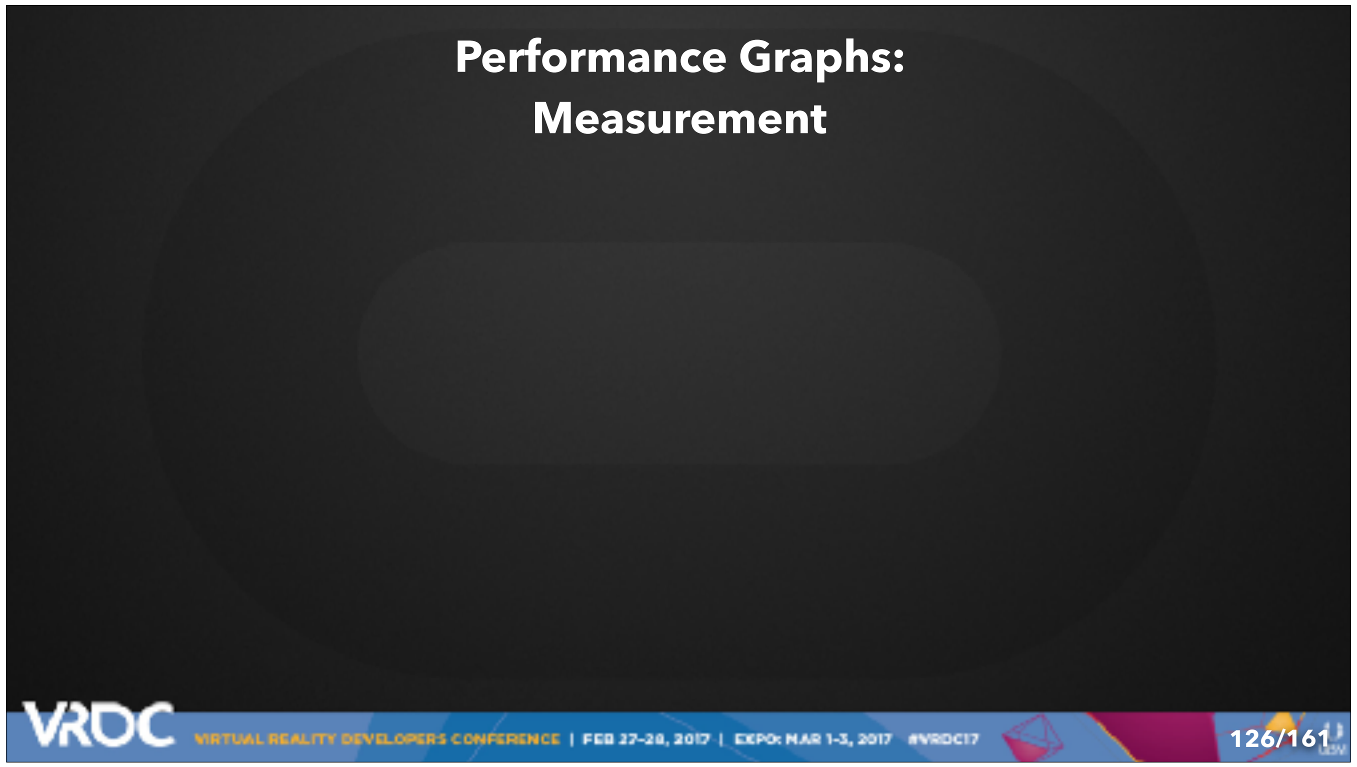
Service

Debug HMD Type	CV1
Pixels Per Display Pixel Override	0

- Manual
- Record/Play HMD Transforms
- Reproducible
- Oculus Debug Tool

VRDC VIRTUAL REALITY DEVELOPERS CONFERENCE | FEB 27-28, 2017 | EXPO: MAR 1-3, 2017 #VRDC17

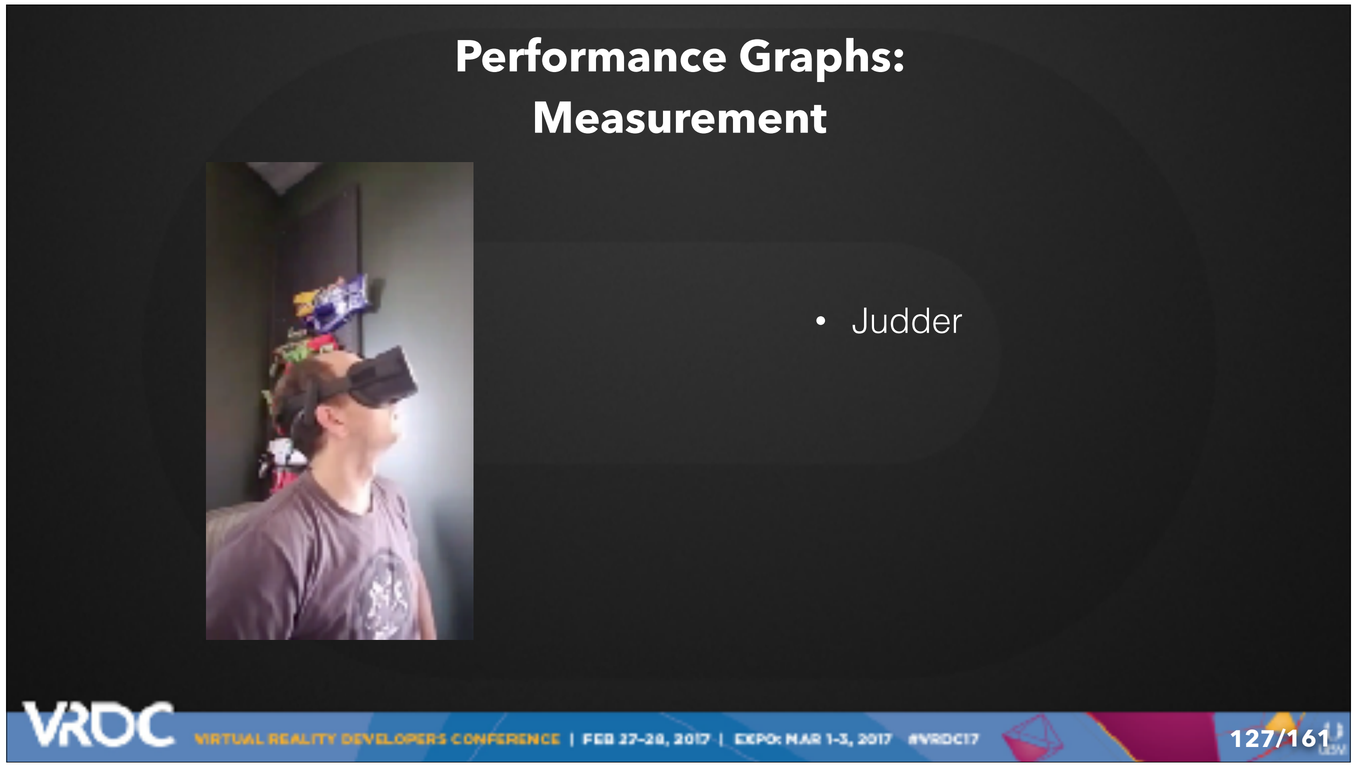
To round out the playback, we used the HMD emulation feature of the Oculus Debug Tool. We could hook up playback to machines without hardware, and we also would bypass the HMD timeout.



Performance Graphs: Measurement

VIRTUAL REALITY DEVELOPERS CONFERENCE | FEB 27-28, 2017 | EXPO: MAR 1-3, 2017 #VRDC17

Correct measurement of judder actually proved tricky for us.

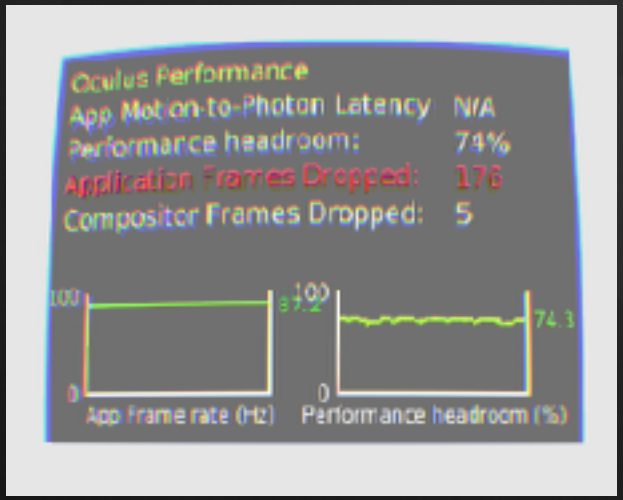


Performance Graphs: Measurement

- Judder

Very early on, the testing technique was to watch the entire experience shaking my head back and forth, looking for the telltale choppiness. That was not sustainable.

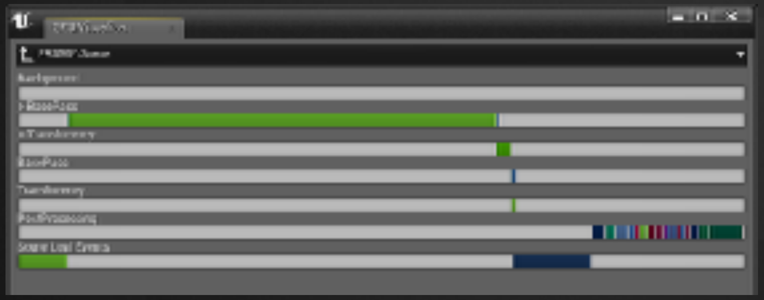
Performance Graphs: Measurement


Oculus Performance
App Motion-to-Photon Latency N/A
Performance headroom: 74%
Application Frames Dropped: 178
Compositor Frames Dropped: 5

- Judder
- Oculus Debug HUD

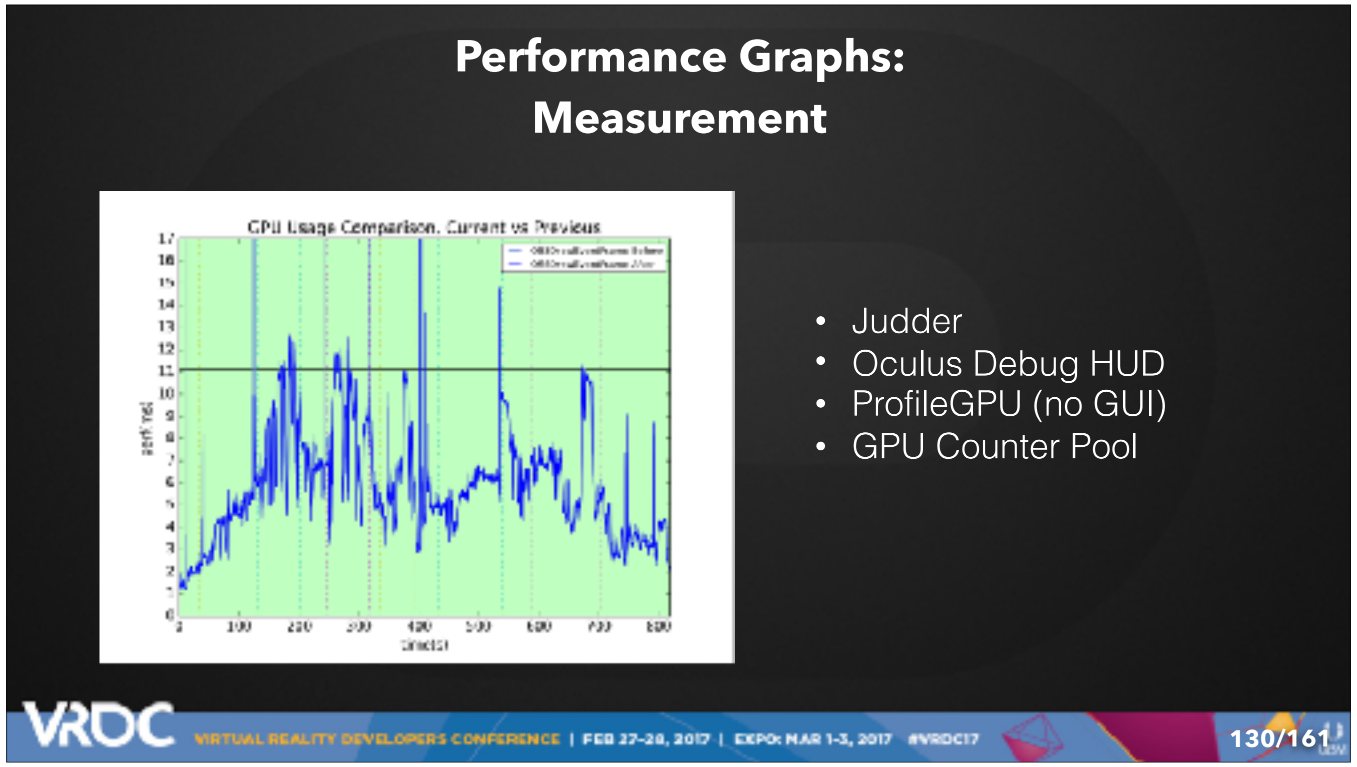
Fortunately Chris told me about the Oculus Debug HUD, and that number of dropped frames would be an indication of judder. But the issue with this, at the time, was these specific stats were not available to query through the SDK. And a plot of Unreal's stat.unit was not showing judder spikes where the Debug HUD was showing us going into the red.

Performance Graphs: Measurement

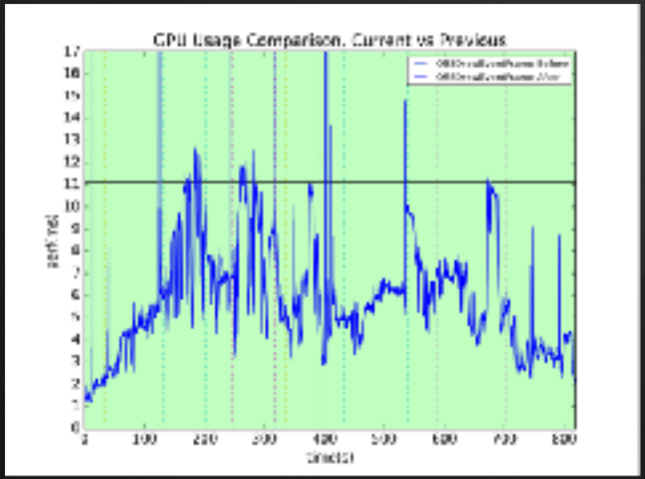


- Judder
- Oculus Debug HUD
- ProfileGPU (no GUI)

So instead we hooked up Unreal's ProfileGPU (without the GUI), and hacked in the ability to retrieve the couple values which were correlated with judder. The problem here was ProfileGPU introduces long GPU tasks which were forcing us to profile at very low frequency.



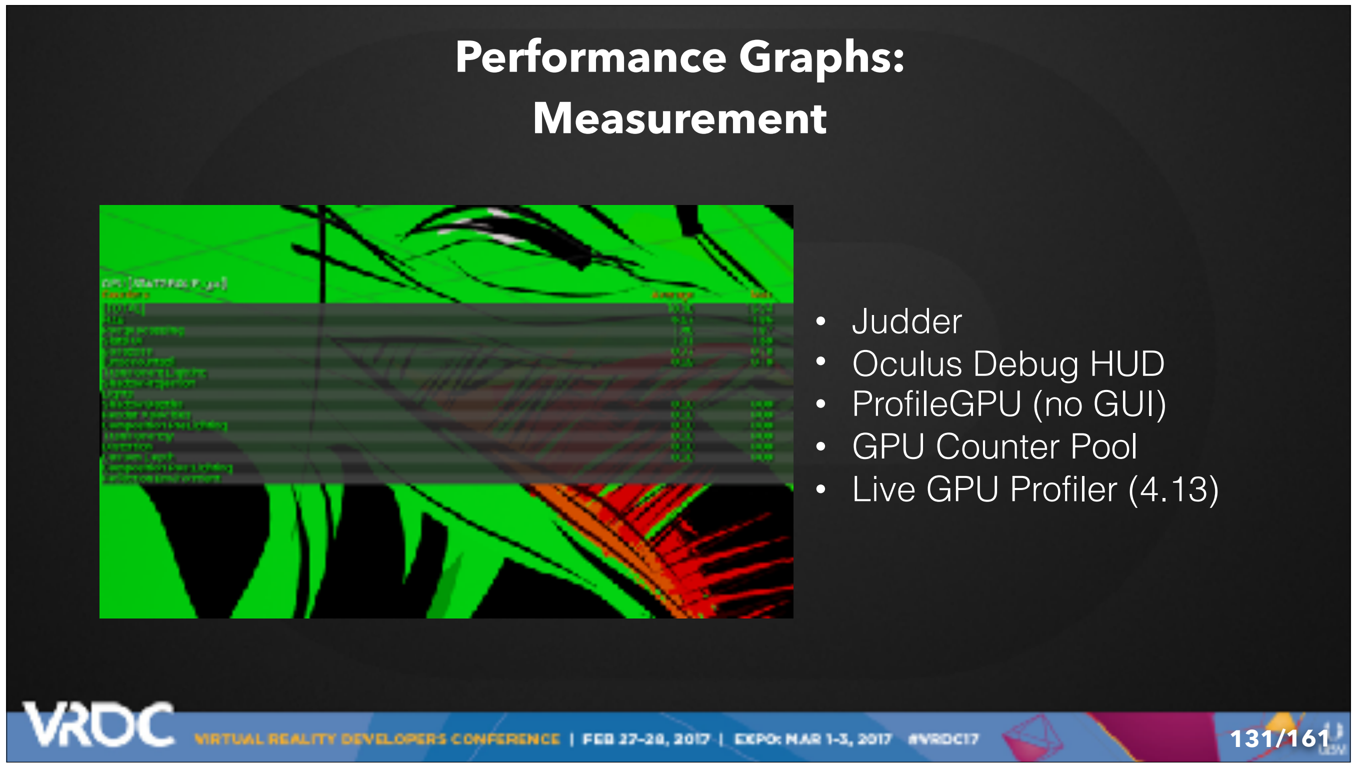
Performance Graphs: Measurement



- Judder
- Oculus Debug HUD
- ProfileGPU (no GUI)
- GPU Counter Pool

VRDC VIRTUAL REALITY DEVELOPERS CONFERENCE | FEB 27-28, 2017 | EXPO: MAR 1-3, 2017 #VRDC17

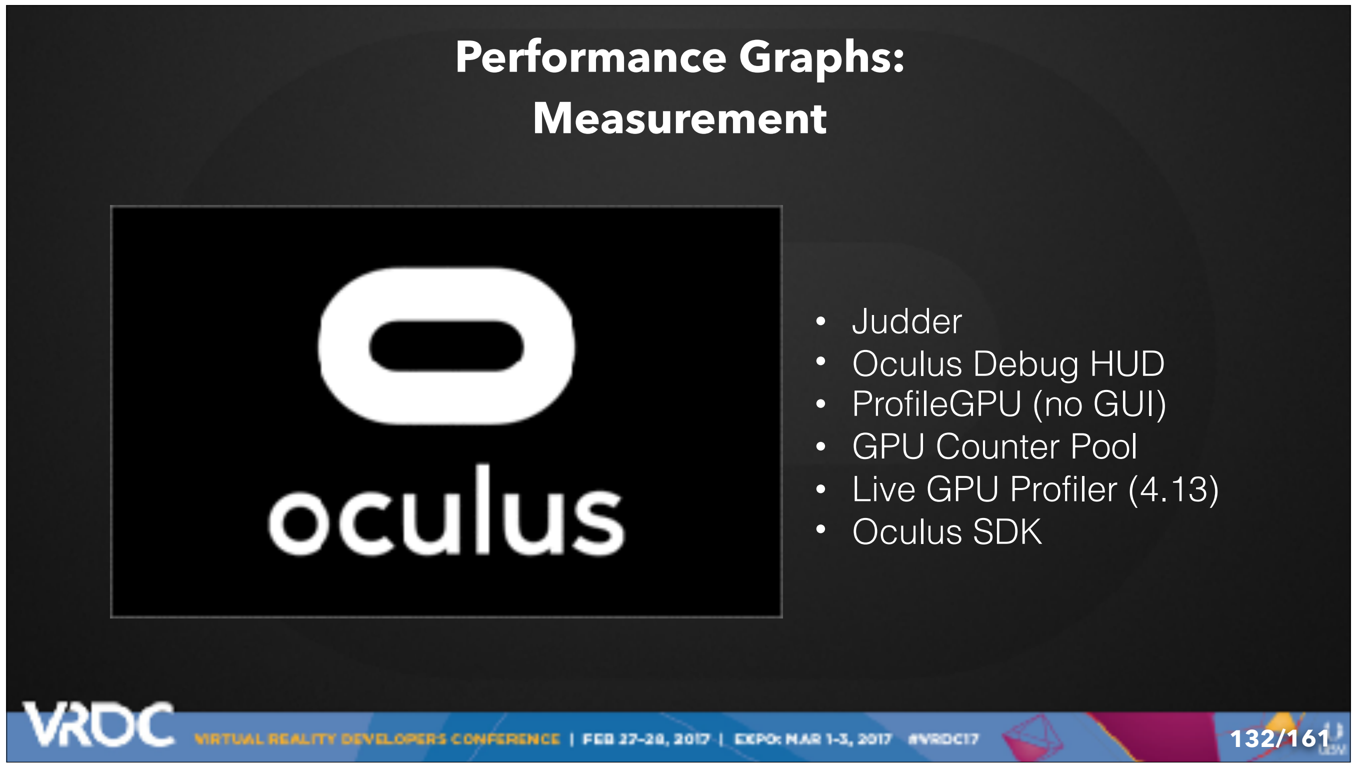
We added a pool of lightweight GPU counters and changed the polling to every frame, and finally were able to produce graphs like this one.



Performance Graphs: Measurement

- Judder
- Oculus Debug HUD
- ProfileGPU (no GUI)
- GPU Counter Pool
- Live GPU Profiler (4.13)

As an aside, if we were to do this again today, we’d probably look at the Live GPU Profiler first which was added in UE4.13.

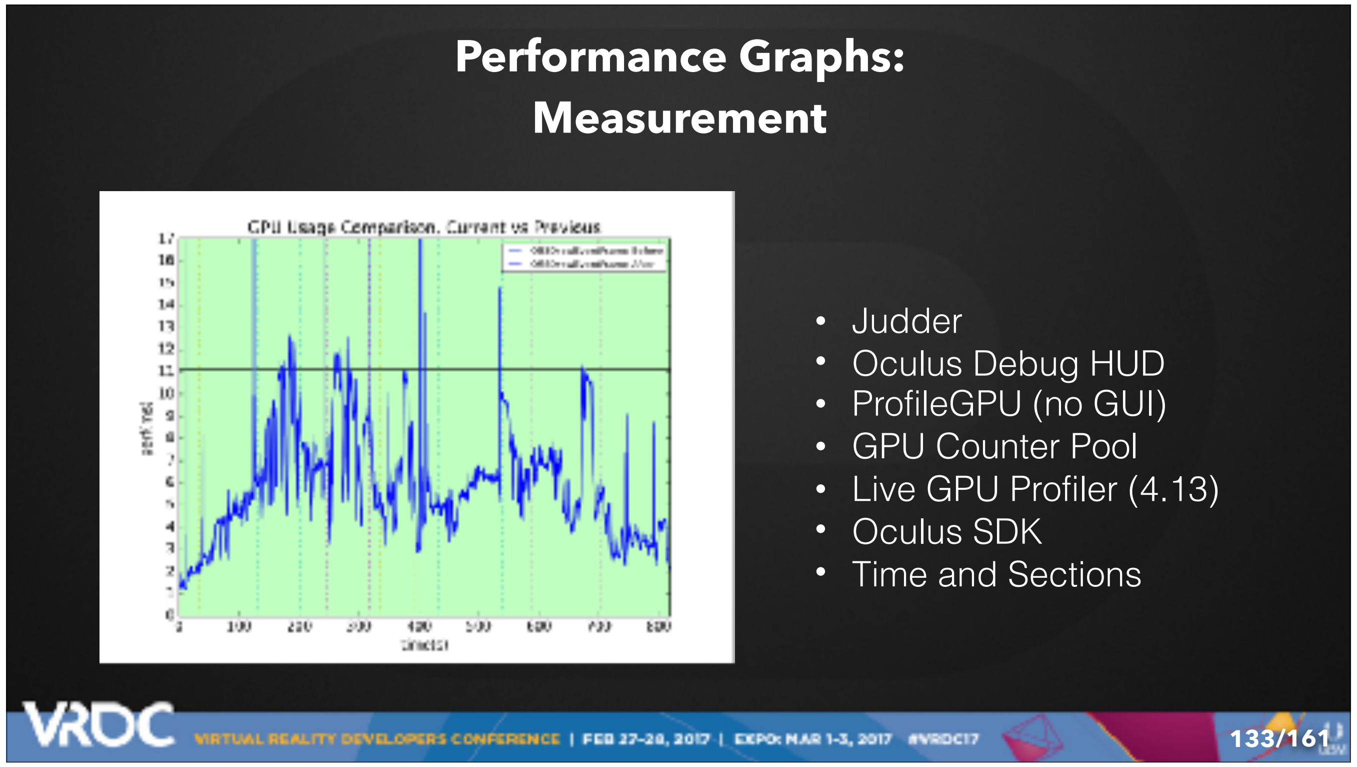


Performance Graphs: Measurement

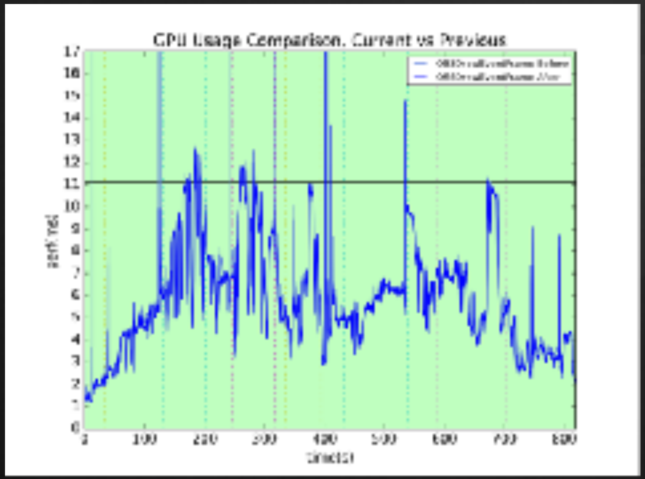
- Judder
- Oculus Debug HUD
- ProfileGPU (no GUI)
- GPU Counter Pool
- Live GPU Profiler (4.13)
- Oculus SDK

VRDC VIRTUAL REALITY DEVELOPERS CONFERENCE | FEB 27-28, 2017 | EXPO: MAR 1-3, 2017 #VRDC17

Or better yet, the Oculus SDK now supports querying the exact performance stats we wanted.

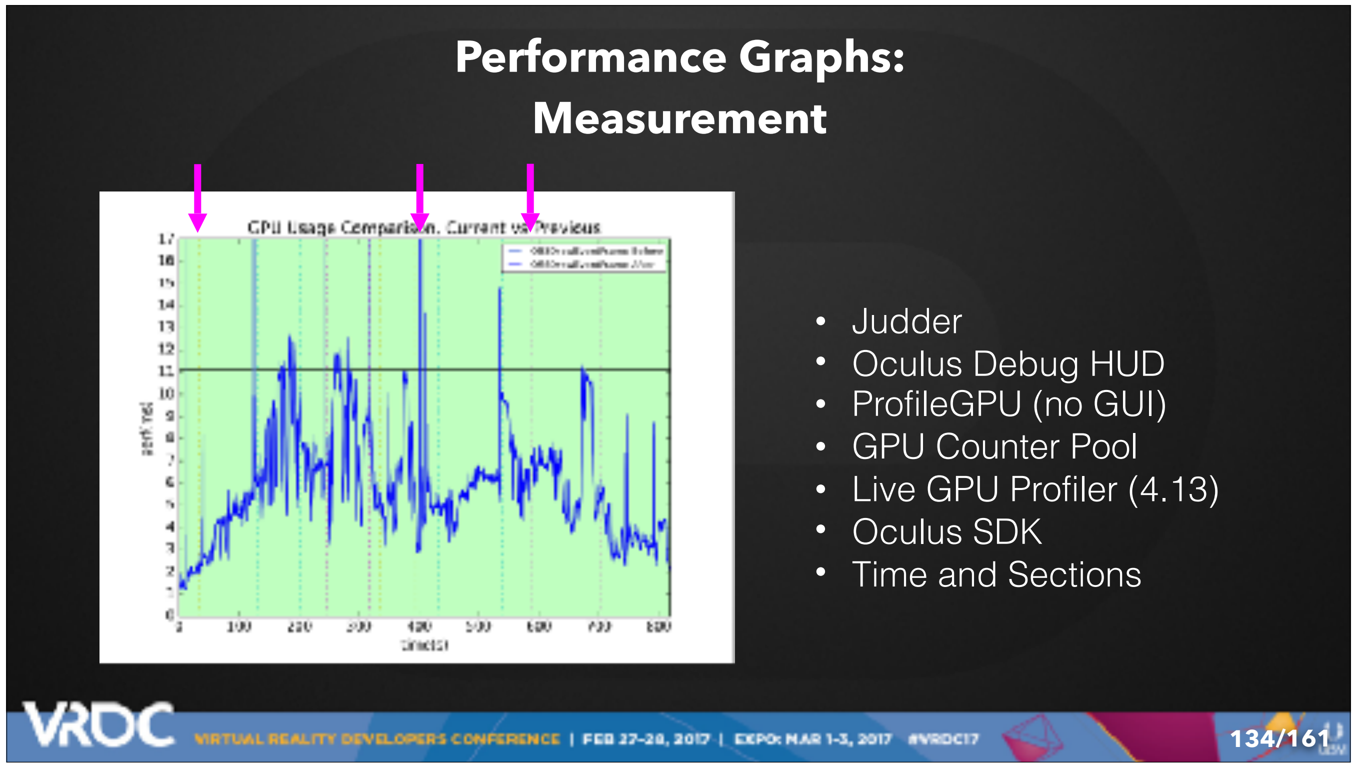


Performance Graphs: Measurement

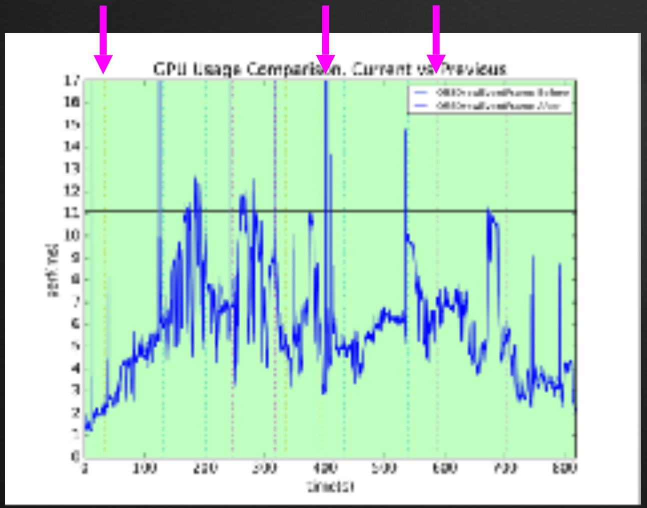


- Judder
- Oculus Debug HUD
- ProfileGPU (no GUI)
- GPU Counter Pool
- Live GPU Profiler (4.13)
- Oculus SDK
- Time and Sections

Once we could make reproducible data for the entire experience, we could prepare graphs like this one.



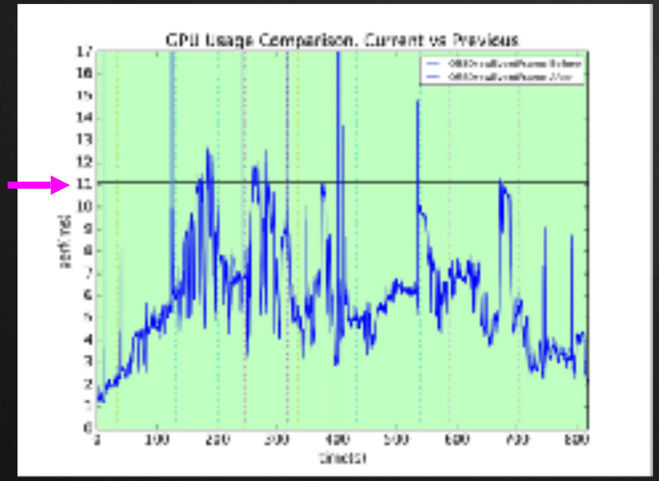
Performance Graphs: Measurement



- Judder
- Oculus Debug HUD
- ProfileGPU (no GUI)
- GPU Counter Pool
- Live GPU Profiler (4.13)
- Oculus SDK
- Time and Sections

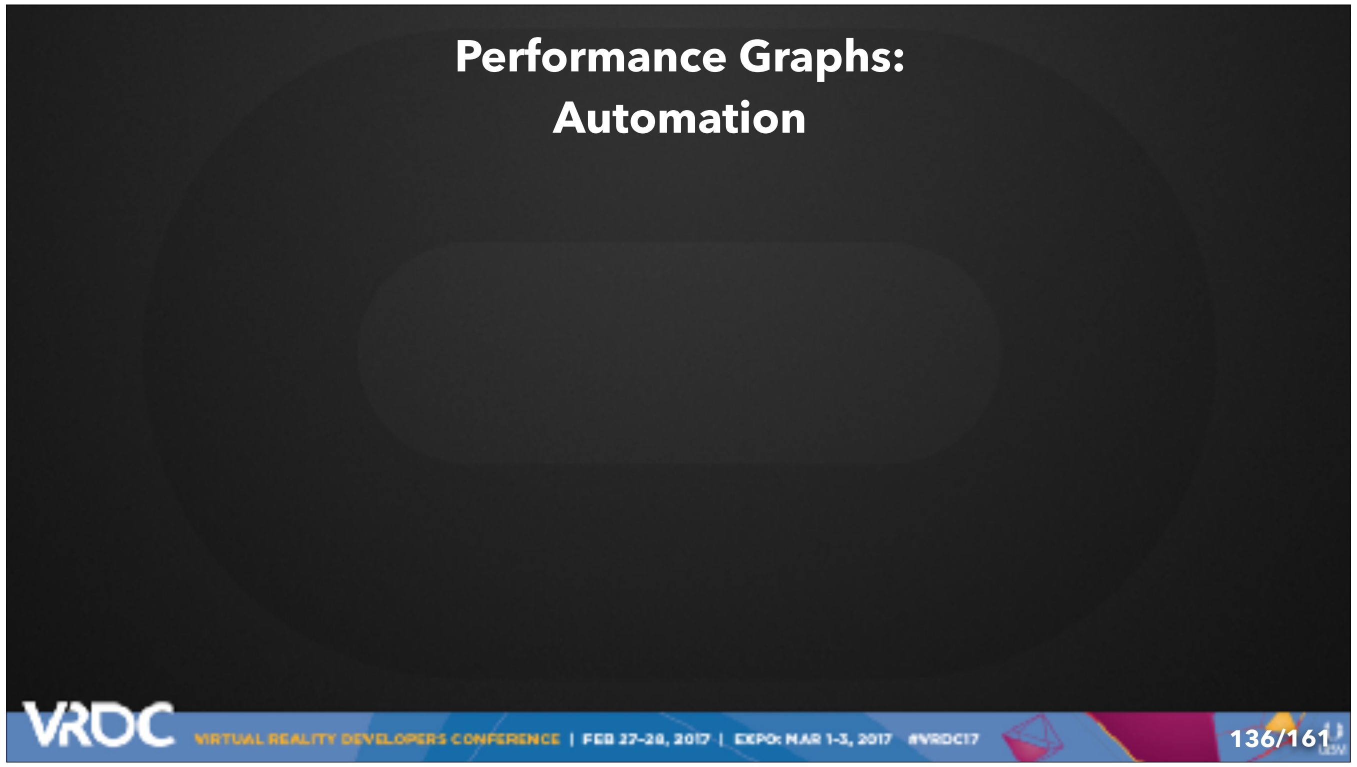
Along the bottom axis is the time of the whole piece in seconds, about 12 minutes. There are a dozen or so vertical lines in the graph indicating each major section. Some of these also correspond to peaks as things are made visible.

Performance Graphs: Measurement



- Judder
- Oculus Debug HUD
- ProfileGPU (no GUI)
- GPU Counter Pool
- Live GPU Profiler (4.13)
- Oculus SDK
- Time and Sections
- The 11.1ms Line

Note the line marking 11.1ms. You can see in this graph we were having some problems in the first main scene, just before 200 seconds, and again before 300.

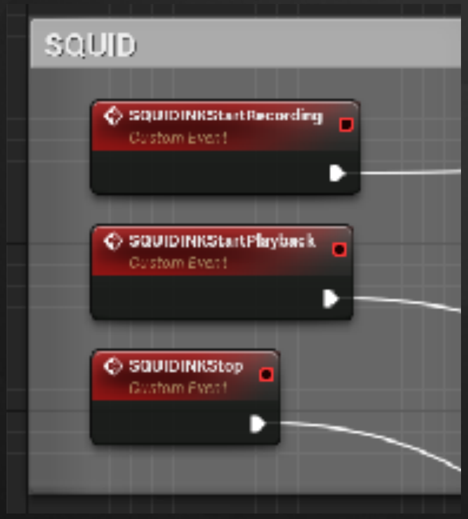


Performance Graphs: Automation

VIRTUAL REALITY DEVELOPERS CONFERENCE | FEB 27-28, 2017 | EXPO: MAR 1-3, 2017 #VRDC17

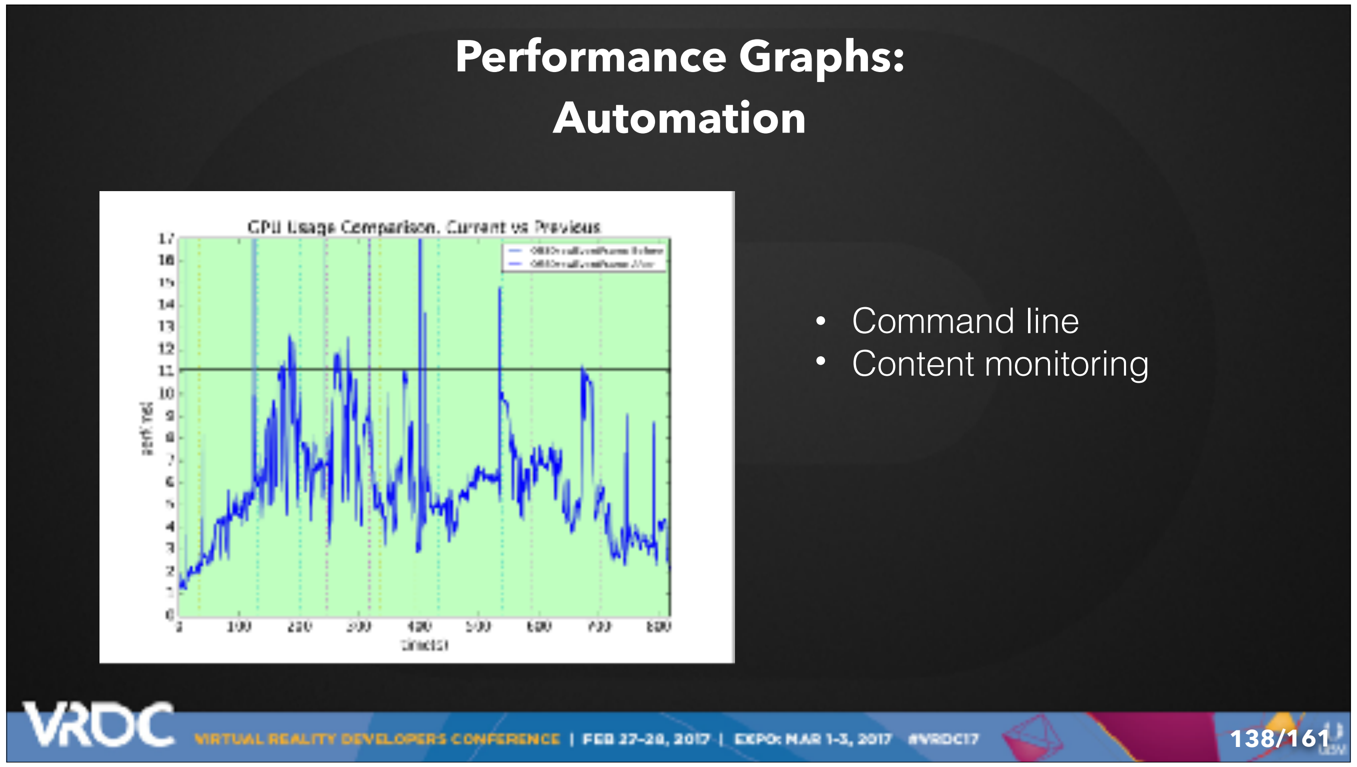
Finally, the finishing touch of automation.

Performance Graphs: Automation

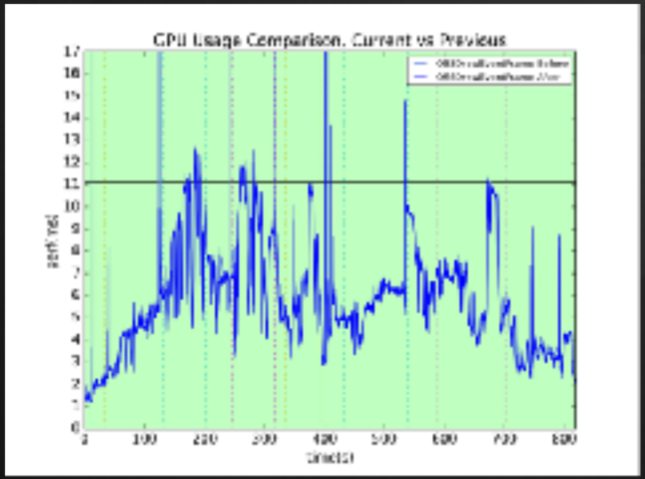


- Command line

After we added some level events so that the profiler could run on any machine with a simple batch command, we were able to do 2 things:



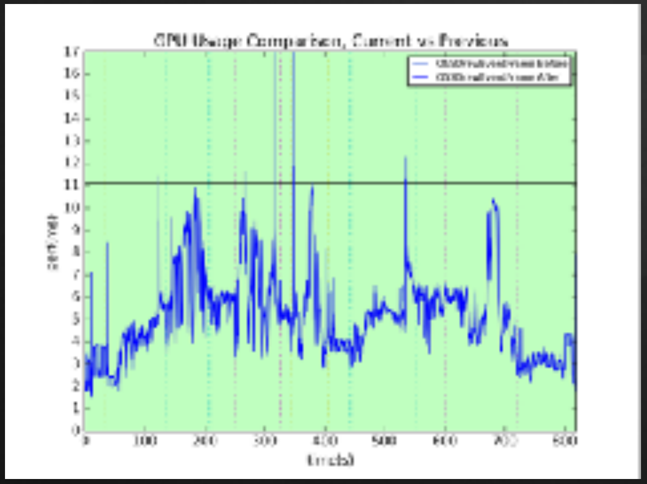
Performance Graphs: Automation



- Command line
- Content monitoring

First, we could keep a watch on any unexpectedly heavy content changes. We could easily see if something was under the 11ms line the day before and then track back to what caused the slowdown.

Performance Graphs: Automation

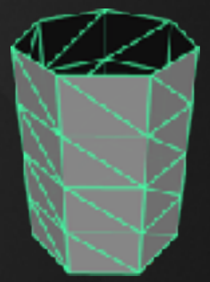


- Command line
- Content monitoring
- Optimization guarantee

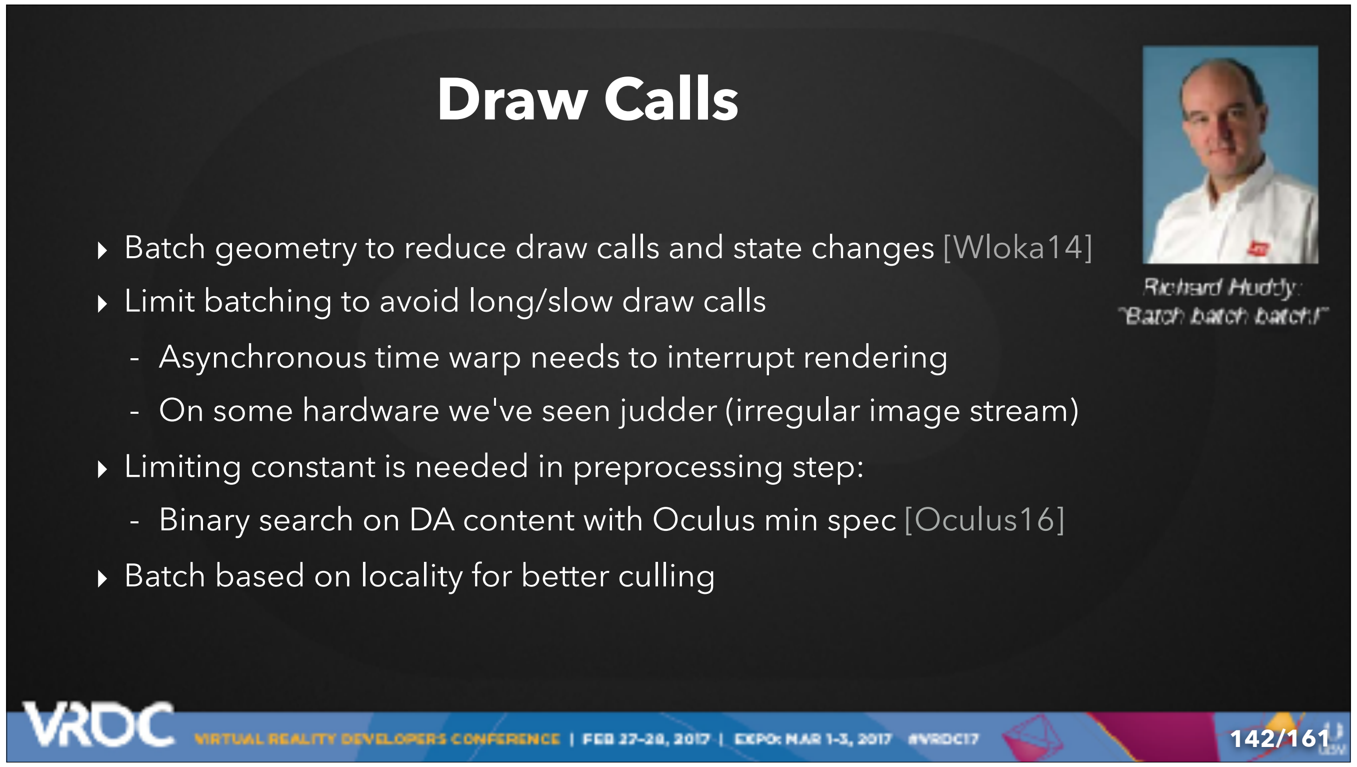
And second, as we started finalling, and optimizing for our rec spec machines, obviously content optimization would be our safest bet. But we still had several ideas for sweeping optimizations, and with a profile of the entire performance in place, we could easily measure the impact of far-reaching changes in about 12 minutes (or less if you passed a slomo value in on the command line).

And now I'll turn things over to Martin, to present some sweet, succulent rendering details.

Why using hardware tessellation?


One Instance
(more vertices
would be better)

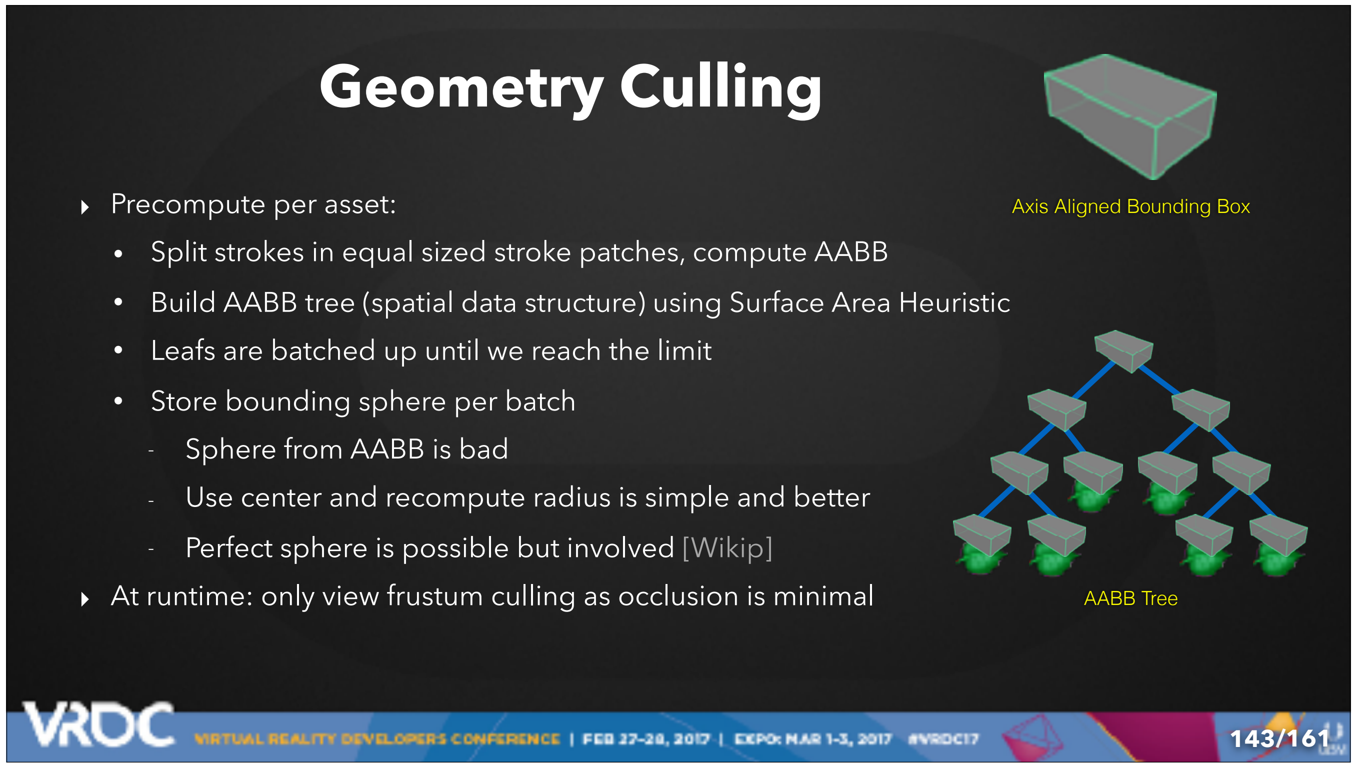
- Quill stroke / vertex data is not yet geometry
- Options:
  - **Static geometry**: Too much memory
  - **Geometry Shade**r: Convenient but slow on some hardware
  - **Hardware Instancing**: Only fast for simpler materials
    (VS runs per output vertex, artist used textures for animation)
  - **Tessellation**: Hull shader allows fine grained culling
    (VS runs per stroke vertex, patch is 16 stroke vertices)



Draw Calls

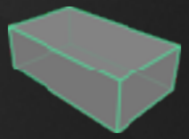
- Batch geometry to reduce draw calls and state changes [Wloka14]
- Limit batching to avoid long/slow draw calls
  - Asynchronous time warp needs to interrupt rendering
  - On some hardware we've seen judder (irregular image stream)
- Limiting constant is needed in preprocessing step:
  - Binary search on DA content with Oculus min spec [Oculus16]
- Batch based on locality for better culling

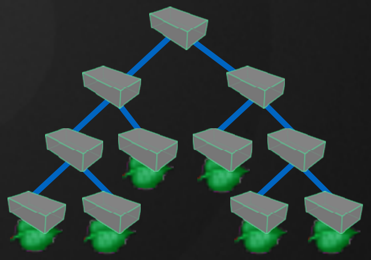
References are written in this notation: [xx] and you can find the long version in the end of the presentation



Geometry Culling

- Precompute per asset:
  - Split strokes in equal sized stroke patches, compute AABB
  - Build AABB tree (spatial data structure) using Surface Area Heuristic
  - Leafs are batched up until we reach the limit
  - Store bounding sphere per batch
    - Sphere from AABB is bad
    - Use center and recompute radius is simple and better
    - Perfect sphere is possible but involved [Wikip]
- At runtime: only view frustum culling as occlusion is minimal


Axis Aligned Bounding Box


AABB Tree

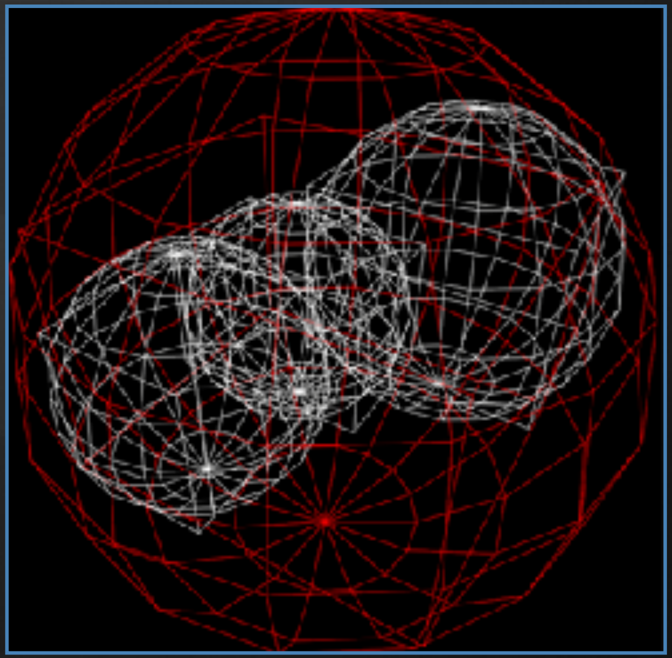
Skip rendering of not visible geometry (worst case cost needs to be less)
AABB: AxisAlignedBounding box
compute AABB: compute min max of the position including radius
SAH: Surface Area Heuristic (normally used in ray tracing)
Bounding geometry: defines cost and conservativeness
Bounding geometry other choices : k-DOP, OBB, …
Sphere primitive for simplicity, low cost, might be more conservative
Flat array / Multiple levels / Hierarchy
Some animated assets have culling disabled

Culling example

- Asset 'FXHorse2'
  - 4K input vertices
  - 32K output vertices
  - 64K triangles

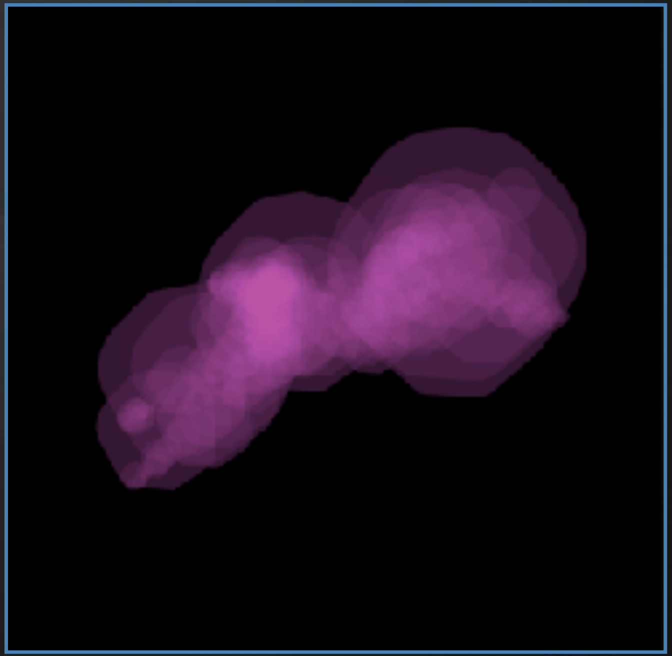
CPU culling

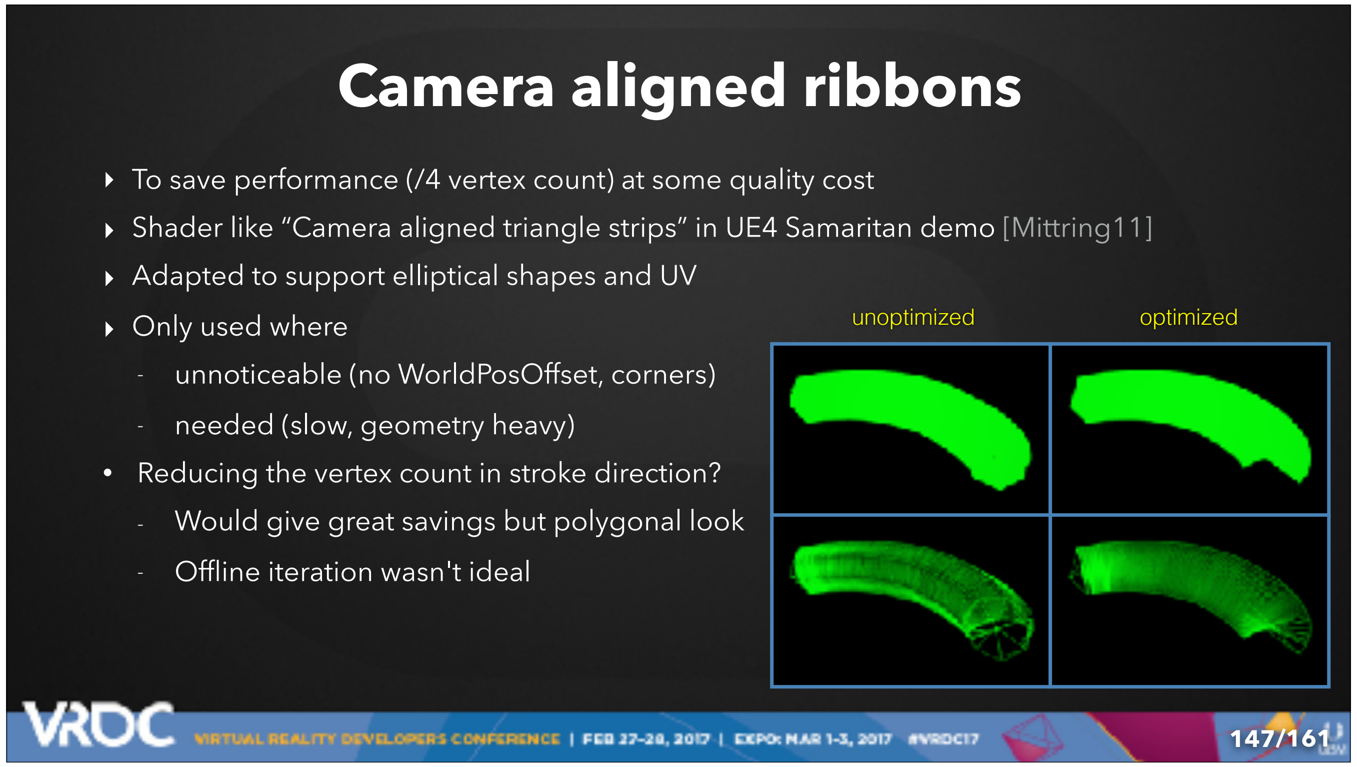
- 1 UE4 Component (Red)
- 3 draw calls (White)



GPU culling

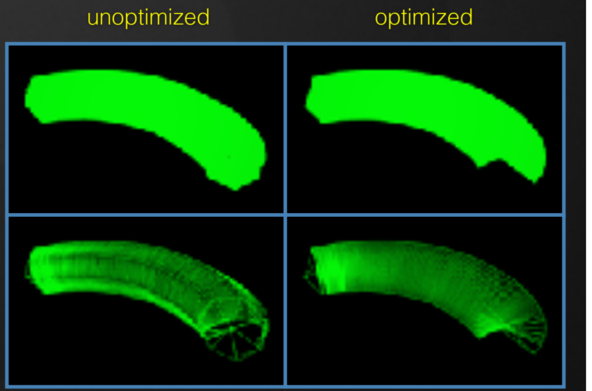
- In hull shader
  - Test sphere against 5 planes
  - TessFactor=0 to cull
- ~250 Patches/Spheres



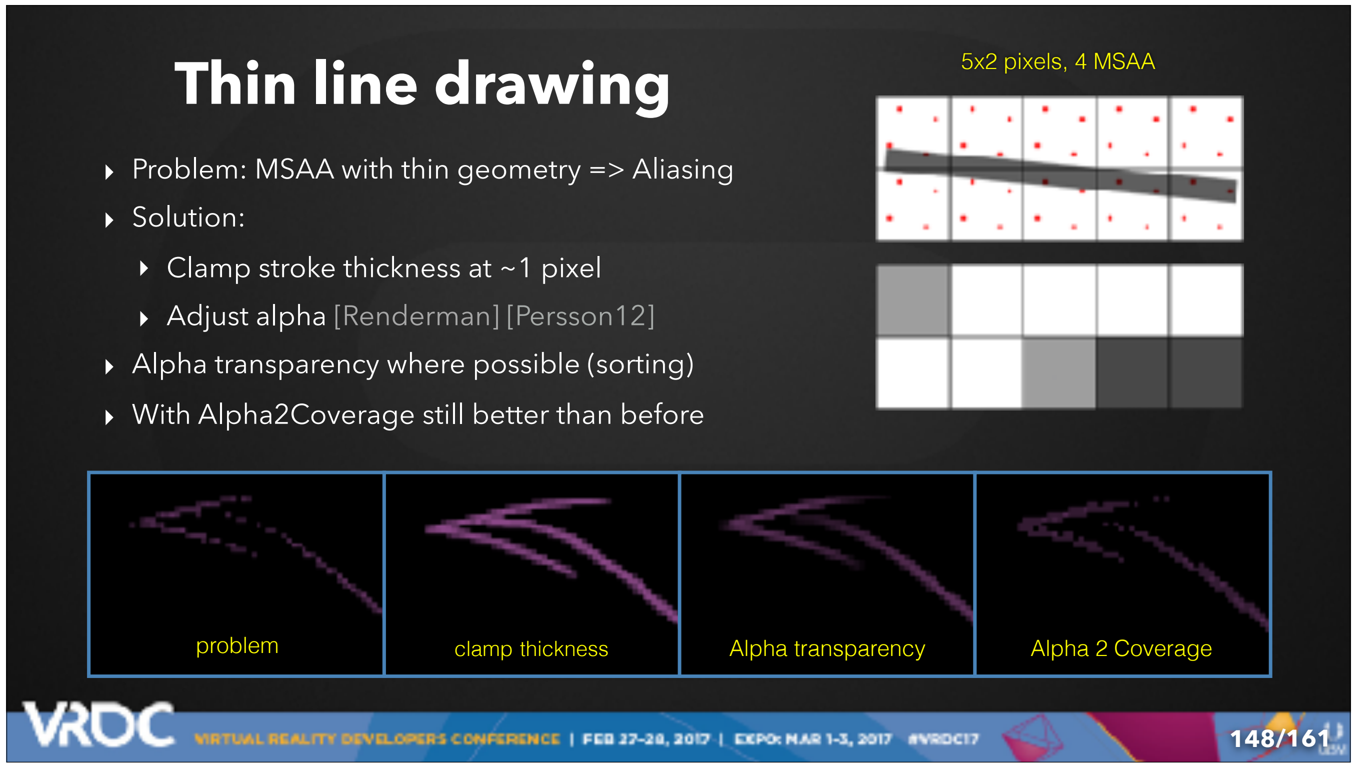


Camera aligned ribbons

- To save performance (/4 vertex count) at some quality cost
- Shader like "Camera aligned triangle strips" in UE4 Samaritan demo [Mittring11]
- Adapted to support elliptical shapes and UV
- Only used where
  - unnoticeable (no WorldPosOffset, corners)
  - needed (slow, geometry heavy)
- Reducing the vertex count in stroke direction?
  - Would give great savings but polygonal look
  - Offline iteration wasn't ideal

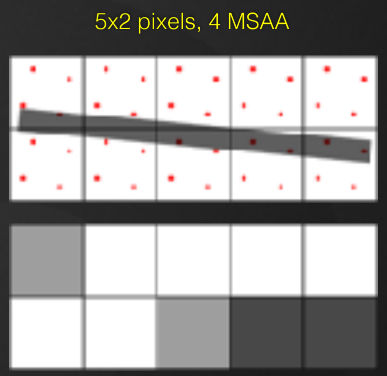


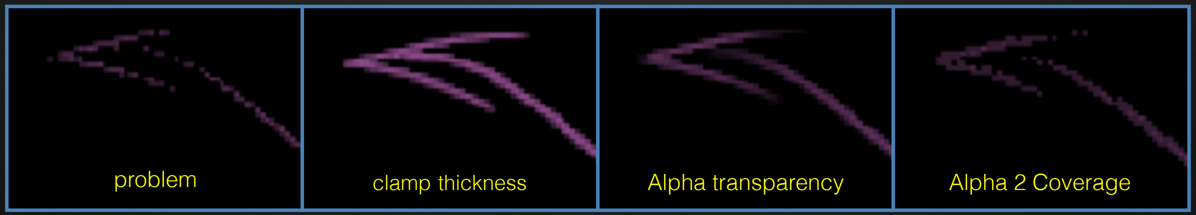
If you don't see much difference in the picture below it means the method dis a good job



Thin line drawing

- Problem: MSAA with thin geometry => Aliasing
- Solution:
  - Clamp stroke thickness at ~1 pixel
  - Adjust alpha [Renderman] [Persson12]
- Alpha transparency where possible (sorting)
- With Alpha2Coverage still better than before

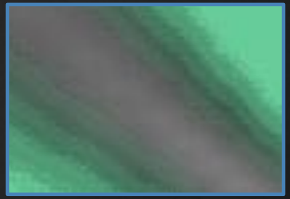


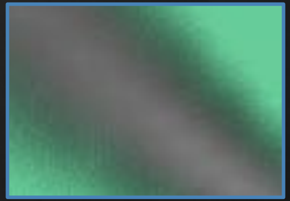


Note: Alpha translucent is bit brighter than A2C - I think this is because MSAA rendering is approximating AA from below (a partly hit sample is rejected), we could counter that but it would visually amplify the artifacts
Re-compute alpha to match opacity: Inigo mentioned this is a Renderman practice

Custom Alpha2Coverage

- Alpha transparency looks great when sorted, OIT expensive
- Hardware Alpha2Coverage is fast but improvable [Persson09]
- Our custom A2C shader version:
  - Shift bit mask by round(alpha*8)
  - Offset alpha with pseudo random 2d noise [Gjol14]
  - Animate 2d noise e.g. offset by const2 * (FrameId % 4)
  - Barrel shift mask for better edge AA with low alpha values
- Even better: precomputed blue noise texture array
- Wrong with multiple translucent layers but artist wanted that, Correct Solution [Wyman17]


Hardware A2C *


Ordered Shader A2C *

* from www.Humus.name

Alpha2Coverage = Stippled Alpha Translucency per MSAA sample
[box] noise: see [Gjol14] as triangular would be higher quality but the difference is subtle and not worth the cost here (other artifacts are more apparent)

mod 4: number is a compromise between flickering and amount of gradients

TranslucenyA2C != AlphaTranslucency because of weighting the HDR samples

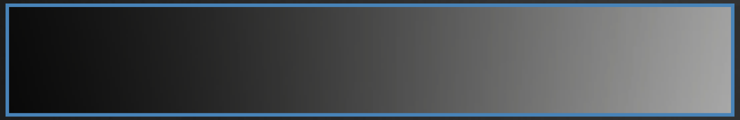
Barrel shift to improve perception of thin geometry-> without that the low opacity can degrade to no MSAA

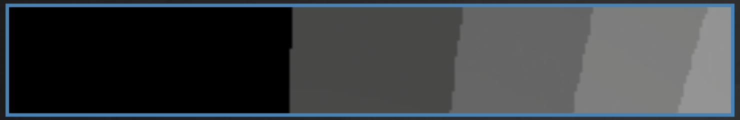
Pixel overdrive in Oculus post process fights makes this more noticeable ;(

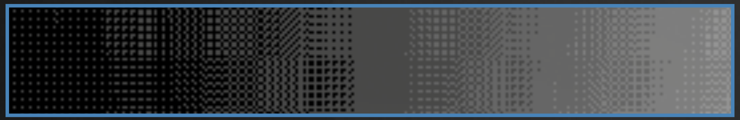
Pretty good but still improvable, not optimized (avoid integer math, consider texture lookups)

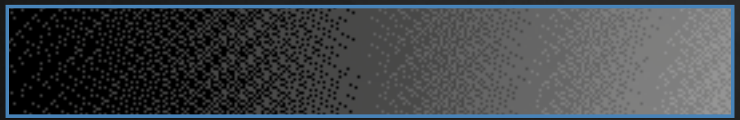
Incorrect for multiple layers but content didn't demand that

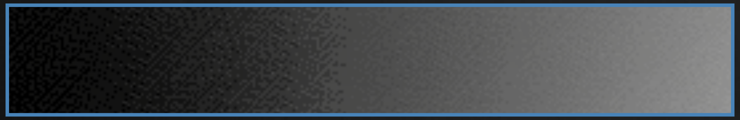
Now also used in Quill


Alpha translucency reference


Bit shift (8 shades for 8xMSAA)


Ordered

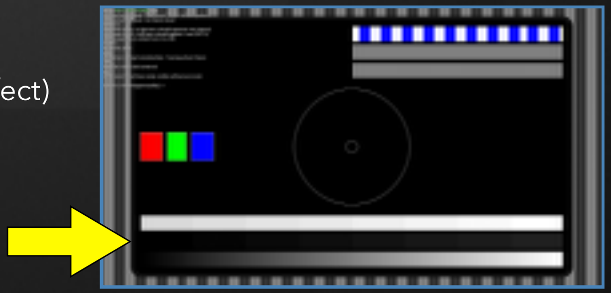

Pseudo Random [Jimenez14]


+Animated (simulated by avg. of 4 frames)

How does that look like in practice?

Investigating very dark Color Artifacts

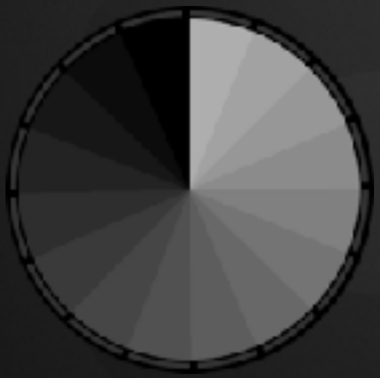
- Many TFT: 6-15 (in 0..255 range) are equally black
- How many shades depends on hardware and settings
- One the reasons why dark games ask to tweak gamma on startup
- Adapted UE TestImage
  - Show 8bit dark greys
  - Not using Headset orientation (pixel perfect)
  - Radial layout for VR

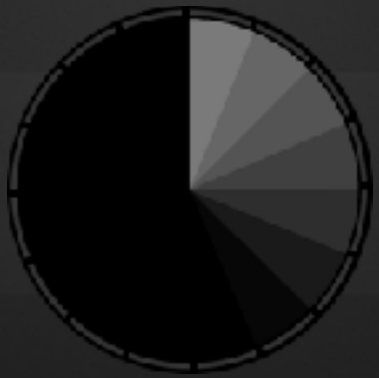


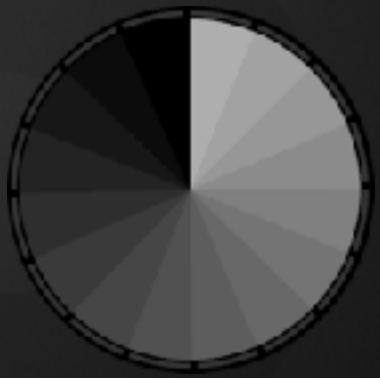
TestImage in UE4
shows 8 darkest values of 8 bit

pixel perfect: not important for 16 shades for high frequency details like dithering it can make a difference

16 Shades of Grey


16 darkest values of 8 bit (expected)

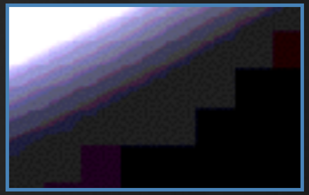

Common TFT reproduction

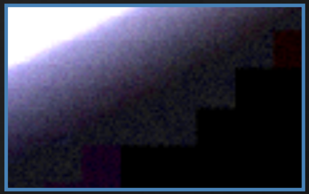

Oculus Rift Headset reproduction

* Images are simulated

Color Quantization

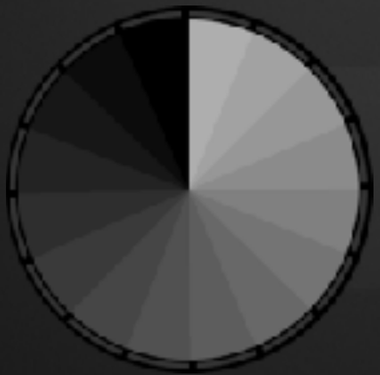
- **Scene Color** (before tone mapper):
  - 32bit for performance and Quill parity
  - Some materials add noise to avoid floating point quantization
  - Tweaked Bloom until good enough (see bonus slides) minor blocky artifacts are hidden in DA content
- **Rift input:**
  - 8 bit RGB with sRGB (at the time it was the best choice)
- **Tone mapper** (from Scene Color to 8 bit sRGB):
  - Linear with Pow 2.2 as sRGB approximation to match Quill
  - Animated noise to fight 8bit quantization [Wronski16] r.TonemapperQuality / r.Tonemapper.GrainQuantization

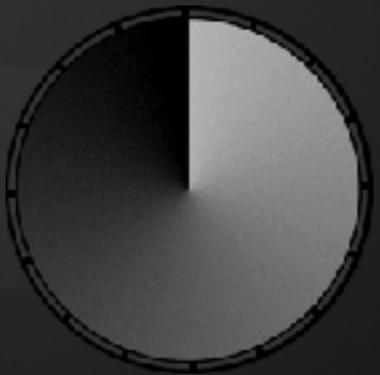

without Noise


with Noise

Rift hardware offers other formats as input as well but for 32bit 8bit per channel is best choice as sRGB filtering is not available for the 10bit format (in DX11?)
Resolution:
  Backbuffer: 2688x1600
  Rift display: 2160x1200
Note: Actual perception in the headset is even better as the noise is animated at 90Hz

More Shades of Grey

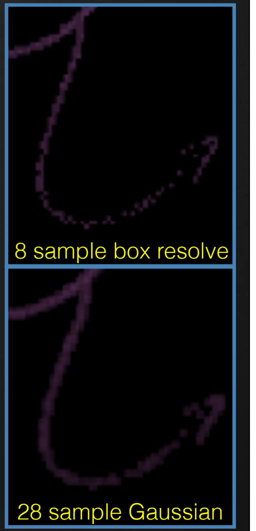

without Noise


with Noise

* Images are simulated

There is more ... we didn't use

- Cull vertex attributes (Tessellation adds complexity, done: VS to PS)
- Change ScreenPercentage at runtime (fixed OculusVR black frame glitch) vr.PixelDensity, vr.ScreenPercentageMode, [Oculus.Settings] PixelDensityMax
- Center focused rendering (distort in VS, undistort in post)
- Scalability (ini file with car settings, target min spec) [Mittring14]
- Instanced Stereo rendering (UE4.11, DX11 only allows GS)
- Non box filter MSAA resolve (wide kernel, sharpen) r.WideCustomResolve



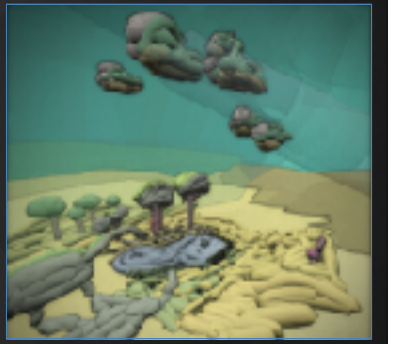
Quill

Dear Angelica was the what was Quill was created for but it doesn't stop there. Quill is actively being worked on it can be used in so many other ways. We are just beginning to realize what else is possible.

Quill and You

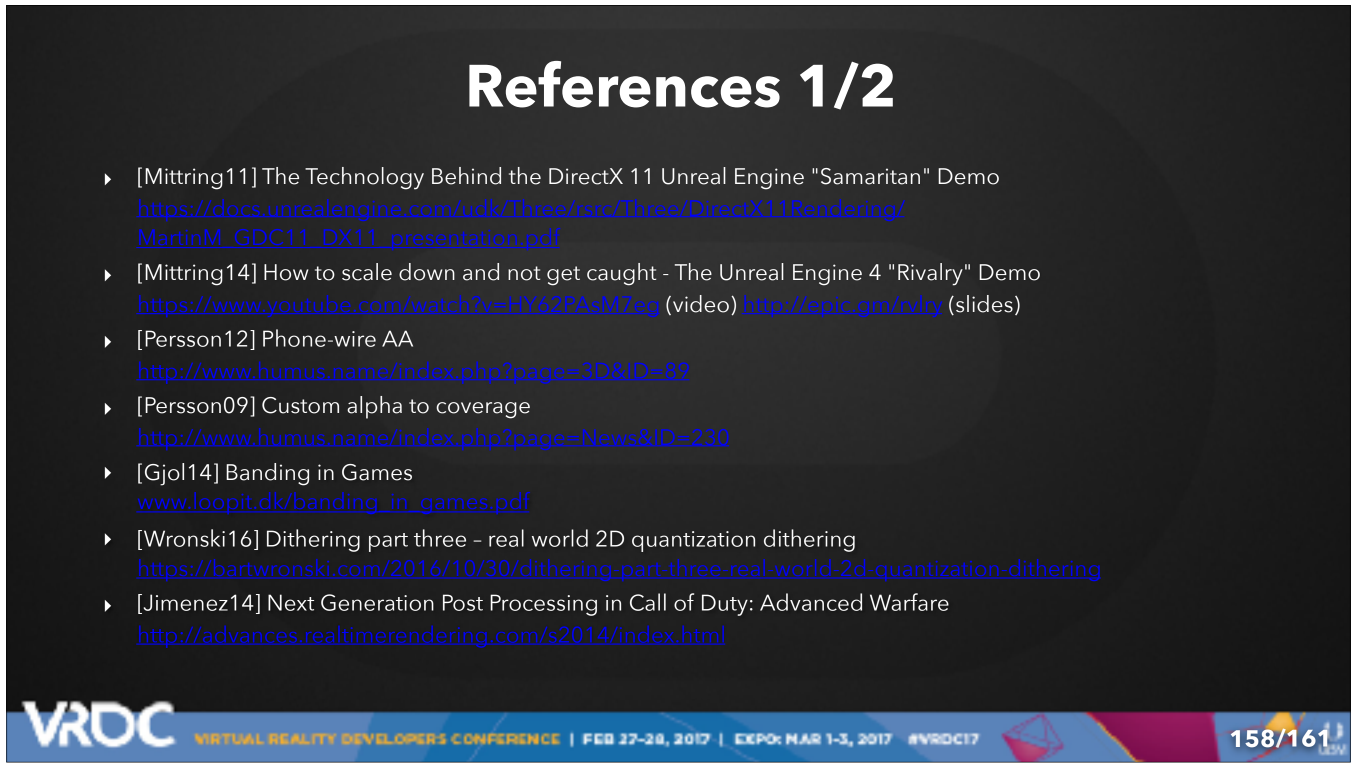
- Quill is free and available for Oculus Rift with Touch
- Quill can export to FBX and Alembic
  - as curves with per vertex data (color, width, orientation)
  - or baked pre-tesselated polygons for quick visualization
- Dear Angelica is using UE4.10 + forward + Quill asset support
- Source code:
  - Quill is using Inigos code base (C++, OpenGL) piLibs http://www.iquilezles.org/code/piLibs/piLibs.htm
  - UE4.14 + Quill asset support (C++, D3D11) we are working on the release

follow @OVRStoryStudio


NPR of Quill content

AO with Quill content

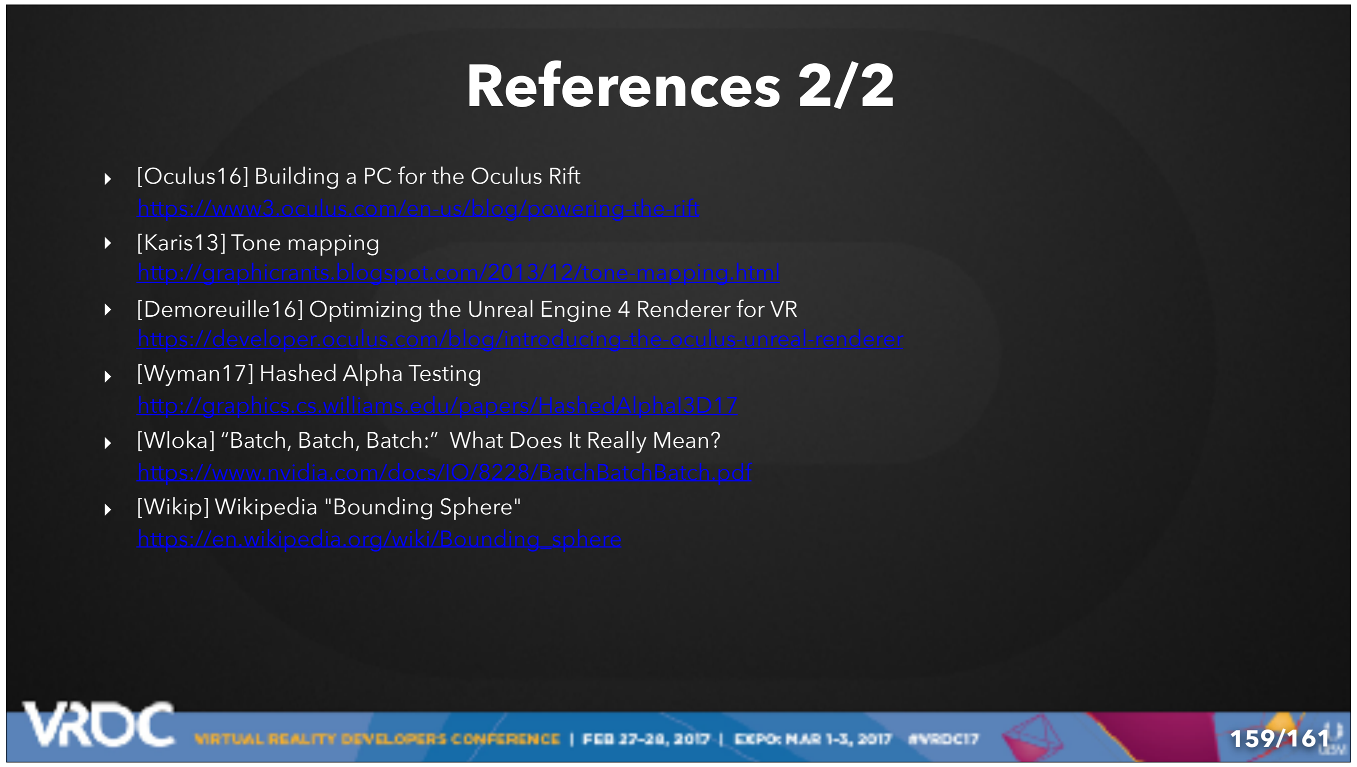
NPR: Non Photorealistic Rendering
AO: Ambient Occlusion



References 1/2

- [Mittring11] The Technology Behind the DirectX 11 Unreal Engine "Samaritan" Demo
  https://docs.unrealengine.com/udk/Three/rsrc/Three/DirectX11Rendering/MartinM_GDC11_DX11_presentation.pdf
- [Mittring14] How to scale down and not get caught - The Unreal Engine 4 "Rivalry" Demo
  https://www.youtube.com/watch?v=HY62PAsM7eg (video) http://epic.gm/rvlry (slides)
- [Persson12] Phone-wire AA
  http://www.humus.name/index.php?page=3D&ID=89
- [Persson09] Custom alpha to coverage
  http://www.humus.name/index.php?page=News&ID=230
- [Gjol14] Banding in Games
  www.loopit.dk/banding_in_games.pdf
- [Wronski16] Dithering part three – real world 2D quantization dithering
  https://bartwronski.com/2016/10/30/dithering-part-three-real-world-2d-quantization-dithering
- [Jimenez14] Next Generation Post Processing in Call of Duty: Advanced Warfare
  http://advances.realtimerendering.com/s2014/index.html

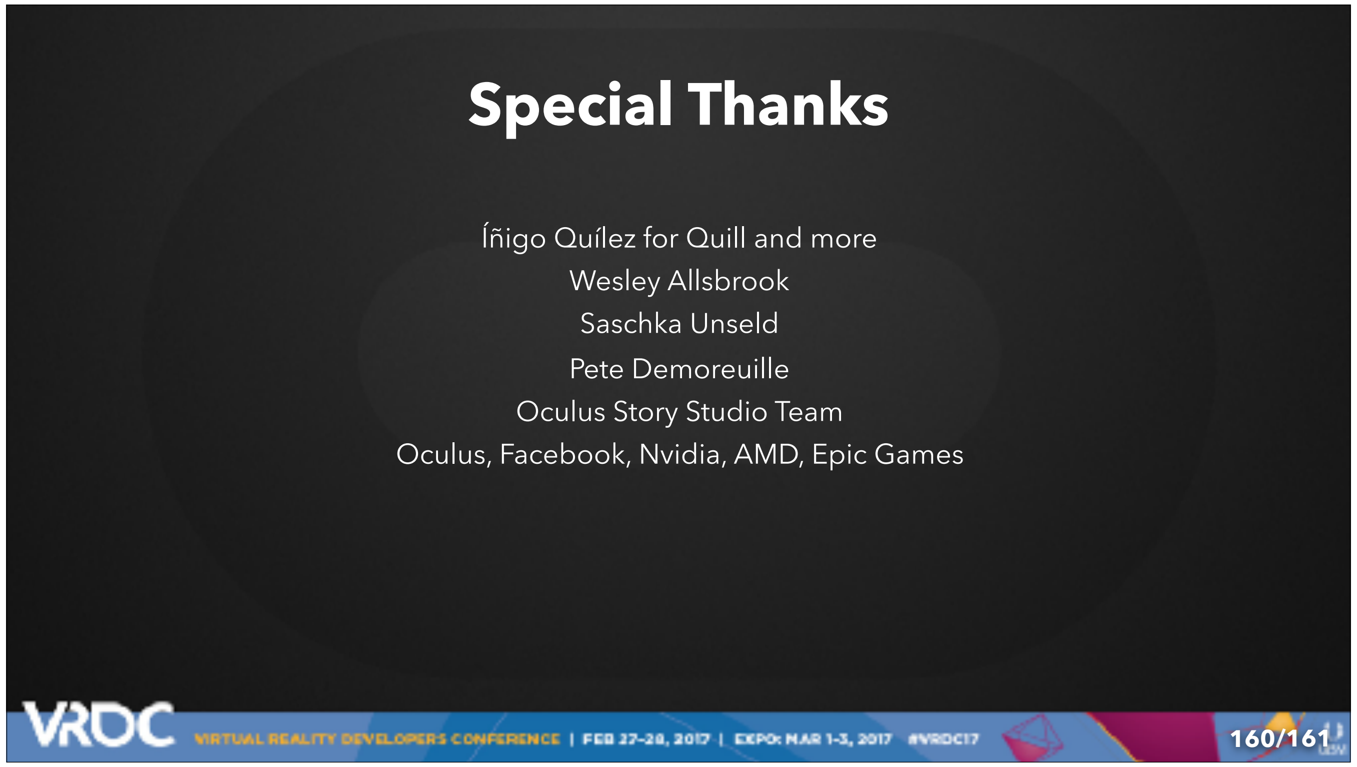
No need to read this, here you can find the references for you to be used later.



References 2/2

- [Oculus16] Building a PC for the Oculus Rift
  https://www3.oculus.com/en-us/blog/powering-the-rift
- [Karis13] Tone mapping
  http://graphicrants.blogspot.com/2013/12/tone-mapping.html
- [Demoreuille16] Optimizing the Unreal Engine 4 Renderer for VR
  https://developer.oculus.com/blog/introducing-the-oculus-unreal-renderer
- [Wyman17] Hashed Alpha Testing
  http://graphics.cs.williams.edu/papers/HashedAlphaI3D17
- [Wloka] “Batch, Batch, Batch:” What Does It Really Mean?
  https://www.nvidia.com/docs/IO/8228/BatchBatchBatch.pdf
- [Wikip] Wikipedia "Bounding Sphere"
  https://en.wikipedia.org/wiki/Bounding_sphere

No need to read this, here you can find the references for you to be used later.



Special Thanks

Íñigo Quílez for Quill and more
Wesley Allsbrook
Saschka Unseld
Pete Demoreuille
Oculus Story Studio Team
Oculus, Facebook, Nvidia, AMD, Epic Games

We are presenting the work of many and this project wouldn't be possible without many others. Here are some special thanks.
Hopefully we have time for some questions.

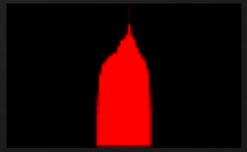
Questions?

This is Henry - from the VR experience the studio made before Dear Angelica.

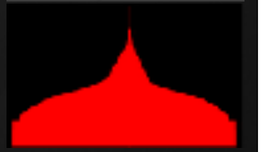
Bonus Slides

Bloom Setting Comparisons

- Reproducible image and to make the artifacts better visible:
  r.RenderTimeFrozen 1
  r.FastBlurThreshold 99999
  r.BloomThreshold -1
  vis Tonemap uv0 *10
- Code fix to get best bloom quality:
  search for "HalfResPass = " and replace "PF_FloatRGB"with "PF_Unknown"
- Settings explained:
  - Grain <x>: r.TonemapperQuality <x>
    in newer UE4: r.Tonemapper.GrainQuantization <x>
  - SceneColor 32bit: r.SceneColorFormat 2/3   which is R11G11B10
    SceneColor 64bit: r.SceneColorFormat 4
    SceneColor 128bit: r.SceneColorFormat 5 (as reference)
  - Bloom 0: narrow, using lower res images with too small radius
    Bloom 1: wider with center spike using low res images better
- Used in "Dear Angelica": Grain:1 SceneColor:32bit, Bloom:1

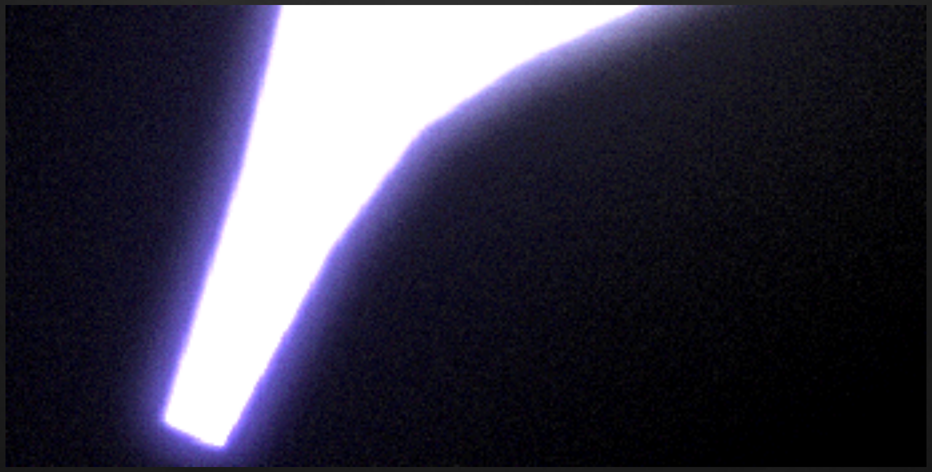


Bloom: 0

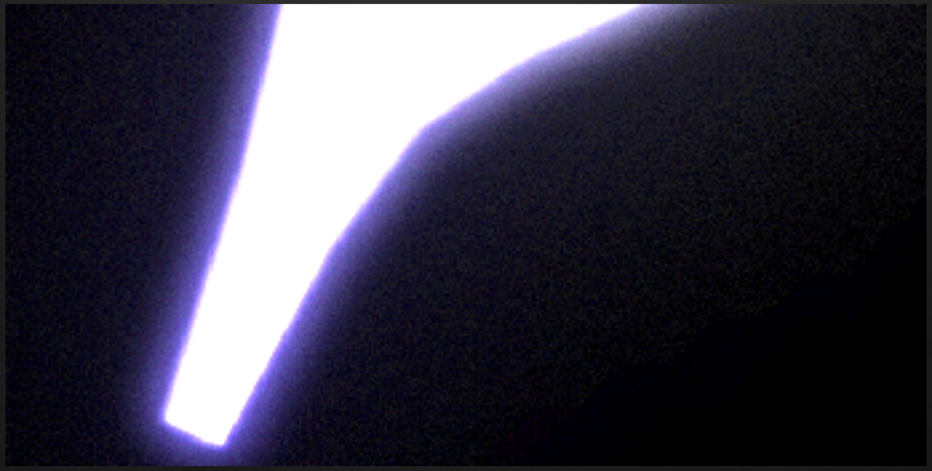


Bloom: 1

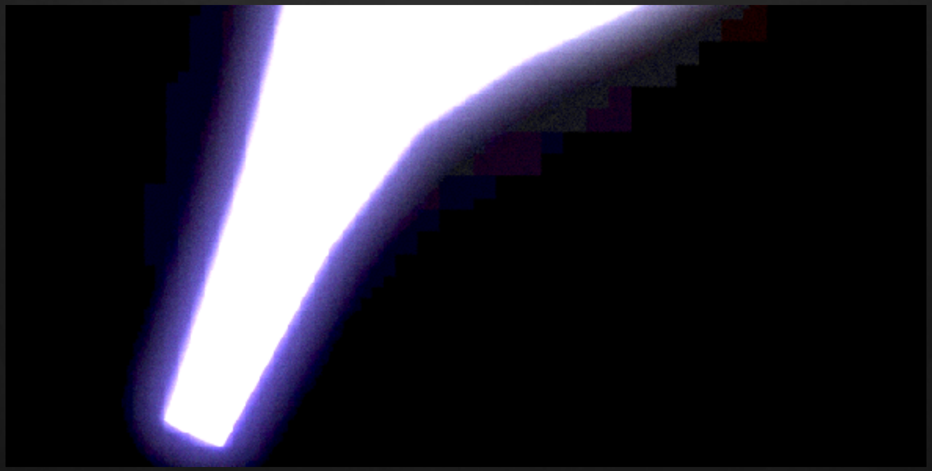
Grain:1, SceneColor: 128bit, Bloom 1 (Reference)



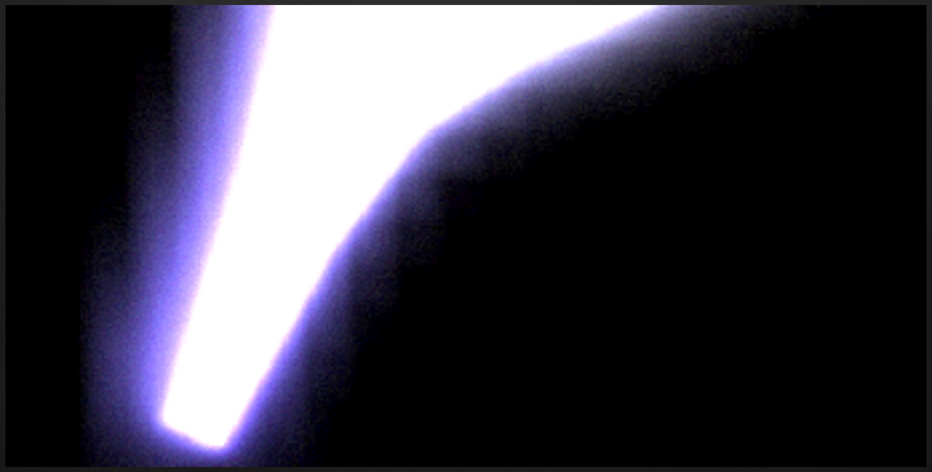
Grain:1, SceneColor:64bit, Bloom:1



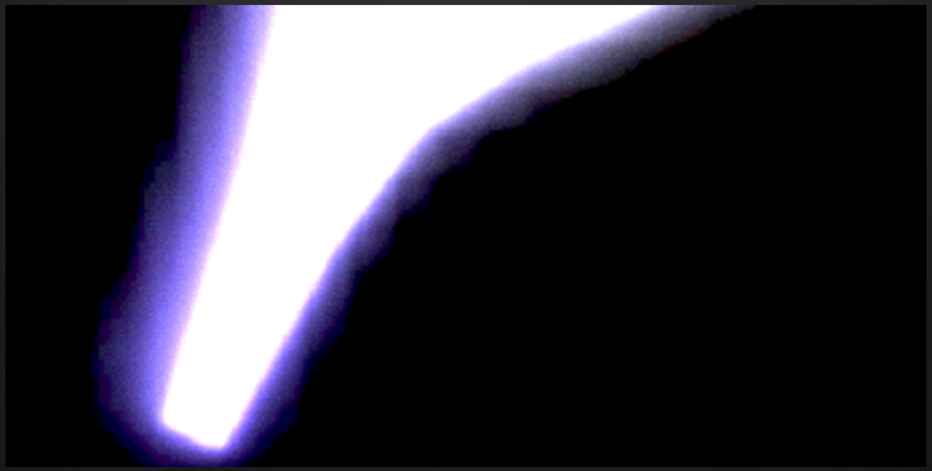
Grain:1, SceneColor:32bit, Bloom:1



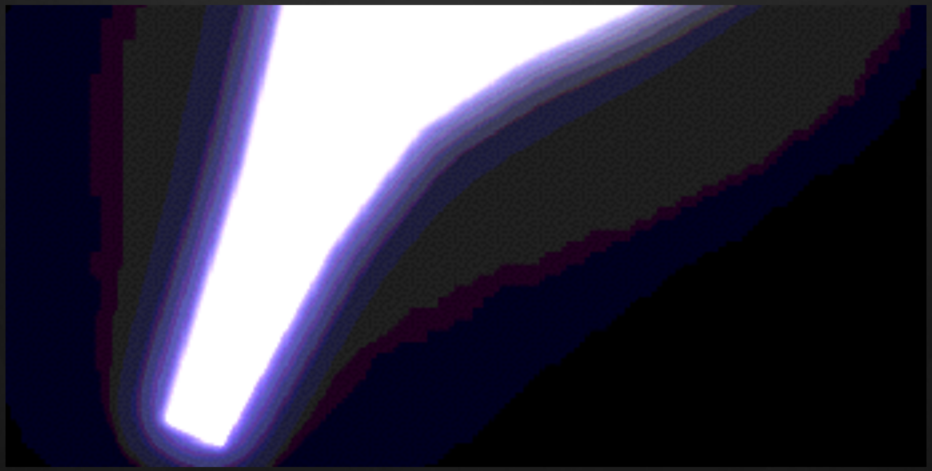
Grain:1, SceneColor:64bit, Bloom:0



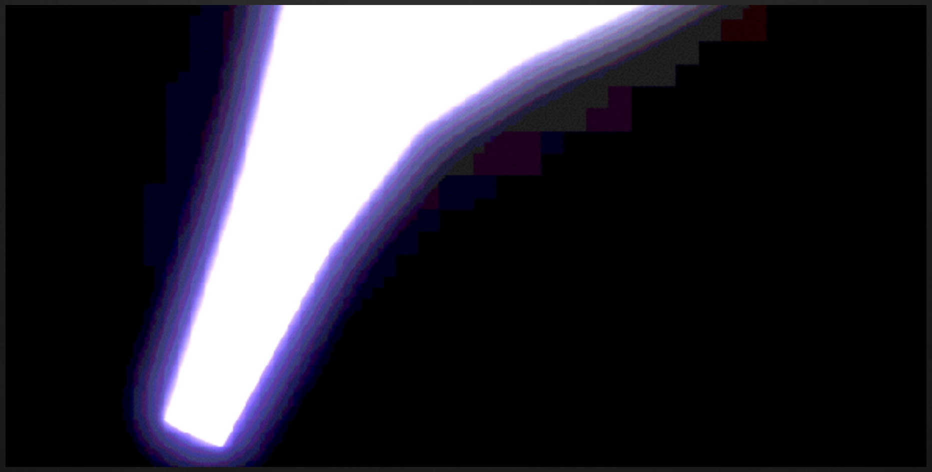
Grain:1, SceneColor:32bit, Bloom:0



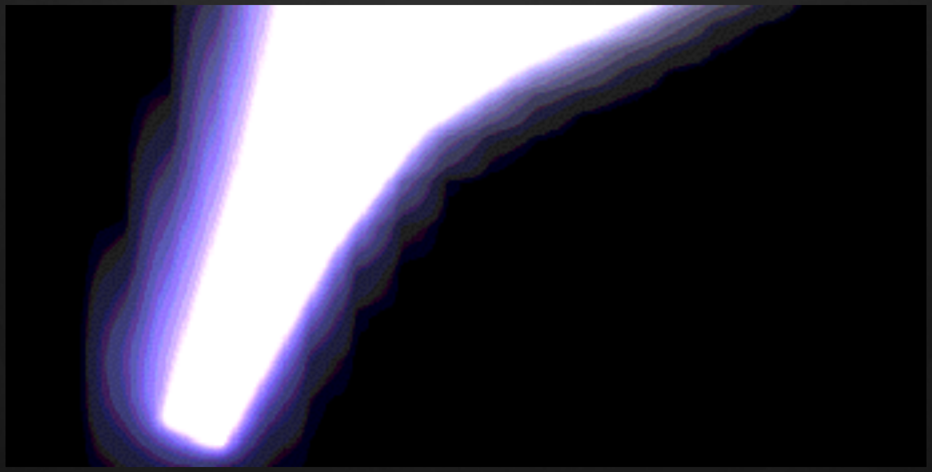
Grain:0, SceneColor:64bit, Bloom:1



Grain:0, SceneColor:32bit, Bloom:1



Grain:0, SceneColor:64bit, Bloom:0



Grain:0, SceneColor:32bit, Bloom:0