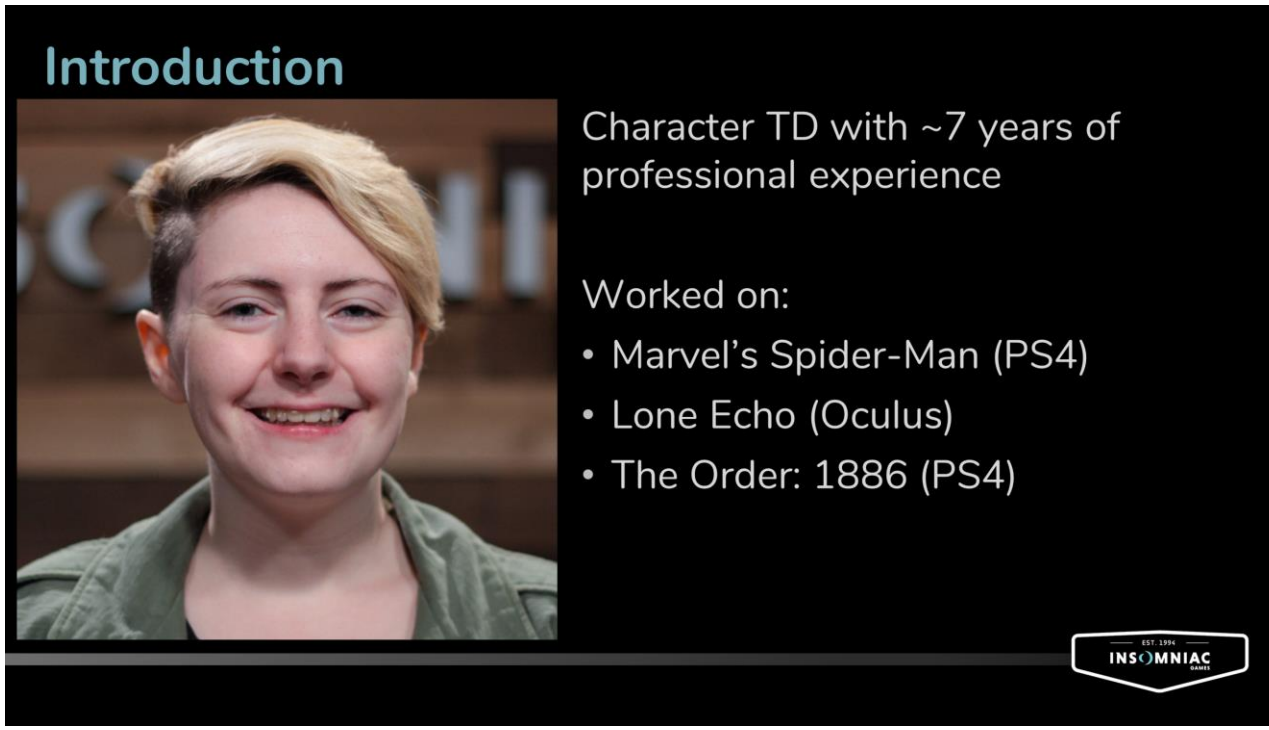
Just want to say that it’s our 25th year here as a company at Insomniac so WOO-HOO! We’re celebrating. Come check out our booth!

Sophie Brennan

Character TD



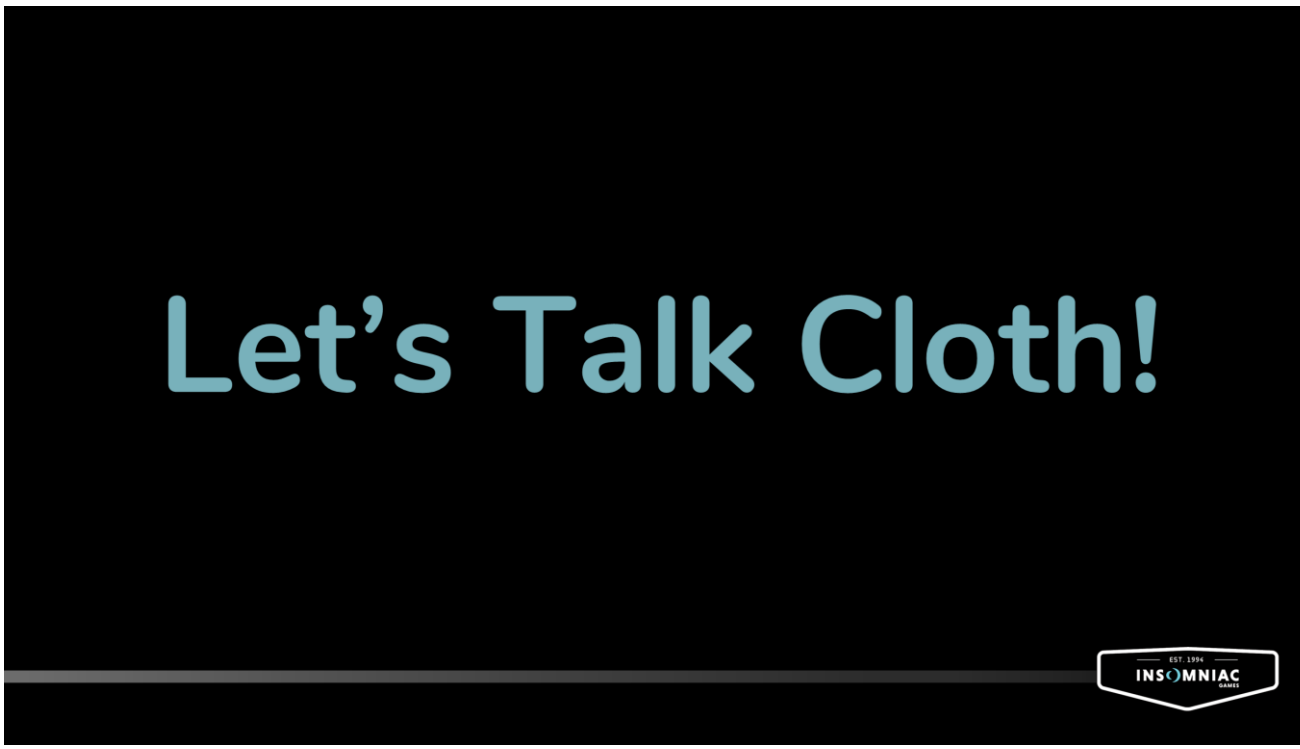
So…Hi, my name is Sophie Brennan!

I'm a Character TD/Character Rigger/Technical Animator/Character Technical Artist with around 7 years experience in the games industry. I've worked on a range of titles, from mobile, to internal R&D, to AAA and VR. I started off in the UK (did the accent give it away?) but now live in the USA. Previously worked at Ready At Dawn on titles like The Order: 1886 and Lone Echo. My most recently published title was Spider-Man, developed at Insomniac Games which is where I am right now!

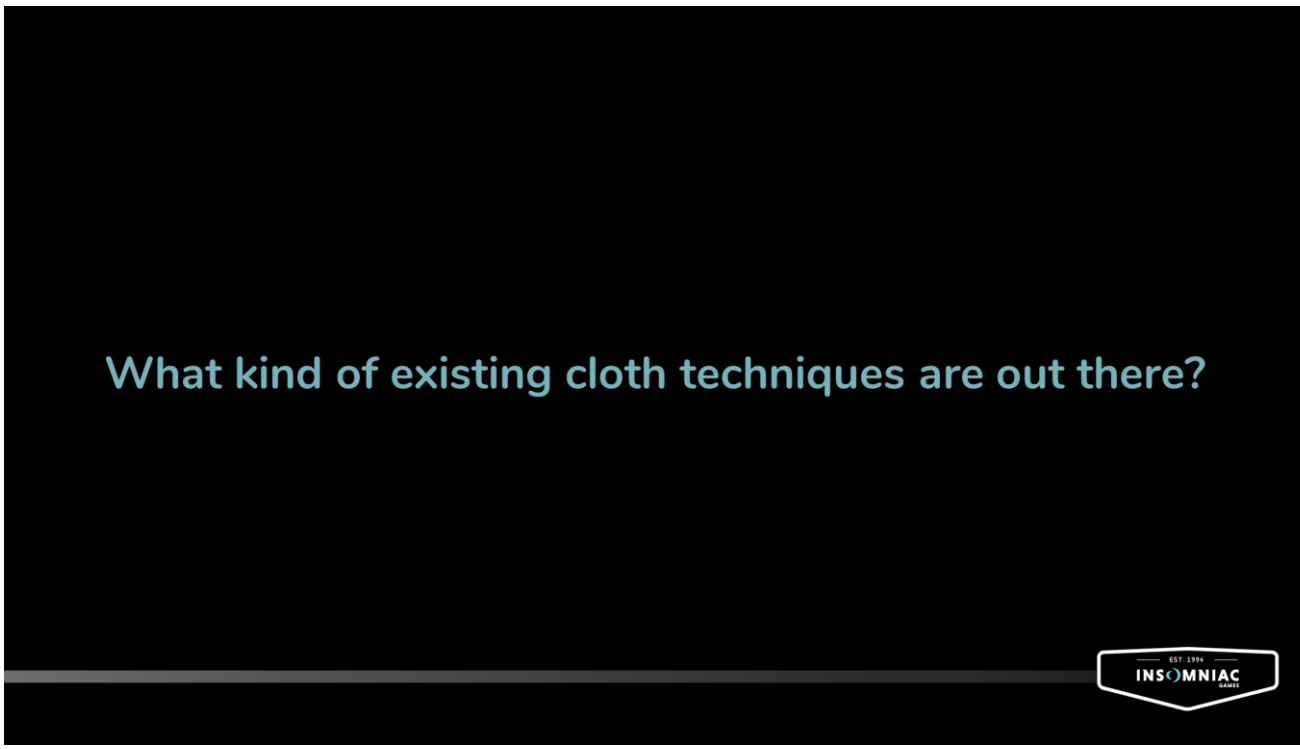
Firstly, I'd just like to say thank you to the entire team – without them this project wouldn't be possible. At Insomniac, we try to stay humble and acknowledge each other as frequently as we can. Together we made this happen and when I talk today I will talk about my work, but also the work of the entire team that enabled this game to be made.

So, without further a due – let’s talk about Spider-Man.
Firstly, I have a little cloth reel prepared for you to give you a little taster of the work we did on the game…

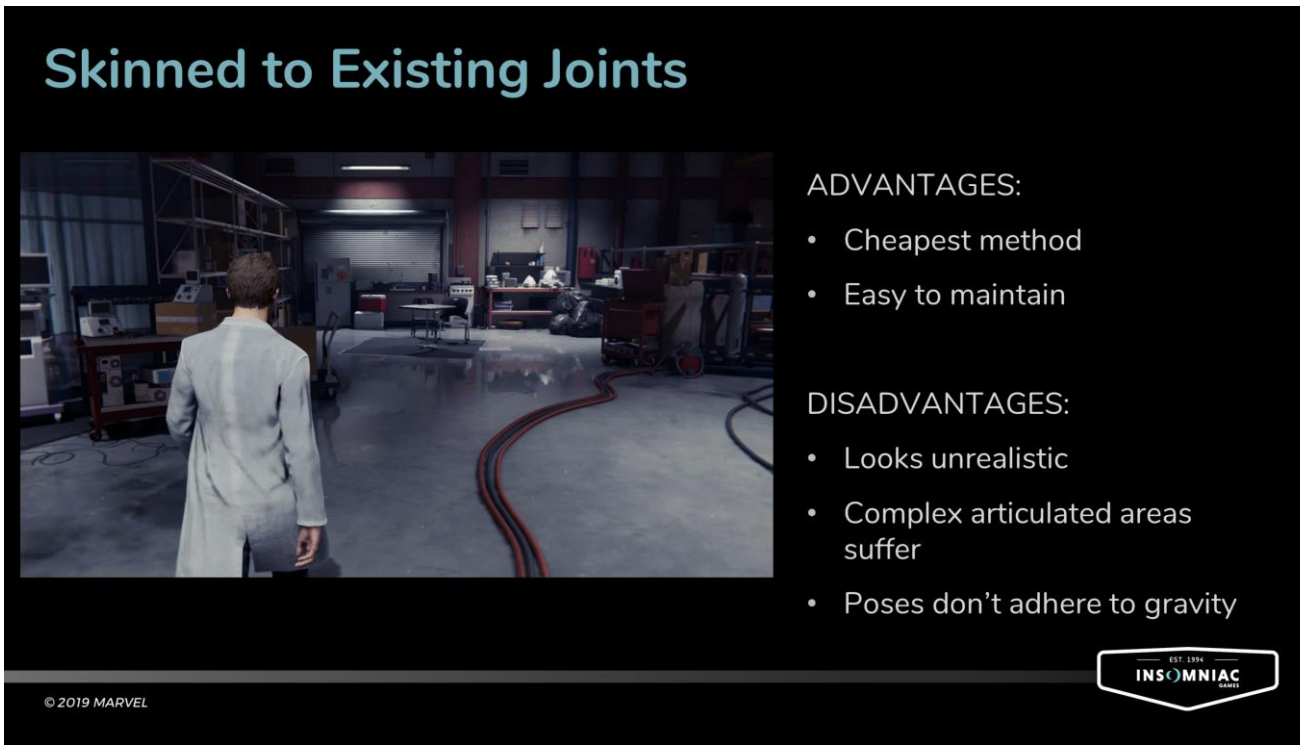
CLOTH REEL!! GO CRAZY, BABY!



Now you got the idea a little about what we did… let’s begin at the start…



So, hopefully that previous video gave you a little taste of what our cloth looked like on Spider-Man. But before we dive into our pipeline and solutions, let’s talk a little bit about the methods available to us…



First we have the simplest solution:

Skinning the cloth to existing joints.

PROS: Super easy to do and maintain.
CONS: Depending on where the cloth is it can look bad. Areas around the crotch, long skirts/dresses REALLY suffer. If the cloth is sculpted in a 'hanging shape' this will look terrible when it rotates.

You can see in the clip that's playing here the cloth looks stiff and unrealistic.

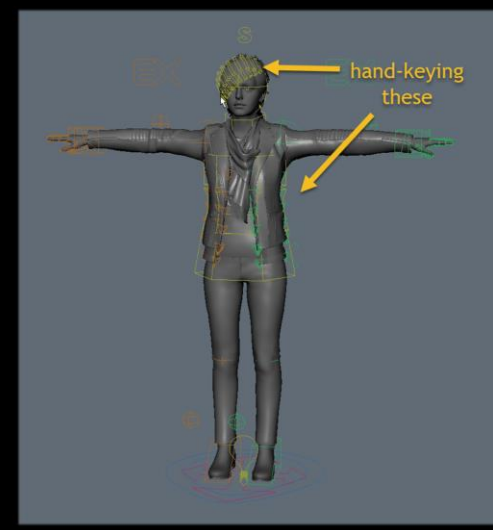
Animated Cloth Joints

ADVANTAGES:

- Can look AWESOME

DISADVANTAGES:

- Time-consuming
- Difficult to animate
- Hard to make realistic



Next, we can put joints on the cloth itself. There are several ways we can handle this…

PROS: Can look AWESOME if done right

CONS: Time consuming (depending on the setup). There can be a lot of controls and it can be difficult to animate. The burden is also entirely on animators. Even without all the controls – cloth is very hard to make look convinving

Baked/Simulated Cloth Joints

ADVANTAGES:

- Accurate and realistic

DISADVANTAGES:

- Time-consuming
- Hard to maintain
- Iteration requires another pass
- Curves are nasty for animators

We can drive the animation via pre-rendered simulation. This can be baked as an animation, or as a pre-rendered cinematic (much less common these days as most cinematics are real-time).

PROS: Accurate & realistic depending on joint fidelity and setup. Can have additive animation to correct things.
CONS: Time consuming to set-up. Hard to maintain. Depending on the animation or any changes made the setup might not work the same. Doesn't work well if elements are changing. (i.e. can't adapt well to gameplay or real-time changes like character costumes on different body types). Baked down curves suck for animators – have to work in

layers.

You can see here the animation on the hair looks awesome – but this is no mistake. This is all simulated and baked down and isn't free.

Real-Time Cloth

ADVANTAGES:

- Super flexible / reacts well to change
- Responds dynamically to player movement

DISADVANTAGES:

- Cloth quality can be low
- Detailed cloth out of scope
- Expensive (in-engine)
- Lacks self-collision

INSOMNIAC GAMES EST. 1994

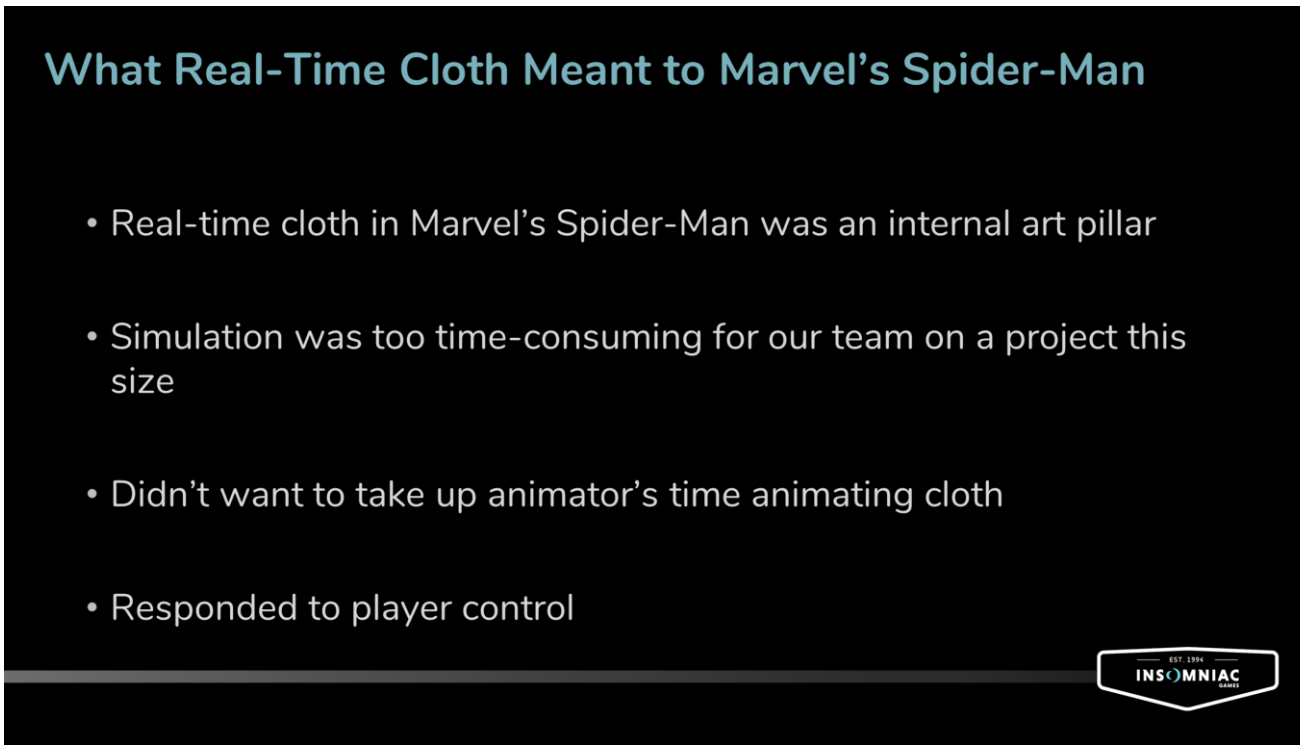
Then we have real-time solutions:

This usually requires writing your own cloth solution or using a middleware solution.

PROS: Super flexible, reacts well to change.

CONS: cloth fidelity might not be able to be very high. Getting micro wrinkles and folds real-time at game resolution is out of scope for most assets. Expensive to run as it’s constantly being calculated and it lacks self-collision.

So why did we decide to over real-time cloth over all the other options out there?



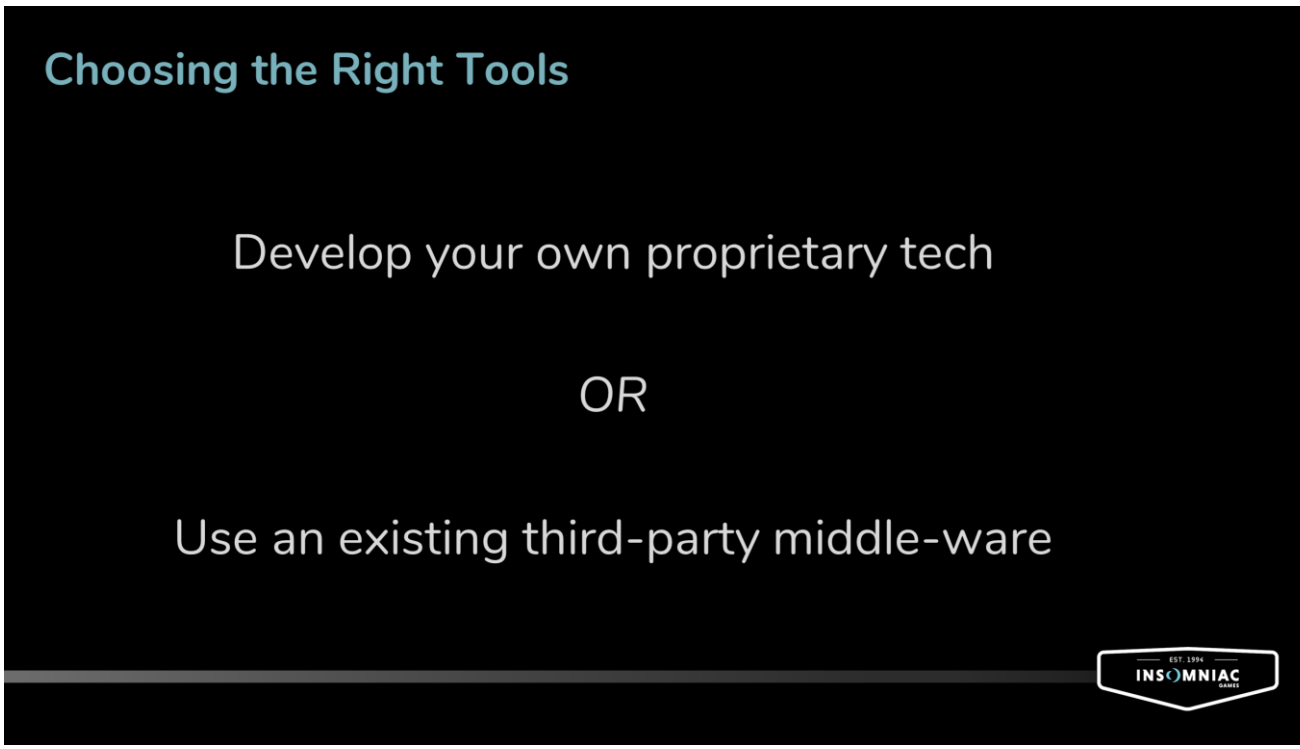
Well, one of our internal pillars for Spider-Man was 'realistic cloth'. We wanted Spider-Man to look as good as it could be and thought that real-time cloth would help set us up to one of the leaders in visuals. We also believed it would help achieve of world of a 'believable real world'. As part of making the game feel as real as possible, we had to make it LOOK real too.

On previous projects we had fallen down the rabbit hole of simulation and saw what a time sink this could be.
Add that to the amount of content in our game, the size of our team and the ever-changing nature of it – it seemed wiser that we try real-time solutions versus baked

simulations.

A personal goal of mine was to remove as much additional work from animators as possible. The last thing they needed was to be animating cloth – so the idea in general was to have no controls on the rig for them. This philosophy worked very well in most instances, but did catch us up at points, especially towards the end – but we'll talk about that later…

Finally, the big one – it responded to player control! None of the other methods listed can react real-time to player movement. Clip and baked simulation will pop between poses rather than fluidly follow through. We wanted some real secondary motion!



A big component of real-time cloth is are the tools used to achieve it.

So our choices were to develop our own tech or use existing tech out there. As we have our own in-house engine, anything tied to a specific engine was written off right off the bat.

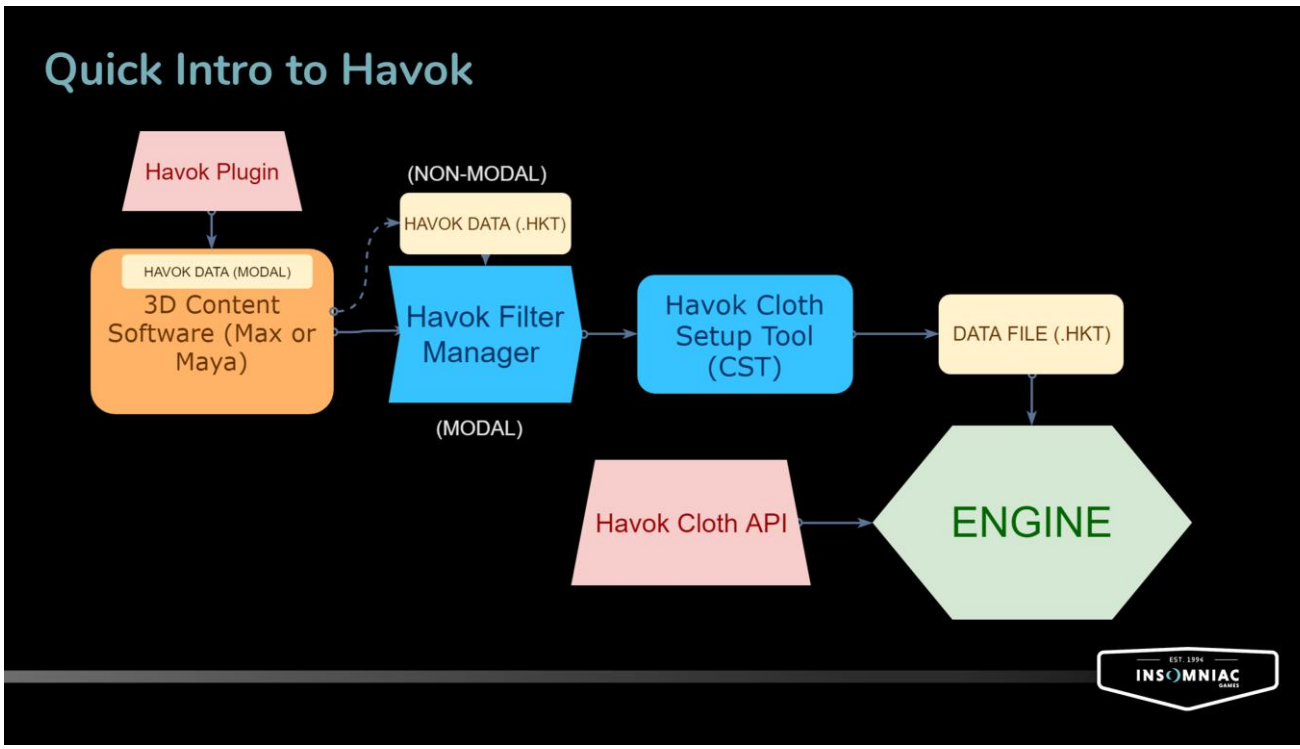
Developing your own tech has big pros and cons:
PROS: complete control over what features you want and need to focus on, design your own workflow to match your needs
CONS: huge upfront cost, will come online much later than is

needed, maintenance, no external support

However, this was just simply not in our budget with the results that we wanted to achieve.

We decided we needed to go third party.

Our choices were fairly limited (there aren’t many solutions out there) but we already used Havok for our soft and rigid body physics and had previously used their cloth technology in Sunset Overdrive, so Havok seemed most viable.



So let's talk a little bit about Havok.

Havok is essentially a plug & play system. So for us, all the tools were already there. While I don't want to get into the full details of Havok itself, I want to explain the pipeline process and tools for context.

The Havok Pipeline is as such… first you author your content in your 3D software of choice between Maya and Max. Havok has support for both.

You author a mesh in your 3D program and assign it some properties via vertex color channels. This will be a sim mesh

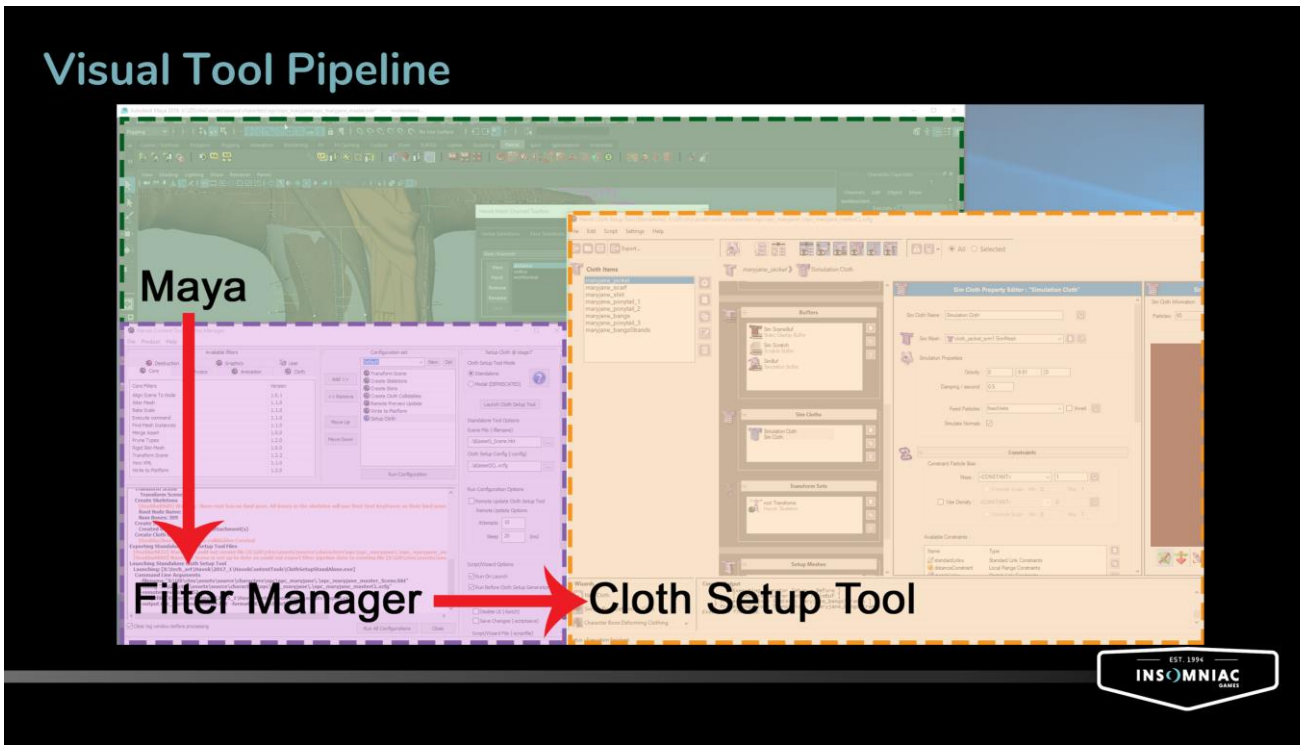
that will in turn be driven in-engine real-time.

The sim, if you want it to collide, will need colliders – from the cheapest pills/capsules to more complex shapes. These are parented to the skeleton and driven by single joints.

From there, you filter this content thru their filter manager (as it is multi purpose and supports all Havok content that comes from other files) and it is taken into their Cloth Setup Tool (or CST).

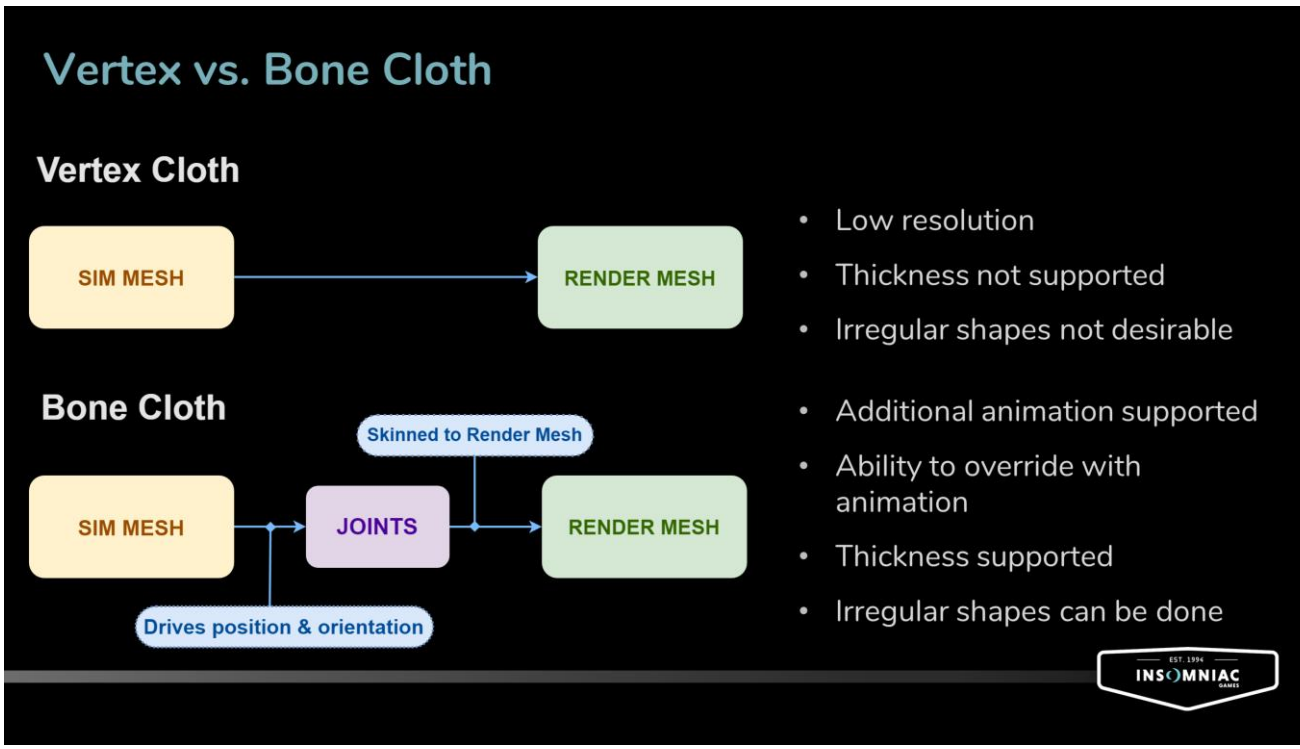
In the CST, you can edit the properties of the cloth, add behaviours or constraints, assign your colliders and such…

When all is said and done, Havok outputs it to a data file – an .hkt – which is read by the engine. This contains all the cloth sim mesh data as well as the properties assigned to it.



This is a little visual of what these tools look like within Maya itself.

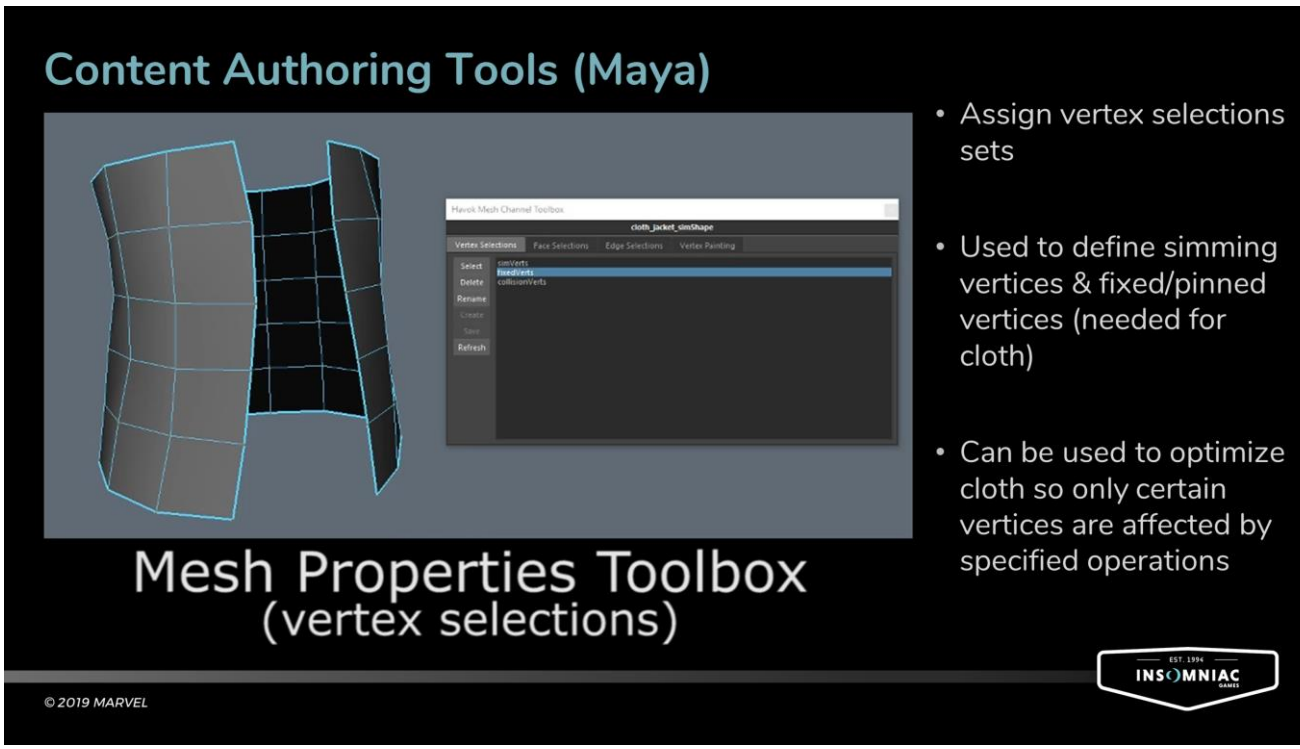
You can see here that we go from our 3D authoring package, to Havok's filter manager (on the bottom left) to the CST (on the right). Which then exports an .hkt file that is read by the engine.



Havok supports both vertex and joint-based sim meshes. What does that mean?

Well, vertex Cloth takes your sim mesh as is and uses it directly in the game. This means you need to have the correct materials applied to the sim mesh. However, this also means double-sided cloth isn't possible (or at least will probably not look good if Havok even deems the surface valid) and will have to be fairly low resolution to run well in engine. It also means content changes to any cloth might affect the behavior of the cloth. Also, irregular shapes aren't great for simming – as Havok prefers evenly sized quads when it sims.

Bone or Joint Cloth means a sim mesh that you create will drive joints (on the surface), which you can in turn use to drive mesh. This also allows for the possibility of additional animation – and for us to turn off the cloth and override it if necessary. It also complicated things. As you can see from the diagram, there are now a few more layers of content to debug when things go wrong.



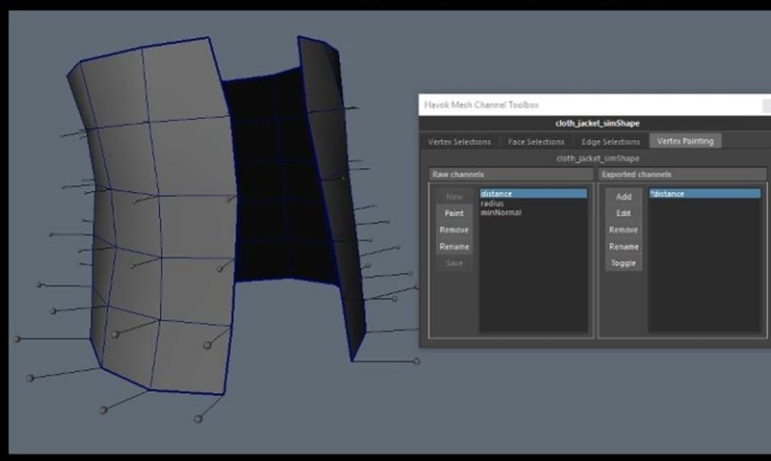
Let’s talk a little bit about creating the initial sim mesh for Havok. Firstly, we use Maya at Insomniac so we had to install their plugins. I apologise I didn’t use it in Max so I can’t really speak for that might entail.

So what does this look like inside Maya? There are a bunch of tool windows we can use to handle things.

You can see in the video here the Cloth Properties window. This is all data set on the mesh itself, and is mostly handled thru native Maya Interfaces like the Attribute Editor/Channel Box. Here we can assign vertex selection sets to the cloth. These are used to define what vertices are simming versus

those that are pinned. You can also use it to create selection sets of vertices for optimision.

Content Authoring Tools (Maya)



- Paint properties on mesh via vertices
- Custom channels that can be used to control things like:
  - Distance
  - Mass
  - Minimum Normals
  - Particle Radius
- Set min/max limits on properties

Mesh Properties Toolbox
(painting vertex properties)

EST. 1994 INSOMNIAC GAMES

You can also create channels that can hold specific vertex data. This can control many things like Distance, Mass, Minimum Normals and Particle Radius.

You can see here you can also visualize the data within maya – so here I am looking at the distance data I painted onto the vertices on the sim mesh itself – utilizing Maya's painting interface.

You can have as many sets as you deem necessary, and can make multiple outputs from a single set. (For example, you may want a gradient of 0-1, top to bottom for both distance and mass – this data can be split into multiple channels.

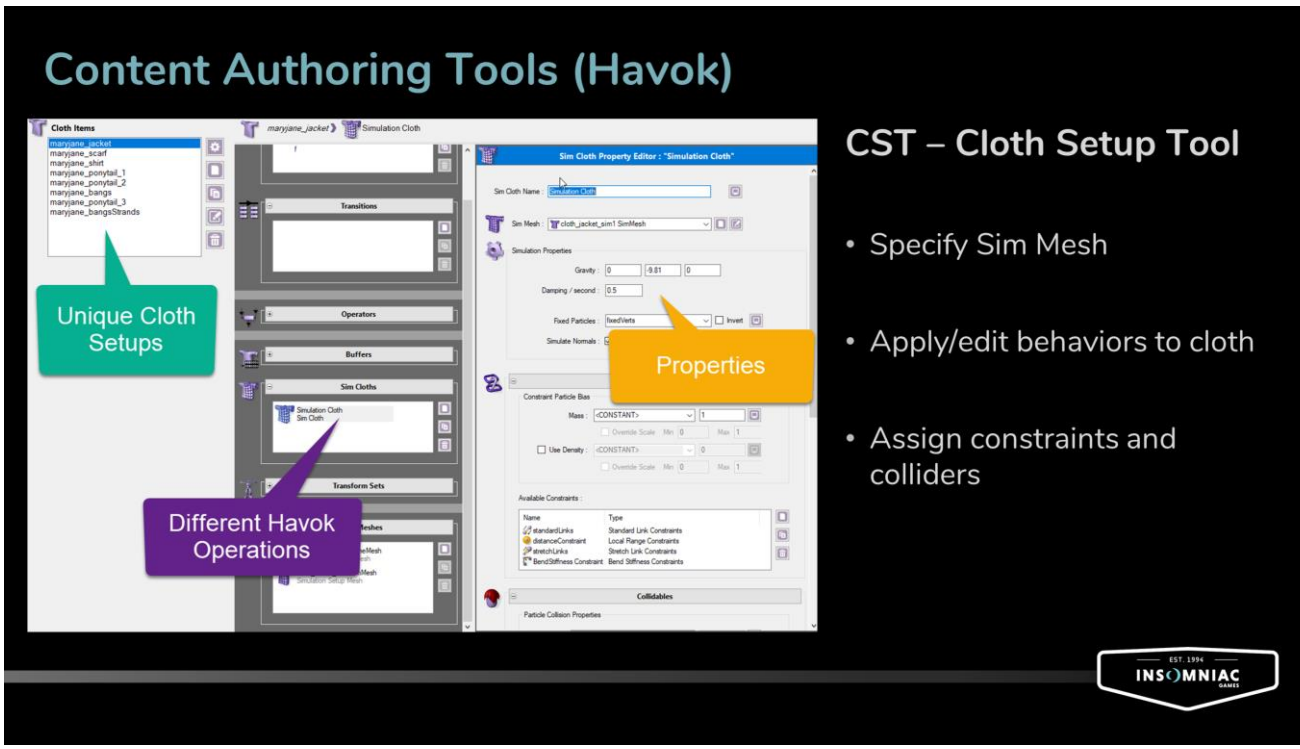
Sometimes this is necessary as Cloth needs different data types for some data sets.

To get your cloth to collide with your mesh- you need to create and assign colliders. In this video example, you can see me creating a collider and assigning it to the chest joint. From there, I can edit properties within the capsule shape itself to adjust radius, height and taper.
(You can also set if a shape is custom on the havok shape node itself.)

We've now got enough content to send it off to the Cloth Setup Tool – where we will start to modify the cloth's behavior.

From Maya, we export this data into Havok's own Filter

Manager, then into the CST. We'll skip the Filter Manager for now – just so we can talk about content we are authoring.



Mentioned briefly earlier, most of the work defining cloth behaviours occurs in the CST. There is no native API to allow you to alter cloth properties live in the game – so all cloth is authored and saved as data – then loaded into game.

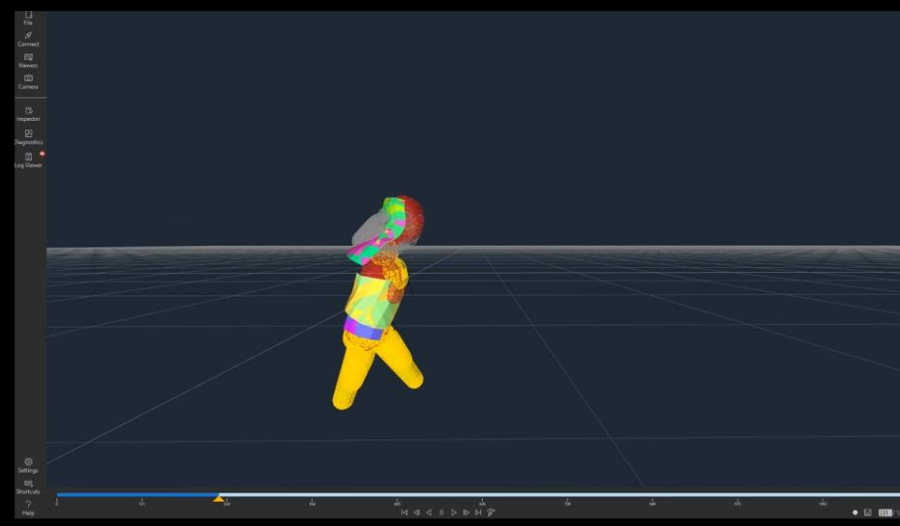
The CST is where the magic happens. It contains ways to create and assign sim meshes, apply properties and constraints and their colliders.

This can get pretty in-depth, but for the sake of brevity – know that almost all settings related to general behavior are set here, while the specifics of those behaviours are added in the Content Creation part of the process – in our case this

is Maya.

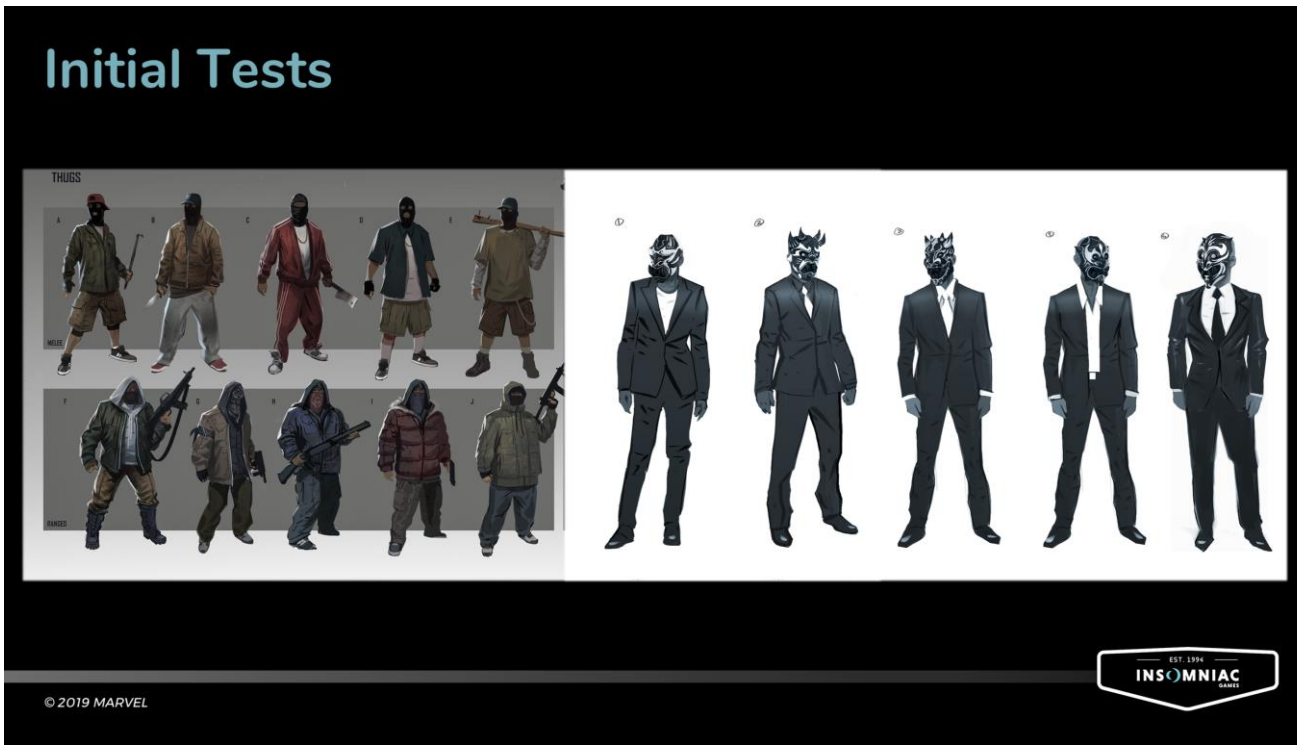
Debugging Havok (Havok Visual Debugger)

- Draws Havok Sim Colliders and Mesh
- Filter specific elements
- Check performance



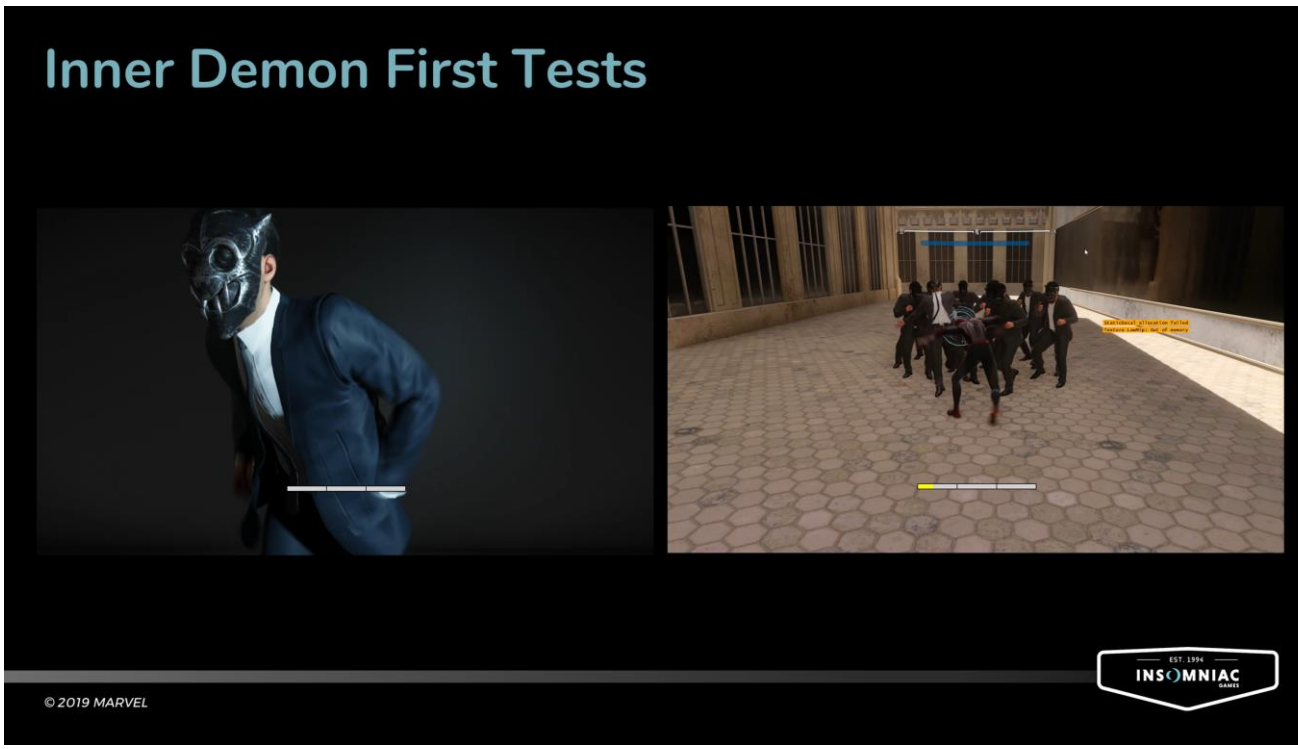
Once you have setup your sim and are viewing it in game, you can use the in-built Havok debugger. This allows us to get important information like how much resources a particular element of the setup uses, as well as visual representations of our havok simulations. This includes ragdolls, cloth sims and their properties. From here, I'll be jumping back and forth between the debugger and the content tools to iterate on my cloth setup.

With that all said – now that the pipeline is setup – let's talk about our first cloth tests for Spider-Man.



Even though we had previously used cloth on Sunset – we wanted to start testing more realistic cloth behavior that the grounded world of Spider-Man would need. Our first tests were of our Inner Demons and Thugs, shown above.

They both have very distinctive but very differently behaving clothes.
For the thugs, they needed baggy, loose pants. For the Inner Demons – smart suit jackets that held form.



We specifically used the Inner Demons as a test bed for both our animated normal maps (to show more granular cloth detail) and our real-time cloth. We got pretty good results from this, if a little floaty. This was our first major pass with the cloth system even though we would revisit these assets later. We also used to this gauge how many enemies we could draw at once using cloth (a lot!).

Through this, we wrote up some internal documentation on the process. However, the workflow was pretty complicated – with many points of potential error. This would continue to cause us a lot of problems and heartache.

This was all previous to when I joined Insomniac. For clarify, before joining the company I'd rarely touched cloth beyond the propriety cloth setup at a previous company. And even then, my exposure to it was minimal. This is perhaps a bonus, because I came in with fresh eyes.

My first real test case was MJ. This was a tough one off the bat for several reasons:

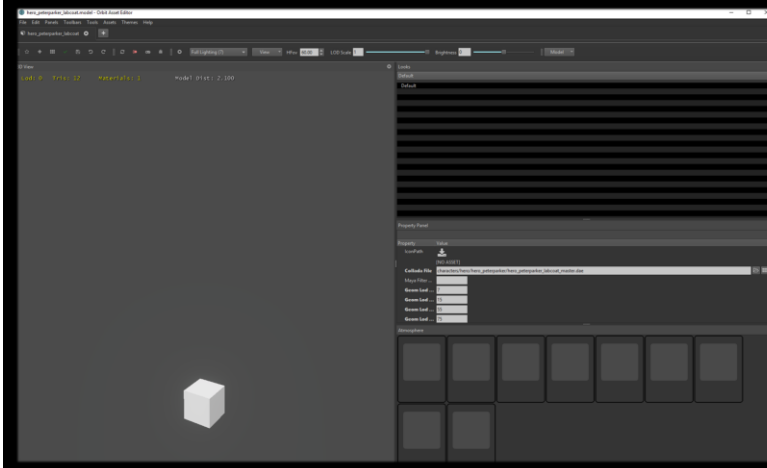
MJ had quite an extreme crouch pose that would cause the cloth to get caught between colliders.
She had two layers of clothing.
She also had a ponytail.

…

And a scarf.

We’re return to MJ later… as this was just a ‘first pass’ with the system. With some experience on MJ, I knew I needed to fix some things to get iteration speed and quality on the cloth up.

Identifying Issues With the Pipeline



- Authoring content took a long time
- Exports were failing often
- Iteration was slow

Our infamous 'white cube' indicated a failed export

Quite quickly, I was able to identify issues with the pipeline as it was.

Authoring content took a long time. There was a lot of confusion on getting assets working in engine. Figuring out where exports were failing was tricky.

Even past that, iteration was slow. Bugs on the cloth simulation were very difficult for us to isolate. On top of that, we were using the system in ways our engine did not anticipate.

There also was the issue of debugging cloth. This also complicated things.

Firstly, we were unable to see the cloth draw on top of our game mesh in engine. This made it difficult to pinpoint what belonged to what specifically. Especially with our more layered systems and multiple colliders.

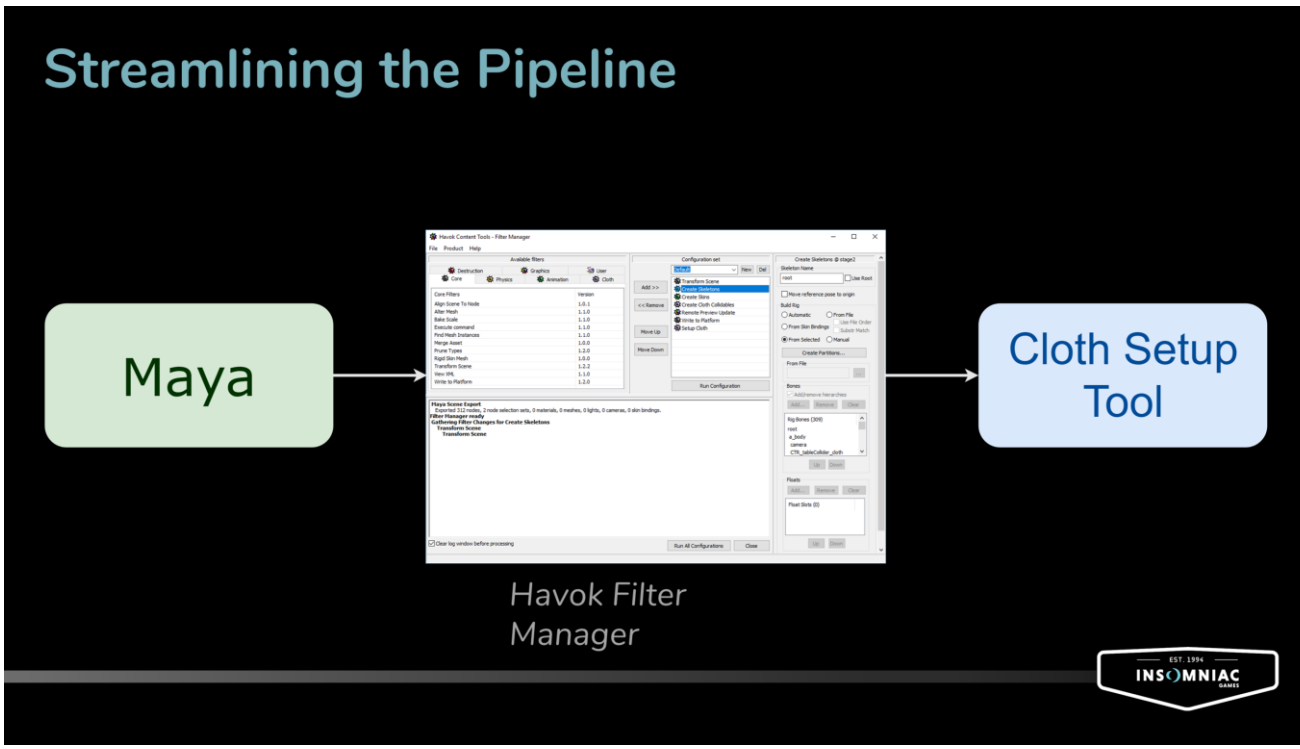
We also lacked debug information. As previously mentioned – elements were laid on top of each other without labelling.

Which led to our next issue – the information that was there was hard to parse. Without any filtering, we couldn't isolate

bits of the setup – which is usually the best method of solving problems.

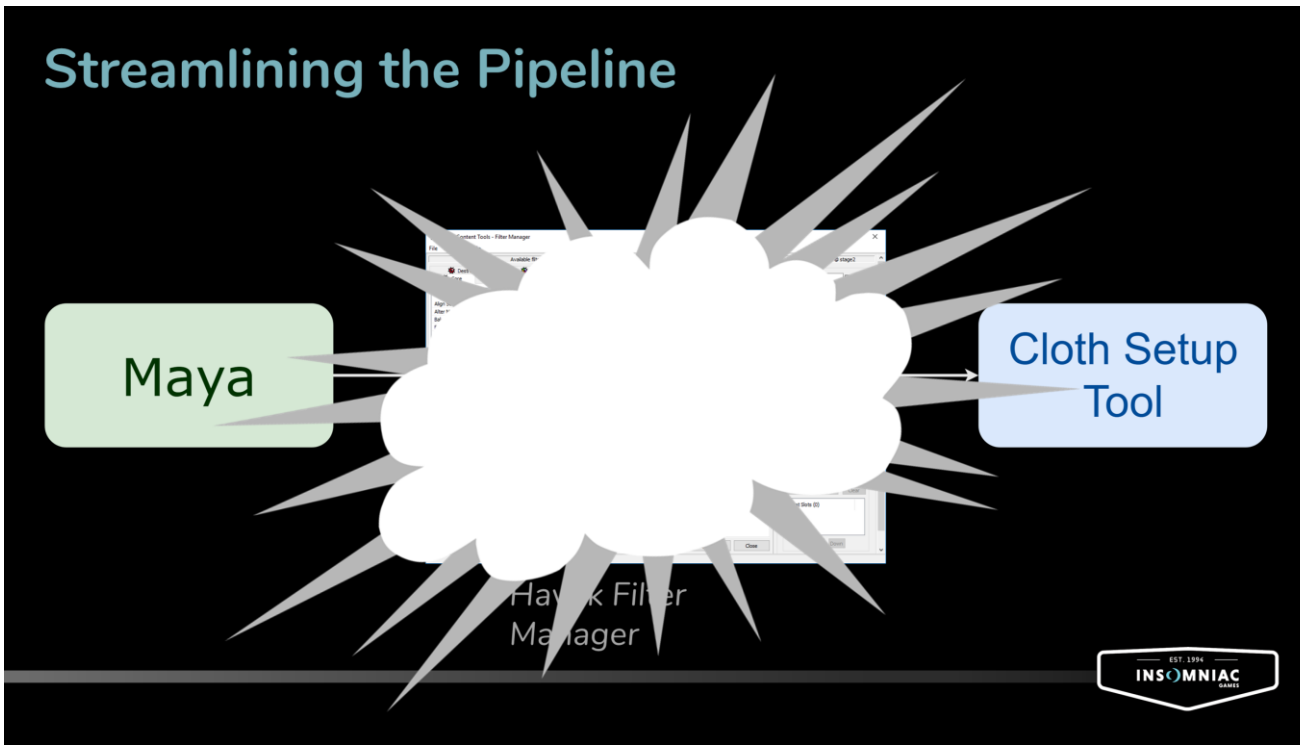
Here’s a visual example:

As you can see…. There are a lot of elements being displayed on the right and it’s quite difficult to tell what is what. When we were debugging – we would have to remove elements bit by bit to see where an issue might stem from – rather than entirely rely on the visual display.



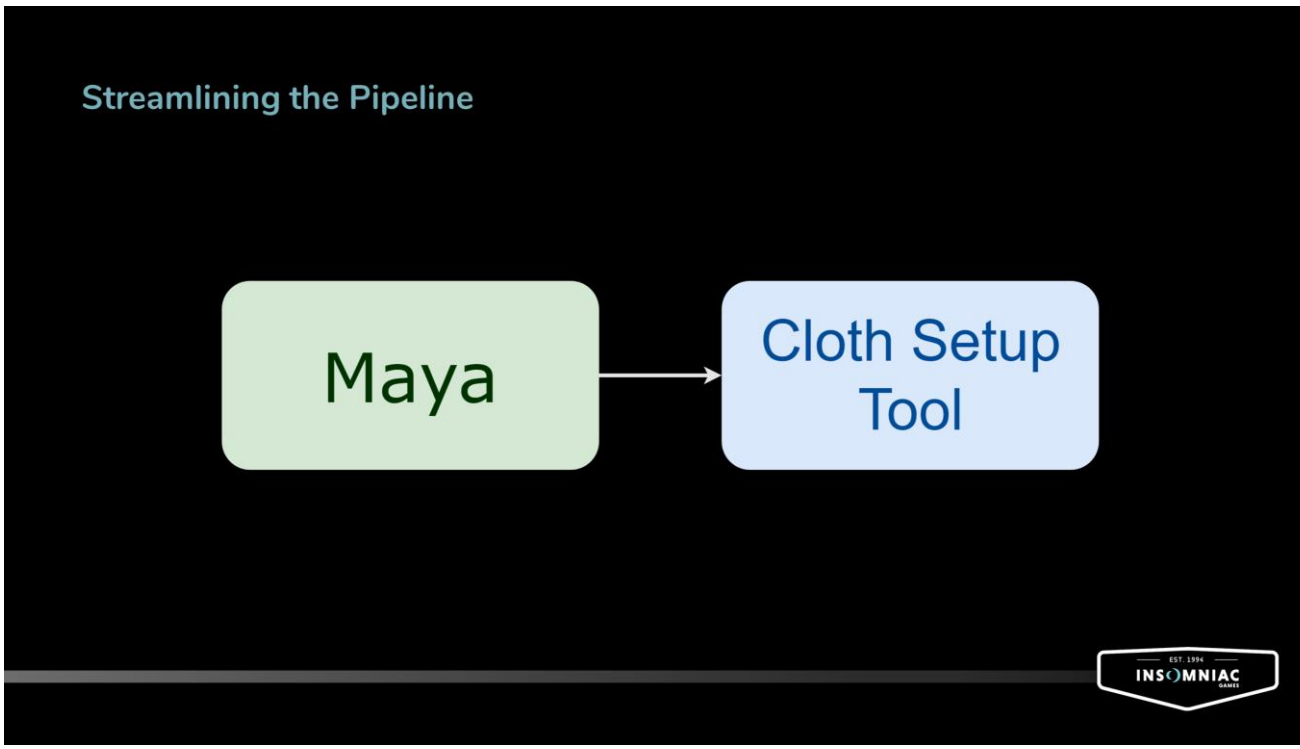
To combat this; first thing to be addressed was anything that could be automated. There were a lot of different windows and many repetitive actions.

My first goal was to remove the Filter Manager. While the Filter Manager is extremely flexible and powerful – we only really needed a generic singular cloth setup for it. I was able to create a template that was generic enough to work with our naming convention.

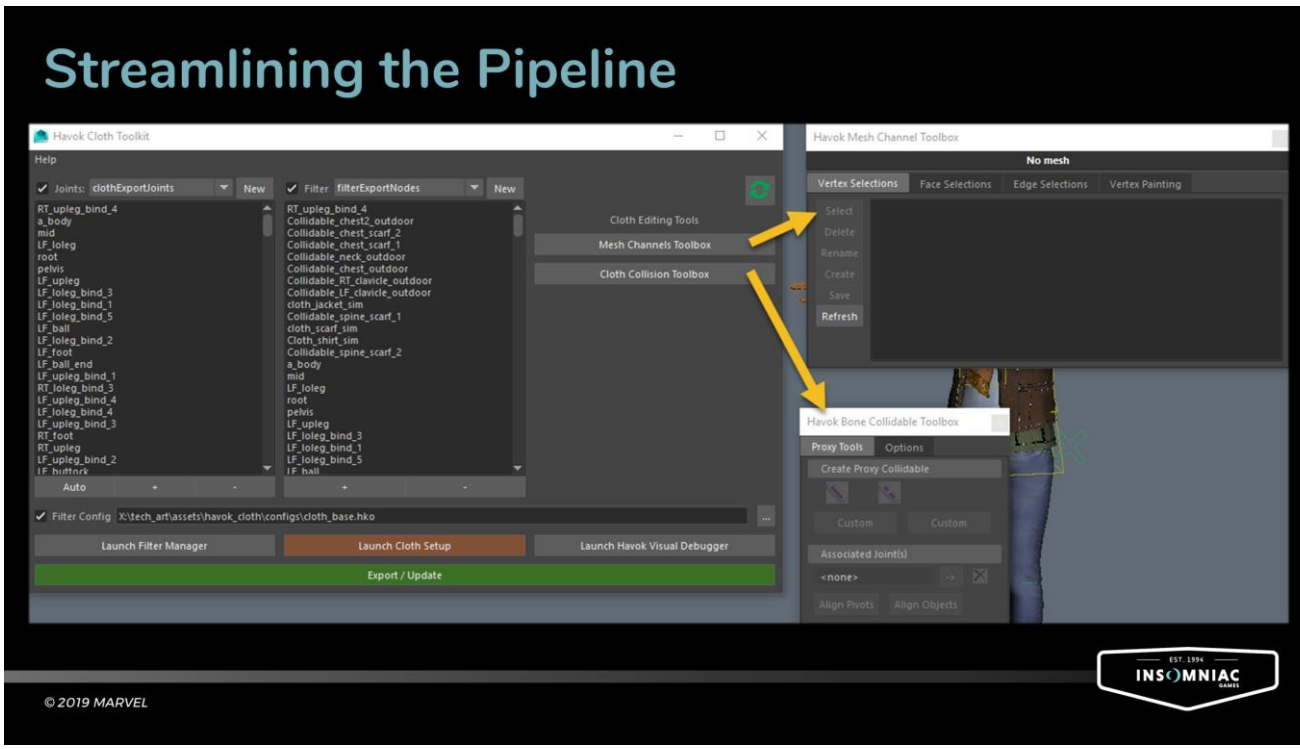


So this removed one additional window and step from getting our content from Maya into game.

BOOM!

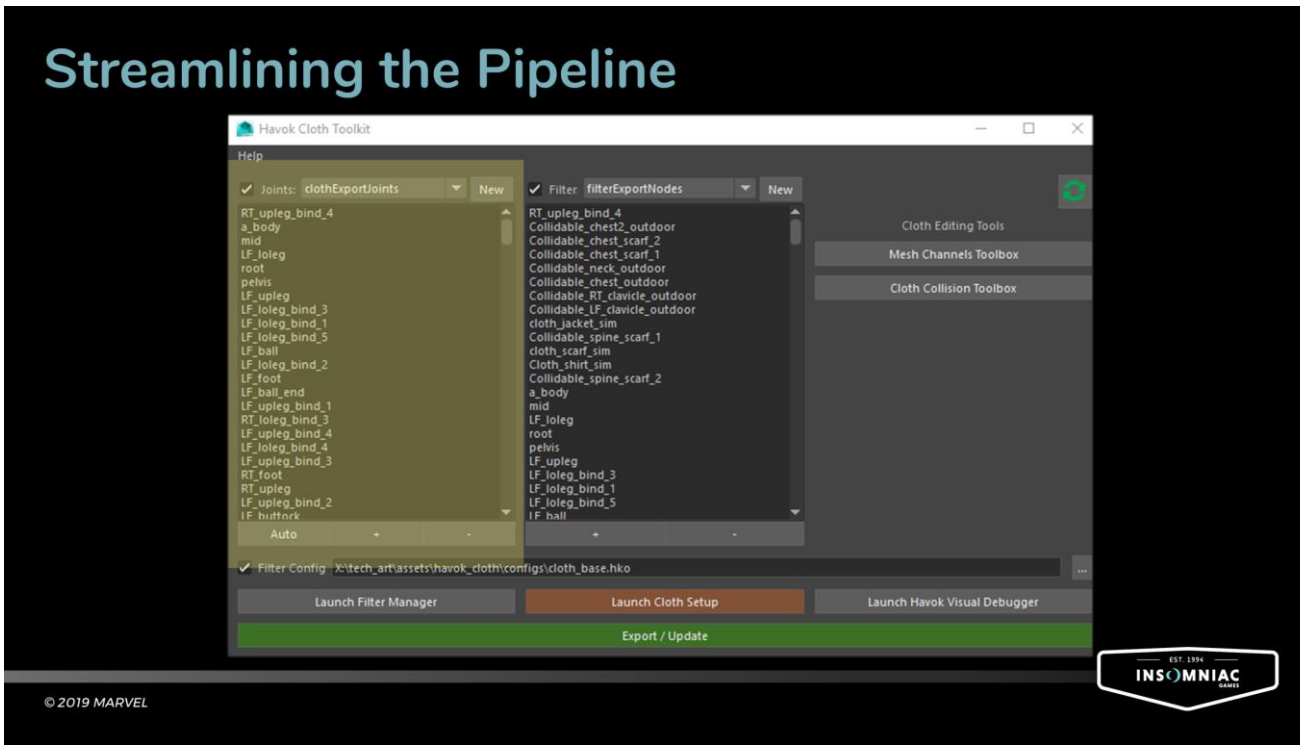


It then meant our content went direct into the Cloth Setup Tool. So we moved from one content authoring system to another.



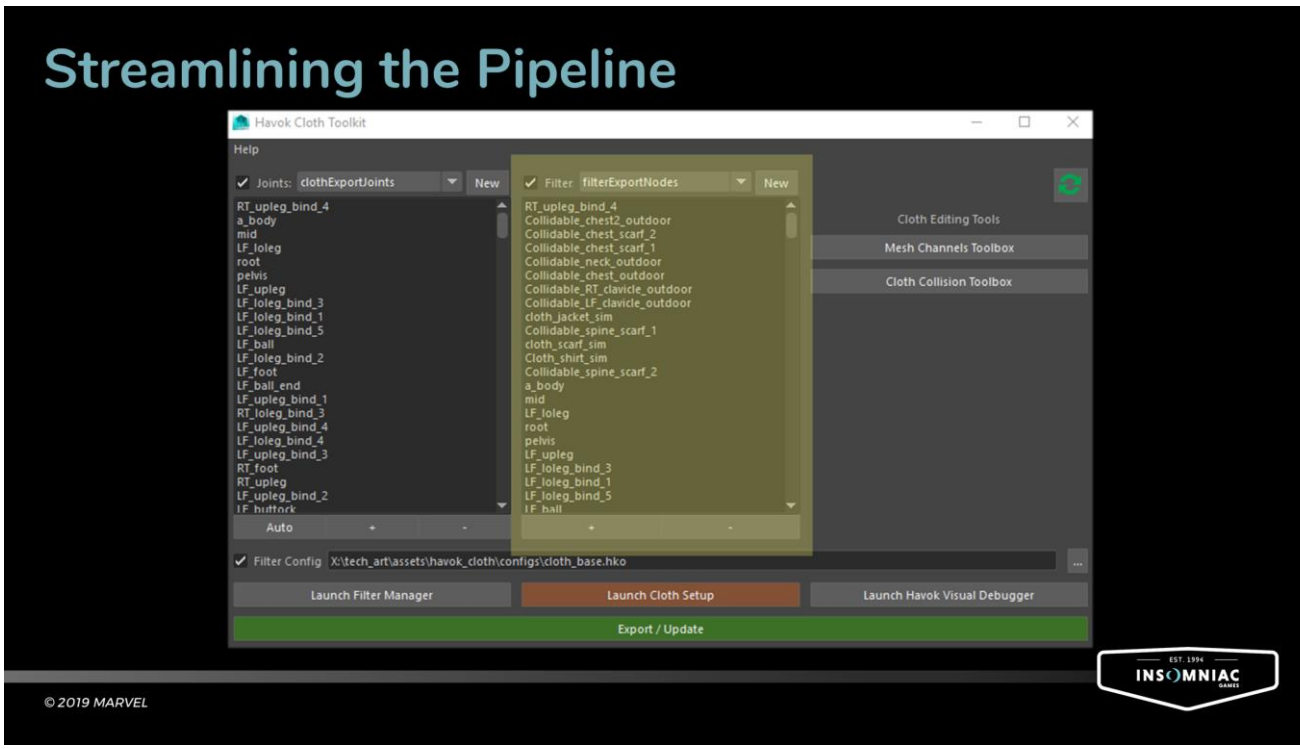
Next, I created a UI.

A UI allowed us not to trip up over the simpler arbitrary stuff. There was a workflow that allowed people to let people know what needed to be defined and if they were missing something.



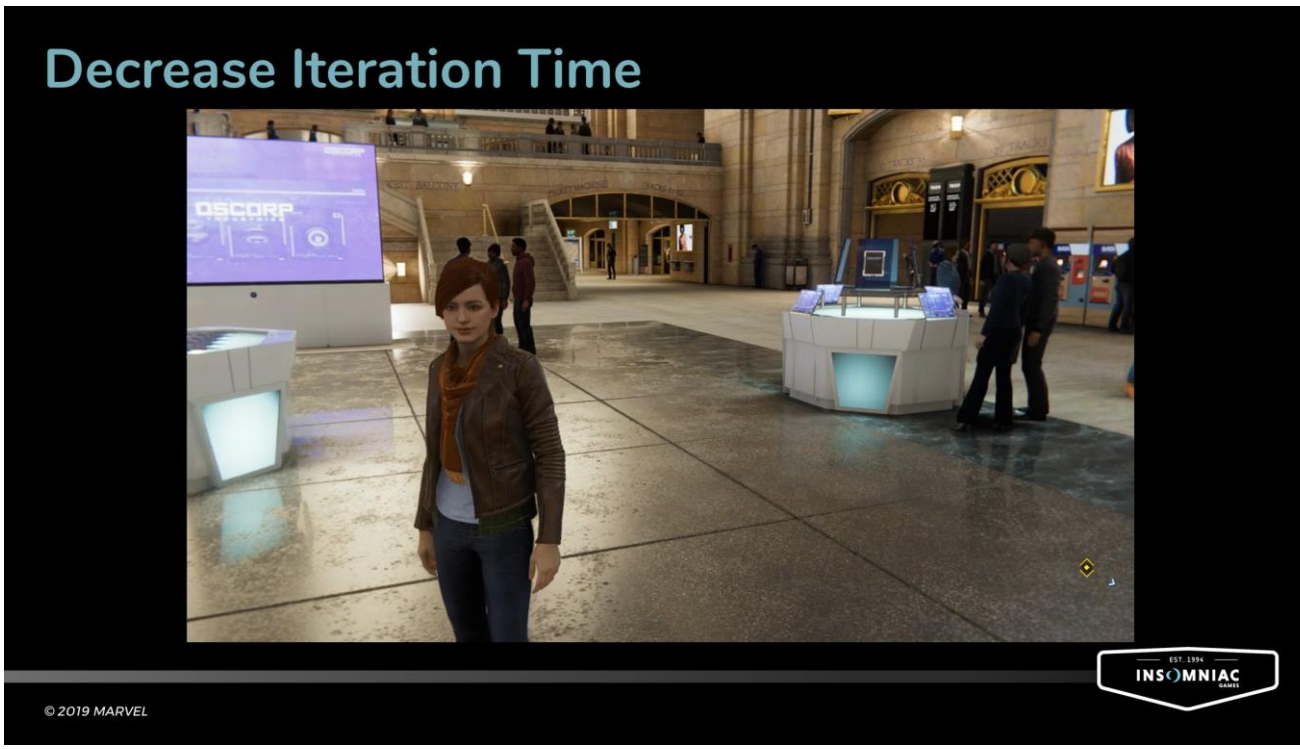
On the left here – was our list of cloth joints that would be passed into the CST.

We found the most reliable way to get only the necessary joints into the Cloth Setup Tool was to make a selection set in the scene that was selected upon export. Havok had a filter that only exported selected objects as joints in the skeleton filter within the Filter Manager.



Havok also had a habit of exporting every node in the scene. For quick iteration on large files with lots of blendshapes this made things very slow. So we created a filter that would only export the nodes in the list. This is the list on the right seen here.

(Unfortunately, Havok doesn't currently have any way of passing a list of nodes to the filter… so we had to HIDE and then UNHIDE every node in the scene to do this… pretty nasty business considering the number of ways Maya can 'hide' a DAG node. This also added time to our export process but it was worth it for savings on not iterating on every node).



Another one of our challenges is a lot of cloth was on playable characters. Since this was all dynamic, we needed to test the cloth while moving the player character to be able to test the motion on it.

Since all the content had to be authored before export, we needed the iteration time on this to be as small as possible. Changing one small value caused an asset rebuilt and if that meant reloading the game every-time it was going to be hard to create quality assets within reasonable time-frames. We needed the updates to be live.

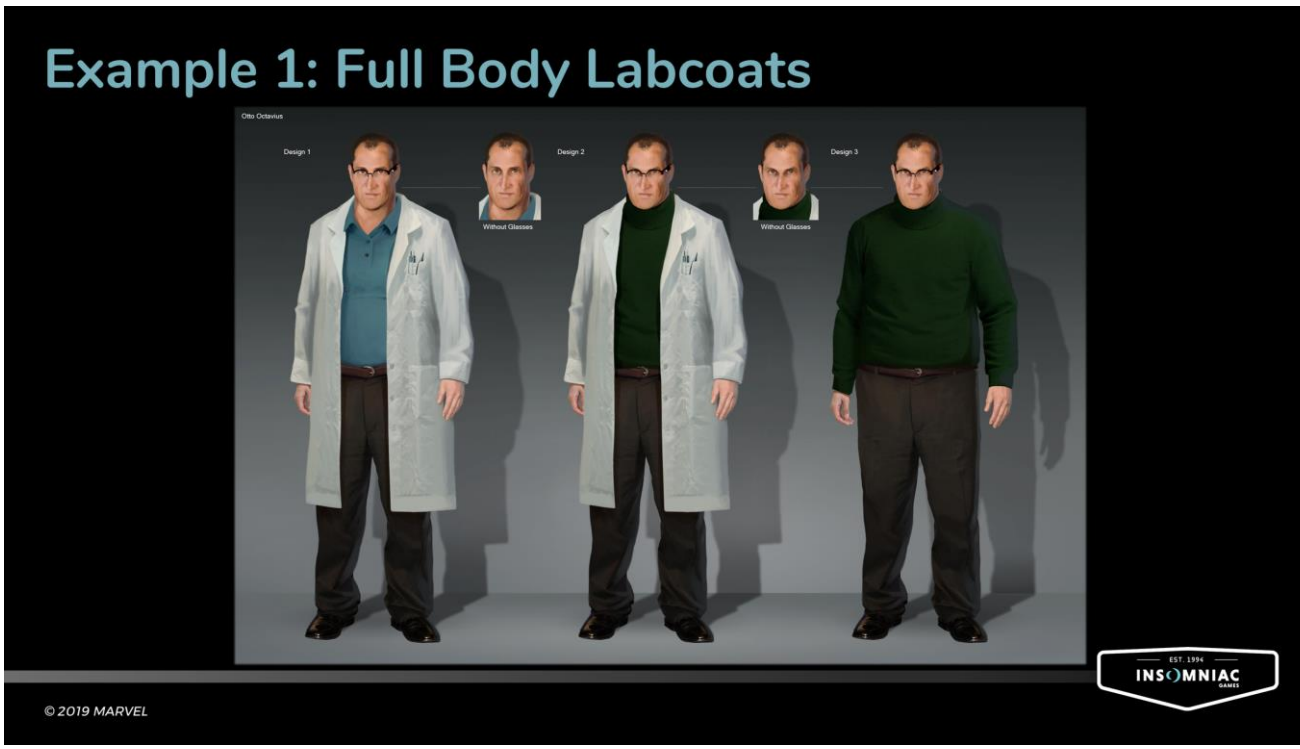
So we worked with our Core department to make sure we

could do this at run-time without massive reloads.

Here's a video showing how our hot-swapping worked. Having this work cut iteration time massively. Look at how I'm increasing the radius on the collision for the outer jacket here and how quickly it goes into game.

The combination of live hotswapping and a reworked pipeline and Maya interface meant we could reduce updating assets to as few clicks as possible.

So now that we made some improvement to our pipeline – let's talk specific cloth examples.

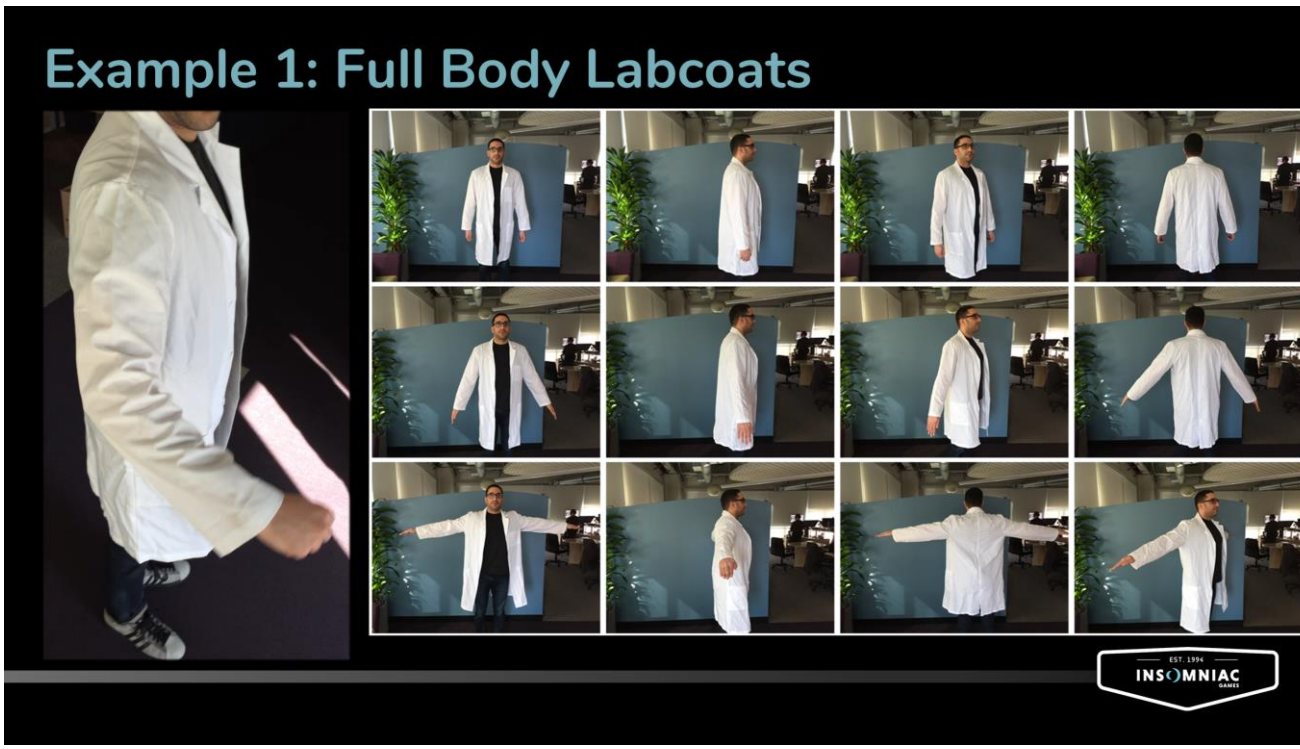


One of our first challenges was to create full-body labcoats for Peter and Otto. This was for the pivotal scene where Peter and Otto finally develop a working prototype for their prosthetics research. We needed this scene to showcase what we could do with the system, so we knew the quality bar was high.

You meet Otto at the very beginning of the game and we knew Peter would be playable in his labcoat – so we had the challenge of creating real-time cloth for both cinematics and gameplay in this scenario.

To reduce complexity, I wanted to be able to share the setup

between both gameplay and cinematics. In hindsight, while this was easier on us for implementation – it made for quite the challenge, as the needs of the each setup could be quite different…

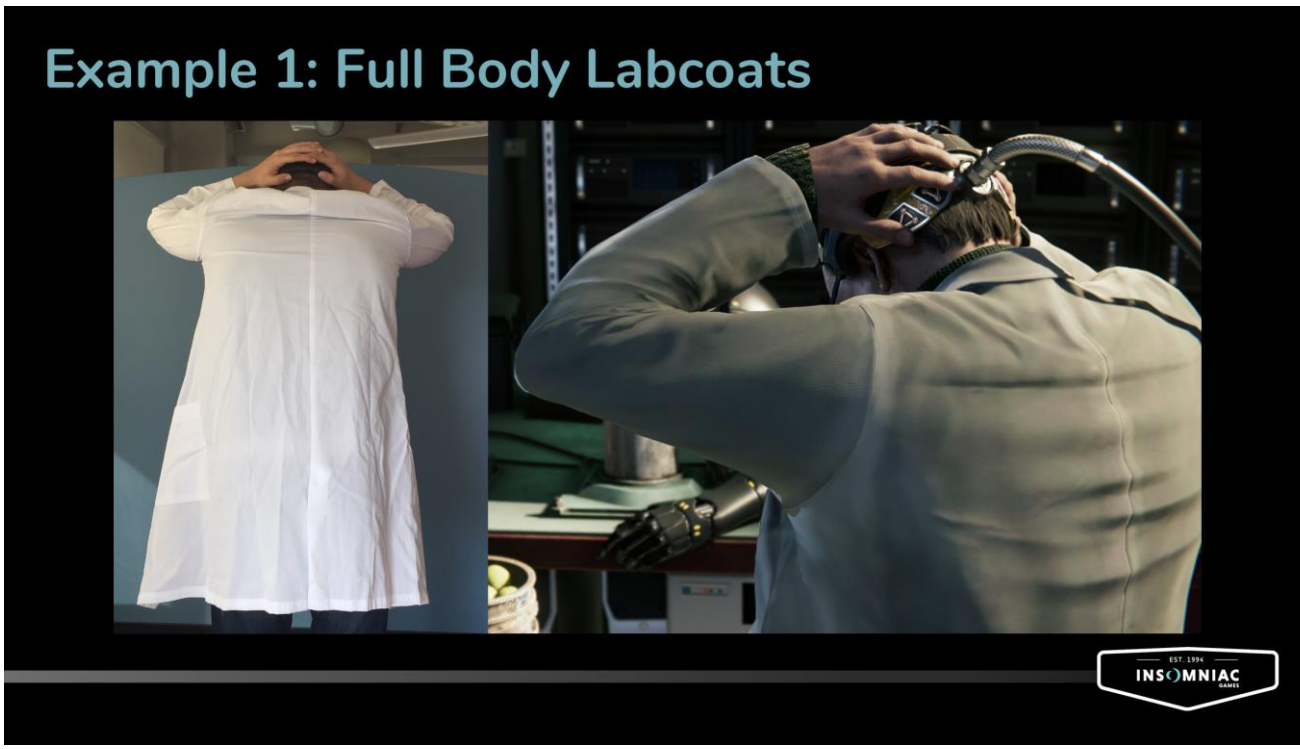


To make sure we got the look right we had to shoot reference. We looked at various labcoats (both official and costumes) and we ended up buying one we thought looked closet to the concept. We recorded walking and general movement to understand how they would flow. (Here you can see Bill, our fellow TD, wearing the coat as it was too big for me and I also wasn't very close to our targets).

This gave us a good starting point (though we ended up going with a heavier cloth).

We also specific shot references. We knew what poses we were going to hit from our mocap and had a couple of

trickier ones we knew that the cloth sim might not be able to hit.

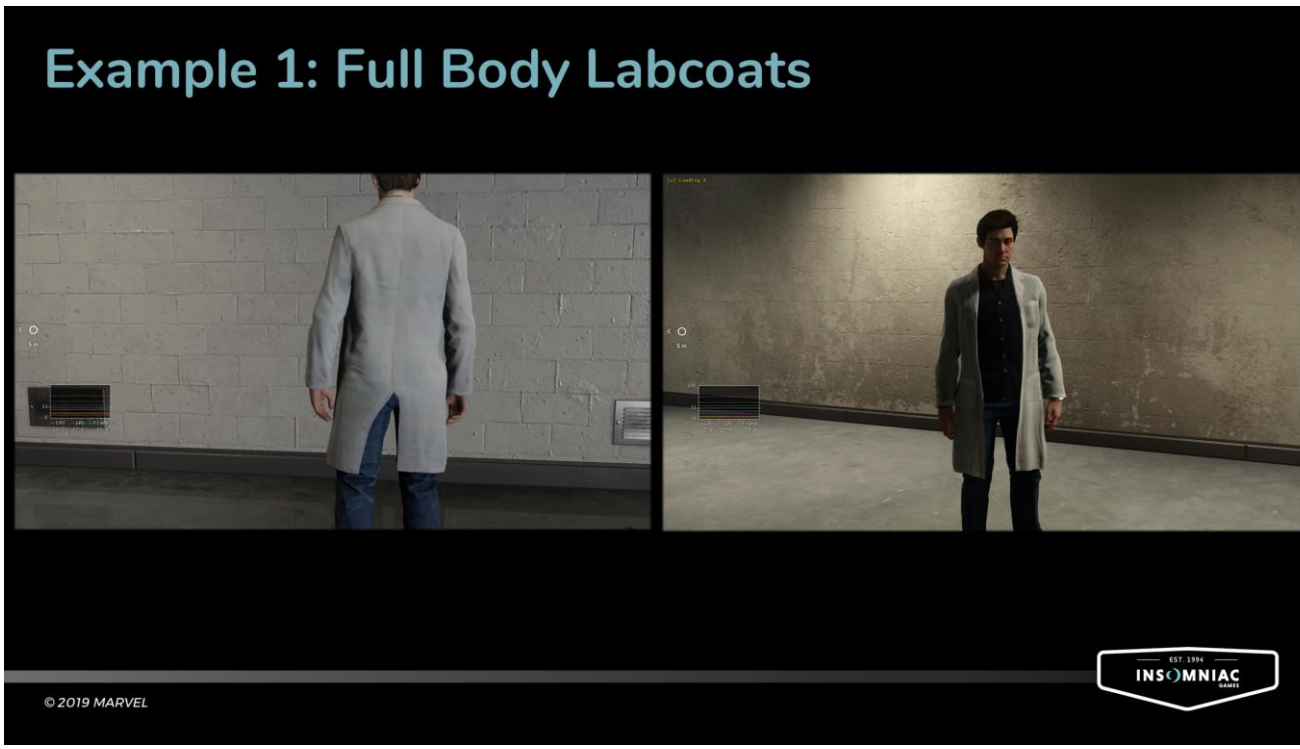


We also specific shot references. We knew what poses we were going to hit from our mocap and had a couple of trickier ones we knew that the cloth sim might not be able to hit.

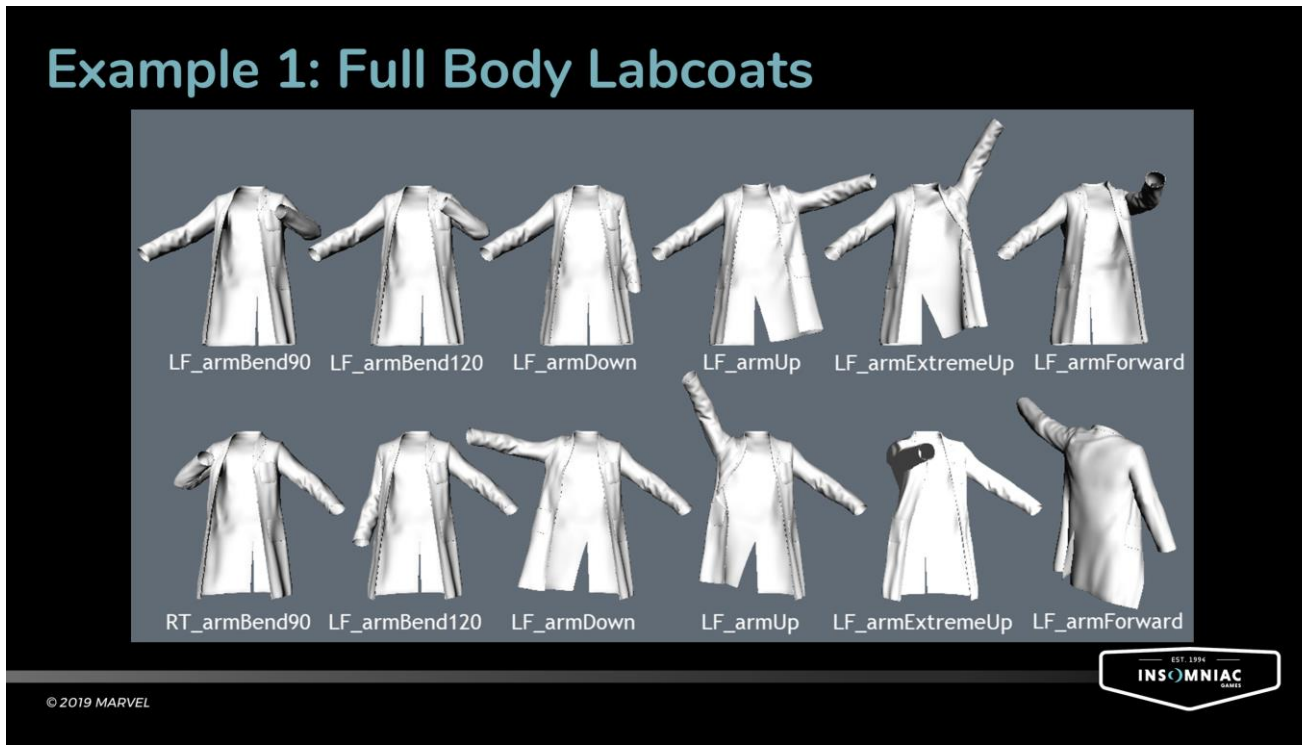
1/2

We also specific shot references. We knew what poses we were going to hit from our mocap and had a couple of trickier ones we knew that the cloth sim might not be able to hit.

2/2

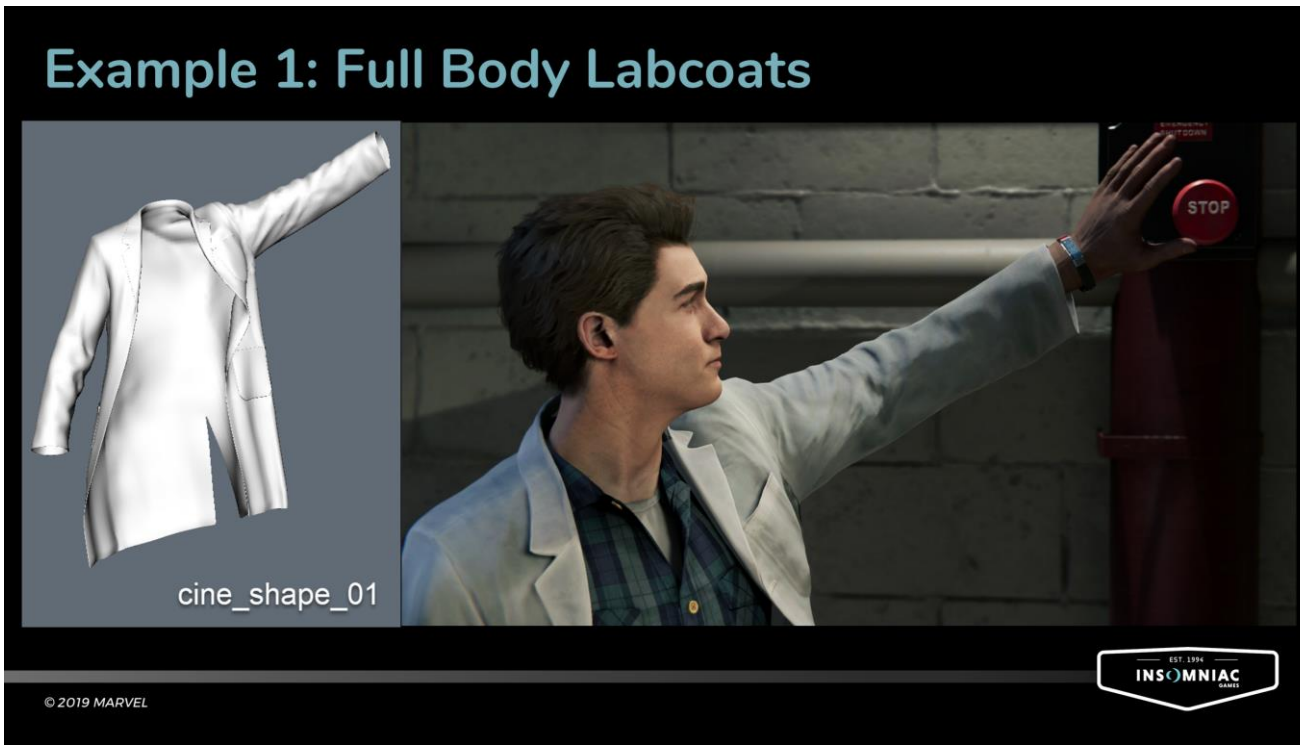


We simulated the sleeves and the coat all the way from the pectorals down. Here you can see some very early screenshots of the setup. As you can see – the cloth sim alone wasn't enough. We needed to do more to hit our high quality bar.

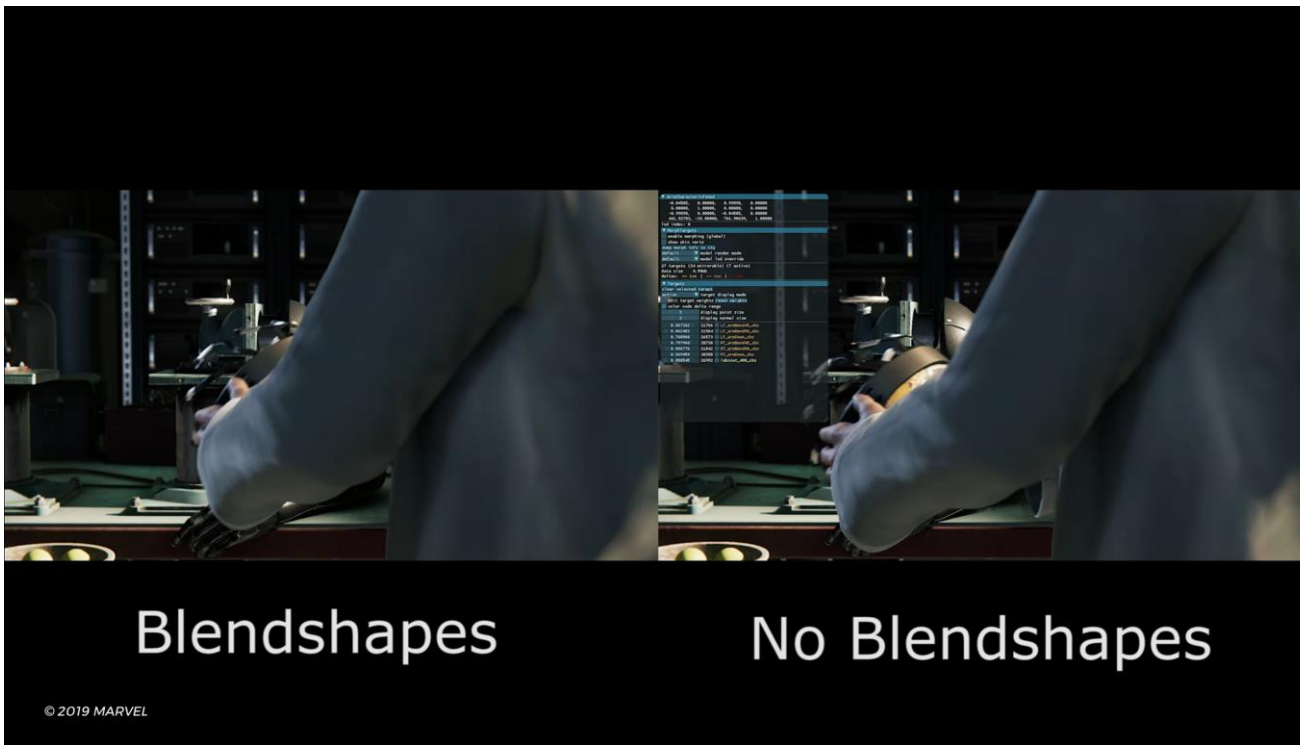


To improve the quality we needed to improve the overall shapes of the labcoats. As mentioned earlier, there were some very specific shapes we needed to achieve the look and feel, which weren't going to be possible with the real-time sim.

We developed two types of corrective shapes.

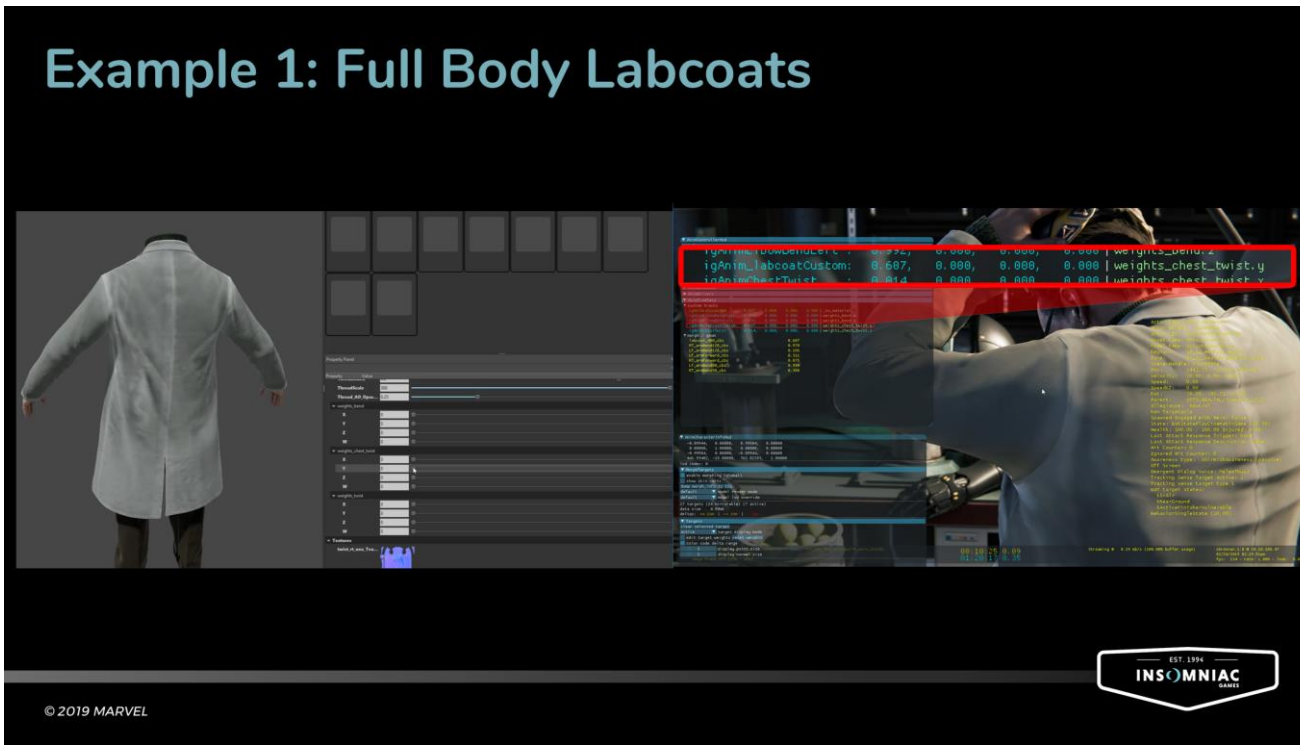


Lastly, you can see we had the specific cinematic blendshapes we would toggle for particularly specific shots.



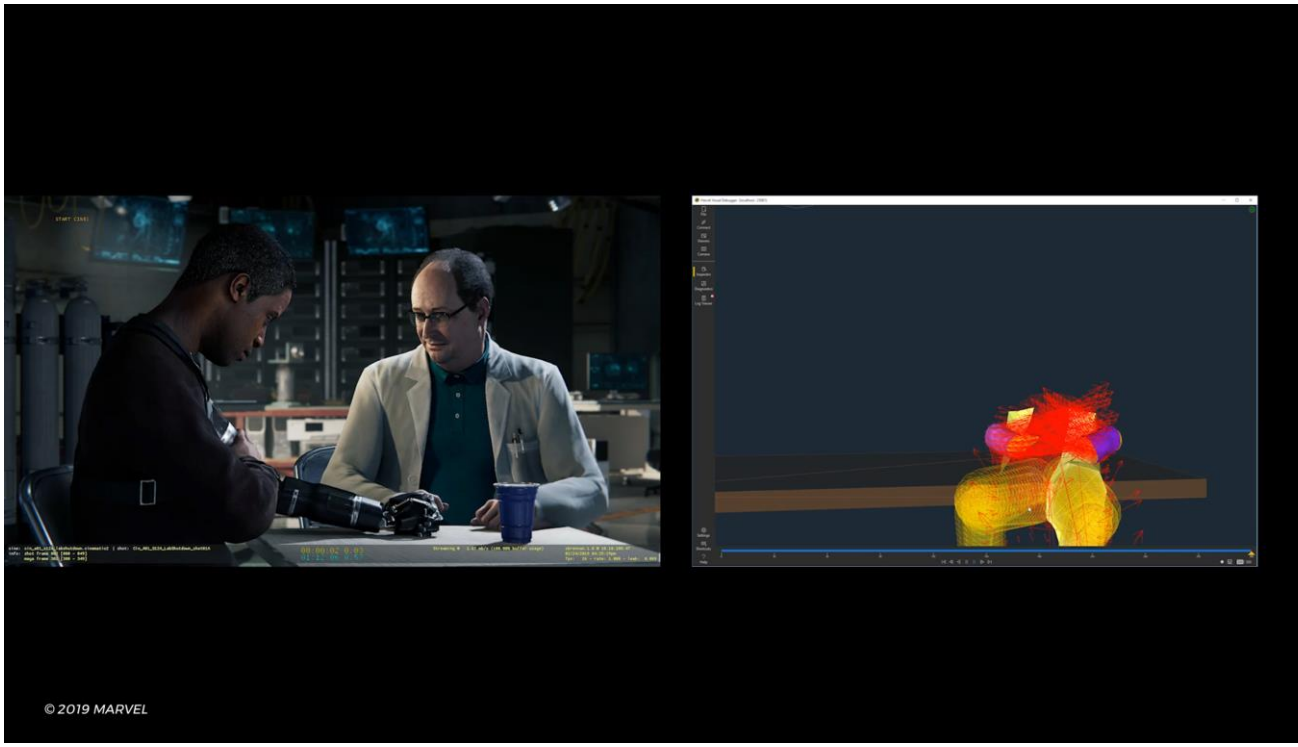
Here’s a small example of what these blendshapes looked like in action and where and how we used them. As you see, the use of blendshapes here really upped the quality of the simulation! Especially in parts where the cloth is lifting or dangling in a specific way.

You can see with blendshapes on the left, and without on the right. On the right side, you will also see our in-engine Morph Debug UI that would tell us when shapes were triggering (the orange and yellow entries).



For a finishing touch – we weren't going to get enough detail from blendshapes OR cloth. We had already doubled the resolution of our labcoats to support the blendshape silhouettes we needed to hit but it still wasn't cutting it. We made use of animated normal maps to give us the wrinkle detail we needed.

We also had a spare channel that we used for specific cinematic targets. We would swap this channel with our cinematic normal map, which was a full body replace normal.



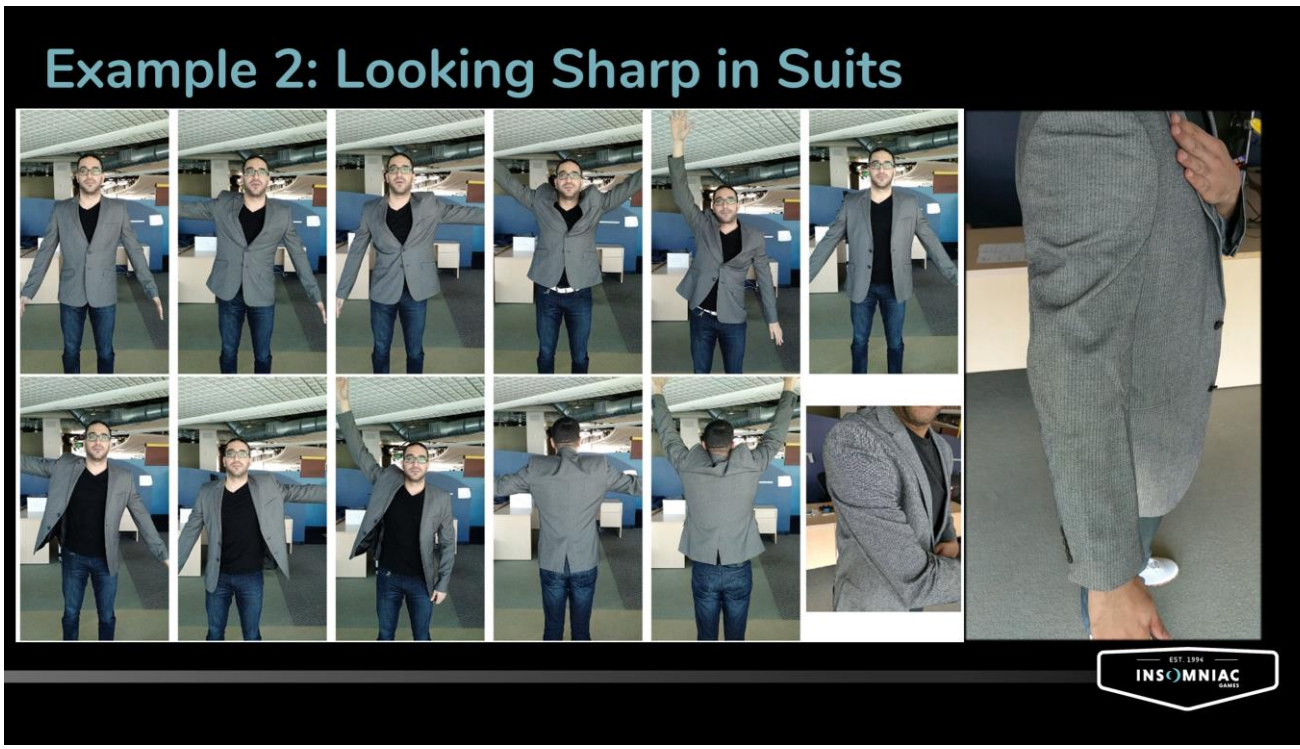
But wait! One final ‘hack’ we had to introduce was that our cloth did not support collision in the environment, so in many of our rigs we would include a ‘cloth collision’ plane or object that would act as a ground plain or table, etc and could be animated into the scene when needed. Our vanity system also worked with cloth and cloth collision (this is something we had to get our core team to implement as we needed to switch collision and cloth setups within assets frequently), so we would have this ‘collision object’ on a separate layer that we could toggle on when we needed, rather than making sure it wasn’t getting in the way in any scenes.

Things like the sleeves would get ‘pinned’ more often than not… so it was safer to let them ‘fall through’ the table in some places – as it often wasn’t super visible.

Let’s move on to another example…

Mr. Negative was another one of our big villains in the game – and we needed him to look the part. He was smooth dressed and clean-cut and that needed to show in his suits. We also had suits being used for all his thugs, as well as Norman Osborn.

A bit part for the suits to look more like suits where for them to lift up like suits do and hold their cut. Well-tailored suits hold their form well and we wanted this to show in game, even for the thugs who are in more ‘sports’ suit jackets.



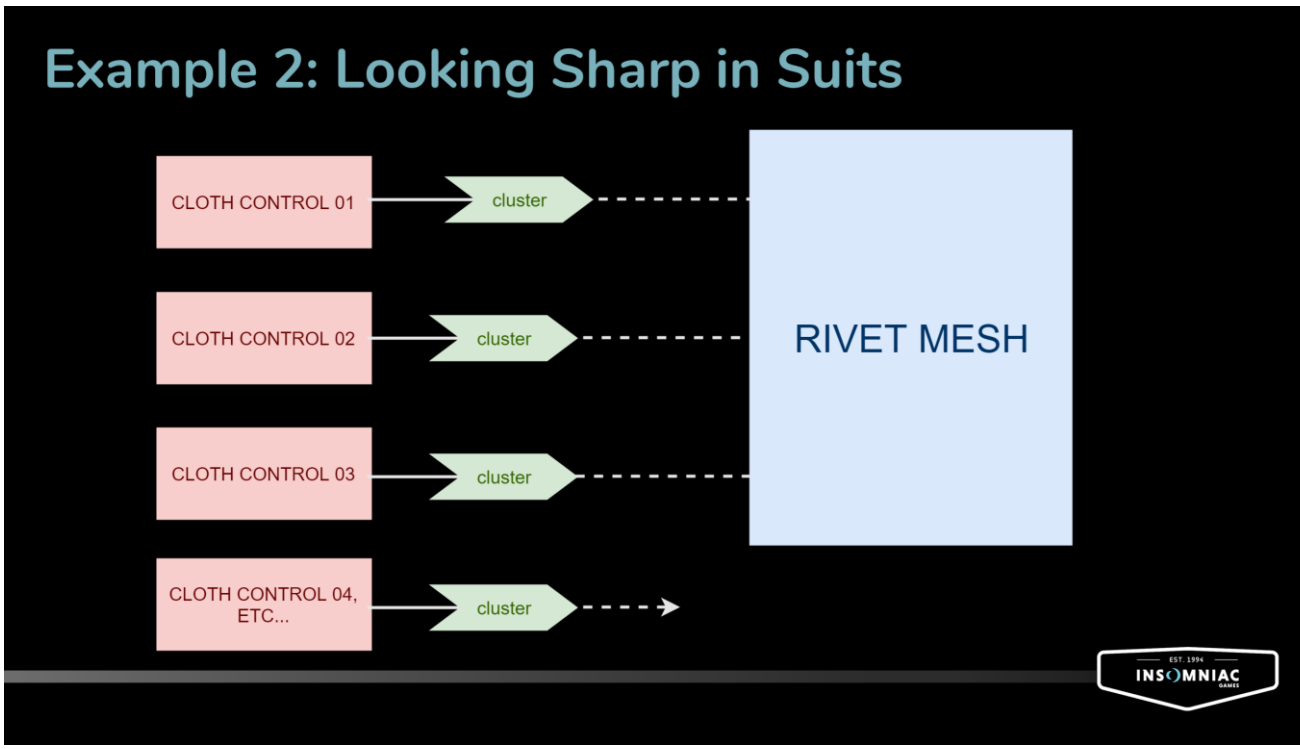
Once again, we captured reference with a sports suit jacket. We captured mostly shoulder shapes, but were also paying attention to how the suit lifted when the arms moved, and how the shoulders bunched.

Armed with all this reference, we knew that getting the jacket to work as required was going to need a combination of two techniques. Real-time cloth and blendshapes.

The first was the get the cloth behaving and feeling realistic in game. This was mainly for the movement. The first big challenge for this was our Mr Negative subway boss fight. Since this was a one-on-one boss fight with a fairly close camera, we had to make sure his suit jacket looked right for the sequence. There were also a handful of cinematic sequences were we wanted to nail a particular look. Above is a little before and after of some of the iteration on the suit jacket in one of these scenes.

We used a lot of finely tuned blendshapes for this, as well as a system on the rig that effectively ran blendshape animation on the sim mesh.

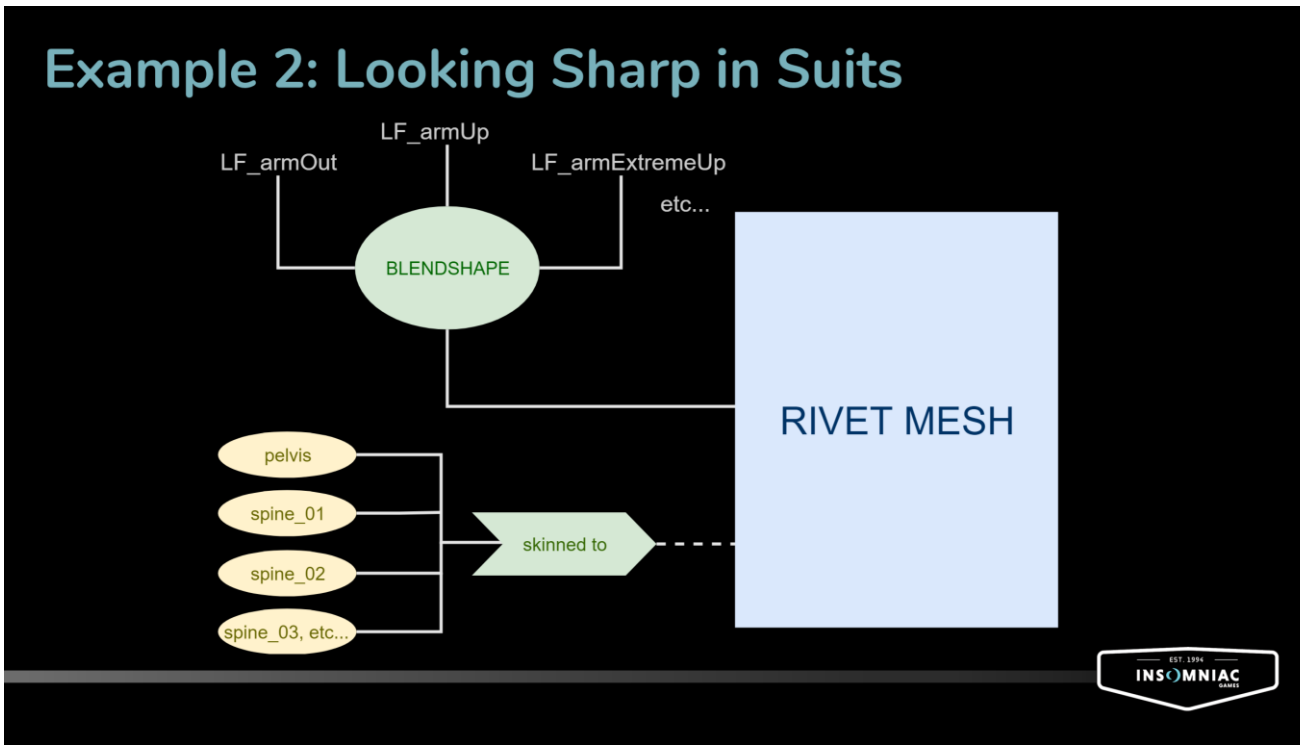
We simulated a lot of these blendshapes first in Marvelous Designer, then sculpted them iteratively to hit the shapes we needed. This was a long process with a lot of back and forth. Luckily, unlike the labcoats, the suits were mostly black or had effects on them, which meant that doing blendshapes on them was a lot more forgiving overall.



Firstly, we would have our ‘rivet’ mesh. This was a direct duplicate of our sim mesh.

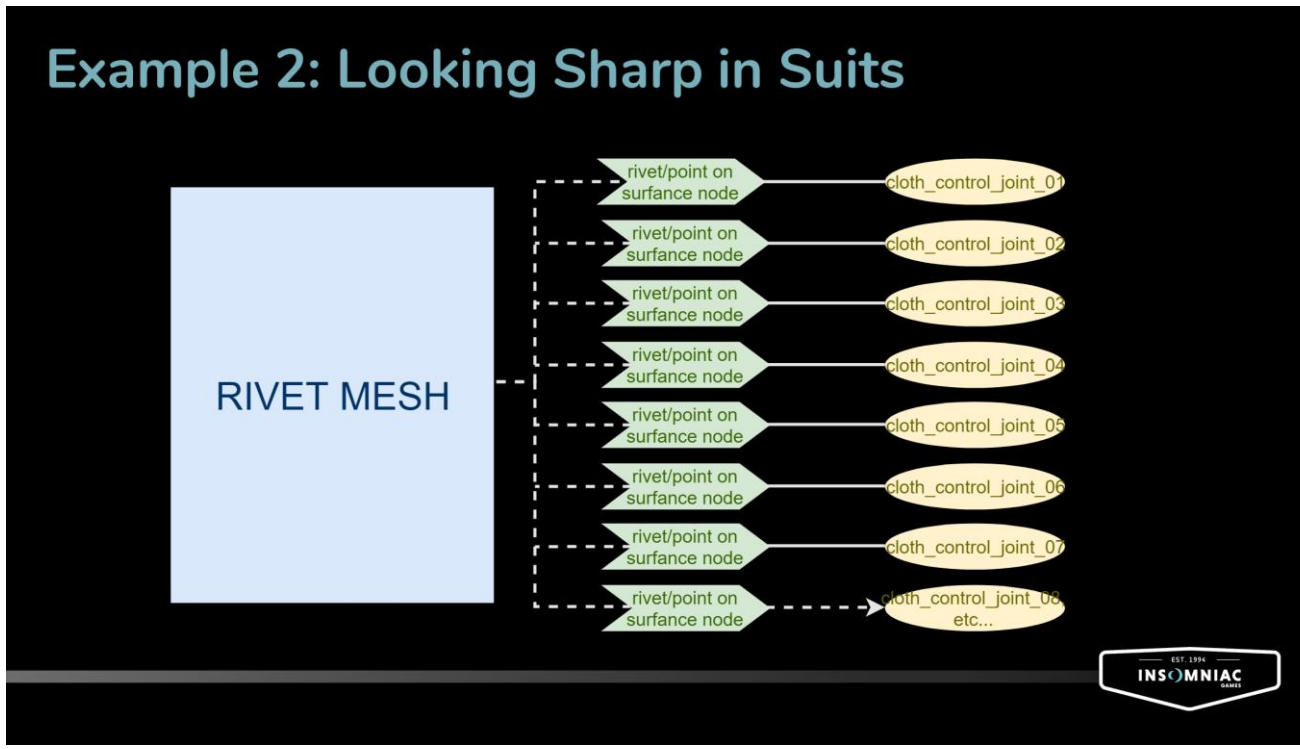
We would also have a series of cloth control joints. These were single joints at the bottom of the mesh that animators could use to pose out the mesh.

These cloth controls were, in turn, driving the rivet mesh with clusters. These clusters were weighted in a soft gradient towards from bottom to top…

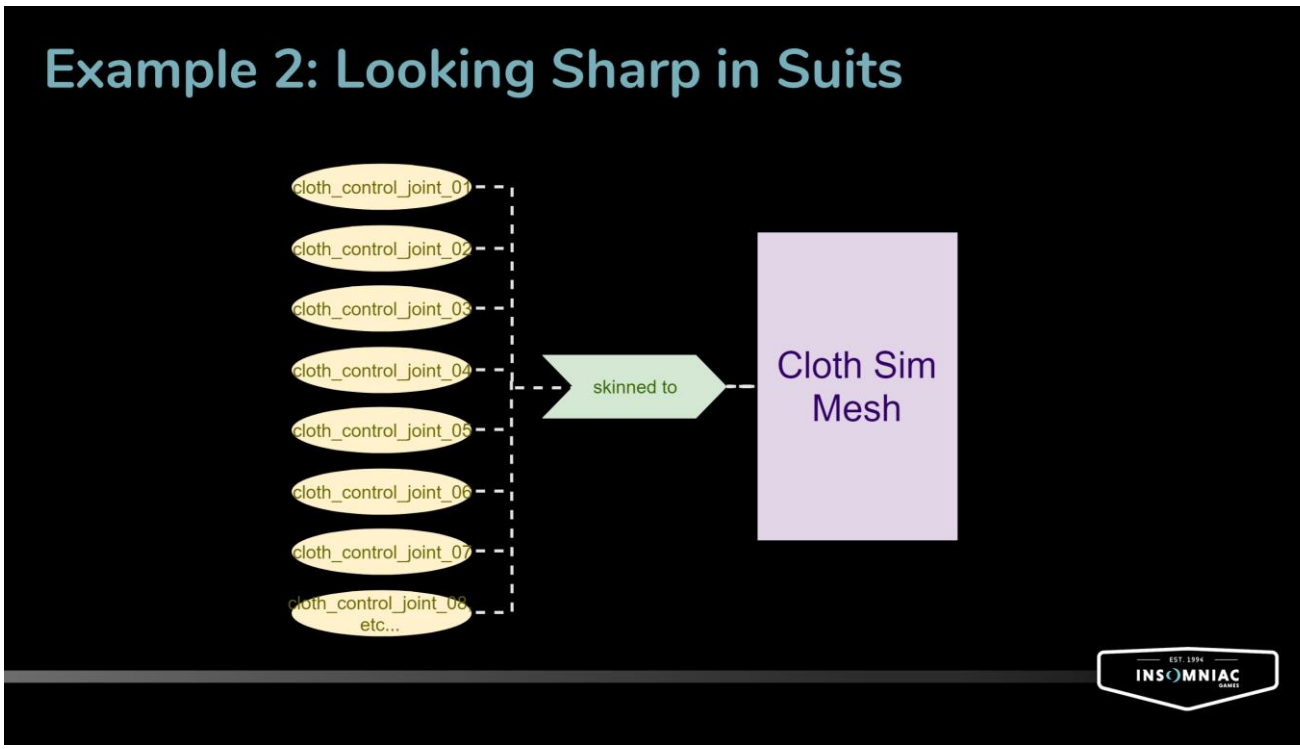


On top of that the cloth controls (which were additive), we had the rivet mesh being driven by a series of core skinned joints such as the pelvis and spine joints. This got us all our major movement.

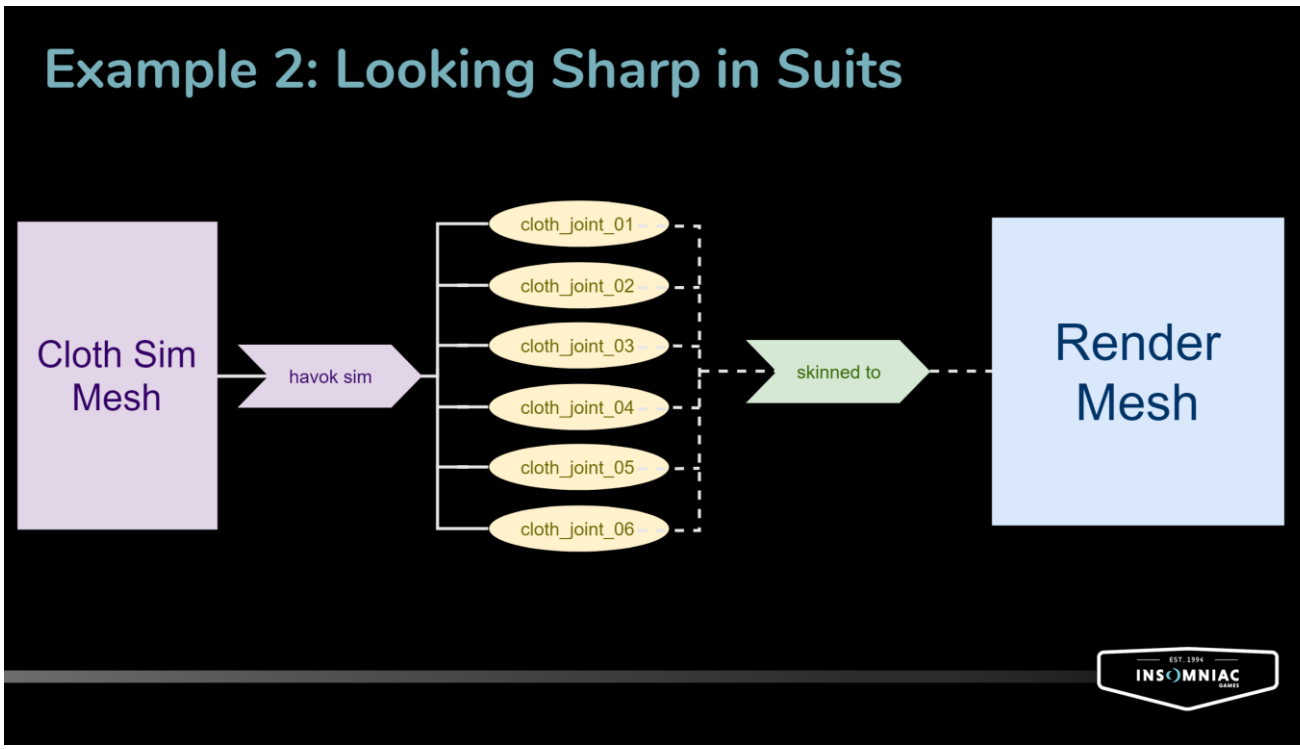
Remember our blendshpes? Those blendshapes were wrap deformed to first match the rivet mesh, and then were applied and directly hooked up to be driven by the rig deformation controls.



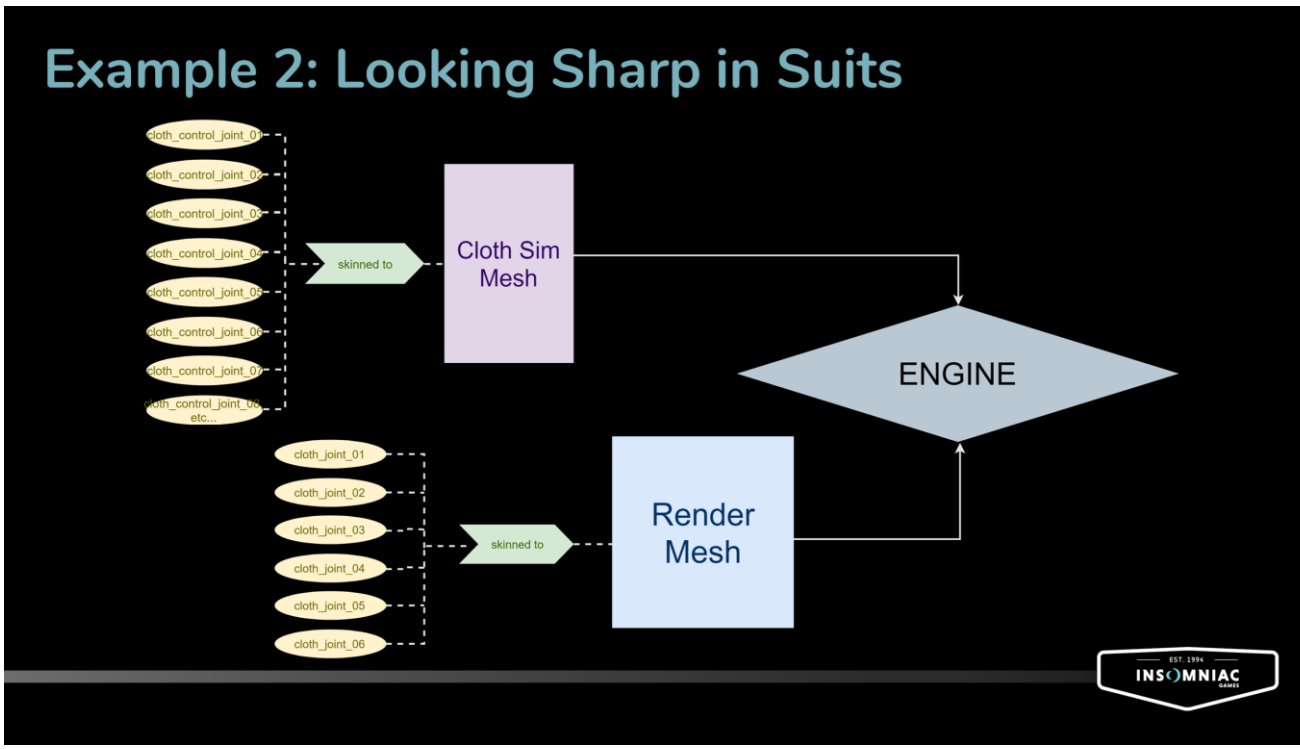
This rivet mesh is called as such as it has a bunch of joints that it drives via surface constraints like point on surface or follicles.



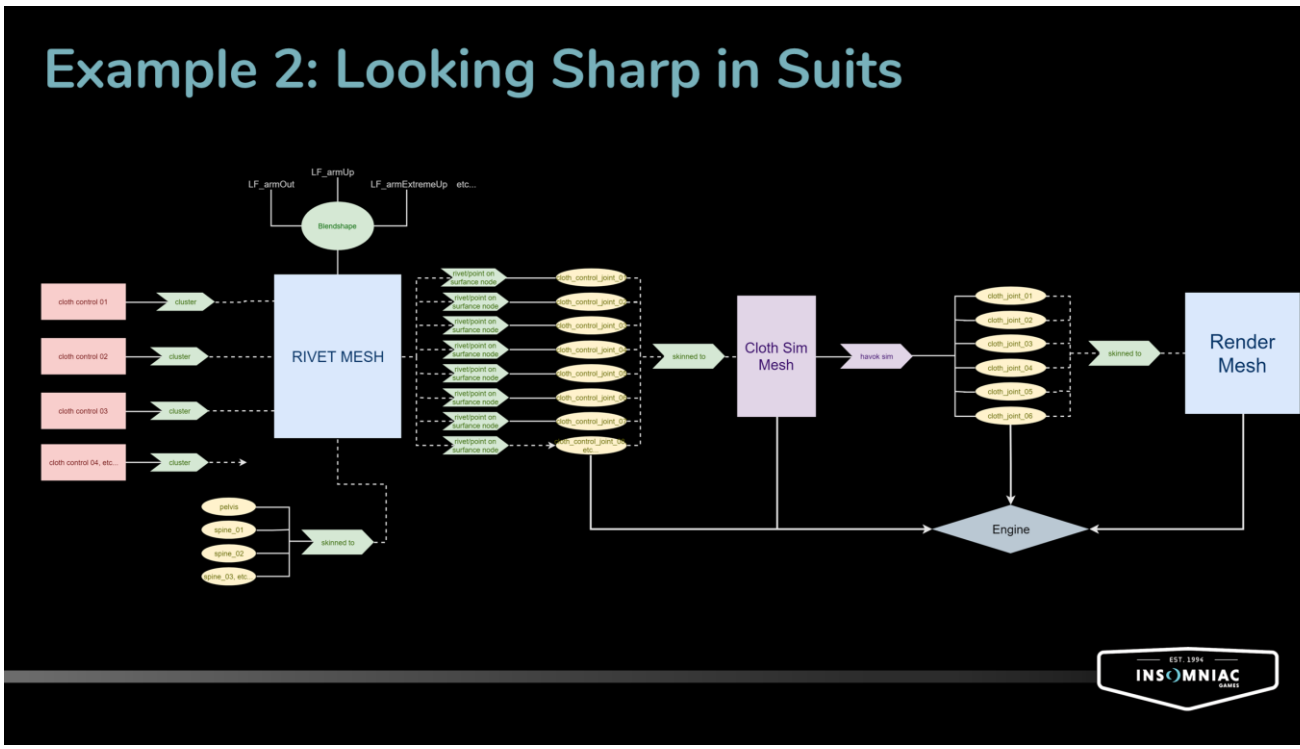
These cloth controls were then skinned to our sim mesh. This is the real-time cloth sim that would go in game. This means it would be following not only animator adjustments from the earlier cloth controls – but any movements from the blendshapes (so collision could be accurate)



The cloth sim mesh fed into engine as a havok sim which in turn our final skinned render mesh joints were being skinned to.



Here’s what went into engine. Both our cloth sim driven by our riveted cloth control joints (which could be animated and would echo our blendshapes) and our actual skinned render mesh which was driven by the cloth sim.



And here’s the whole setup. Not too bad when you break it down.

The wind effect seen in the helicopter was actually not done by cloth, but by iterating between a series of ‘noisy blendshapes’ that were added to give the rippling effect. This is different to Spider-Man’s suit which was done by animating normals. However, because the suits were so dark, normal animation would be lost, so we had to create a moving surface to get the wind rippling effect.

Finally, we tuned the shapes and behavior on the cloth, ties and sleeves in our real-time cloth sim to get the weight and dampening on the cloth feeling correct.

All 3 of these elements, blendshapes driving the sim, the

real-time cloth and corrective blendshapes on the shoulders helped sell the final look and feel of the suitjackets in Spider-Man.

Let’s move on to our final example for this presentation…

We mentioned MJ earlier in the talk, so let's go over the particulars of her setup. She 4 elements to her cloth setup.

Her hair,
Scarf,
Outer jacket (the leather one)
Inner jacket (the green one)

All these separate elements had their challenges, but the biggest issue we had is that our cloth had no cloth-to-cloth collision built in, so we had to be very careful with our limits and use collision capsules smartly to make sure we didn't have cloth clipping issues. For us, in cloth, this was our

BIGGEST hurdle. Making sure cloth did not clip for the majority of the time was a real challenge. You might notice in a lot of games cloth sim clips all over the place and goes straight through a lot of objects. We wanted to avoid this as much as we could – so we build a lot of our cloth elements to avoid those kind of scenarios.

Here's an example of the sim meshes on the left and the behavior in game on the right.

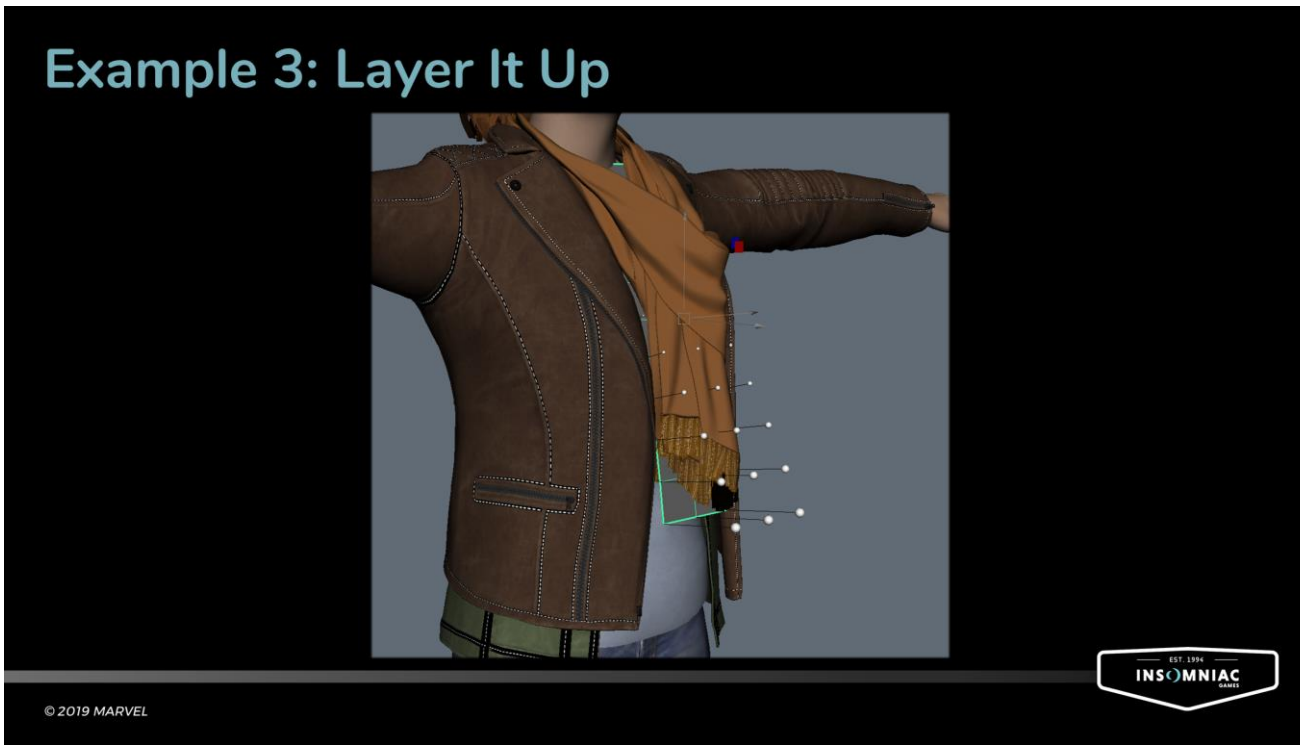
Much like our other examples, we used a series of blendshapes to enhance the shoulder and elbow shapes on MJ's jacket. But on top of that, she had this layered jacket with a longer inner portion and a shorter outer portion. Not only that, but MJ's crouching (used mainly in the mission taking place in GCT) caused her knees and legs to often 'trap' the cloth. Some games avoid that by modelling the cloth to fall to the side of the legs, but we wanted to try and avoid that as it can often look unappealing and unrealistic.

Firstly, we DID sim these two elements separately. We could have bundled them into a single sim, but this wouldn't have had the pleasant overlapping action you can see in the final game as even if we skinned them at different rates (risking clipping happening) we probably wouldn't have got such realistic behavior. The main trick to this was making sure the inner layer had VERY strict distance limitations on it. This meant that it couldn't really escape this really tight cone of forward/back behavior but had some nice side-to-side motion.

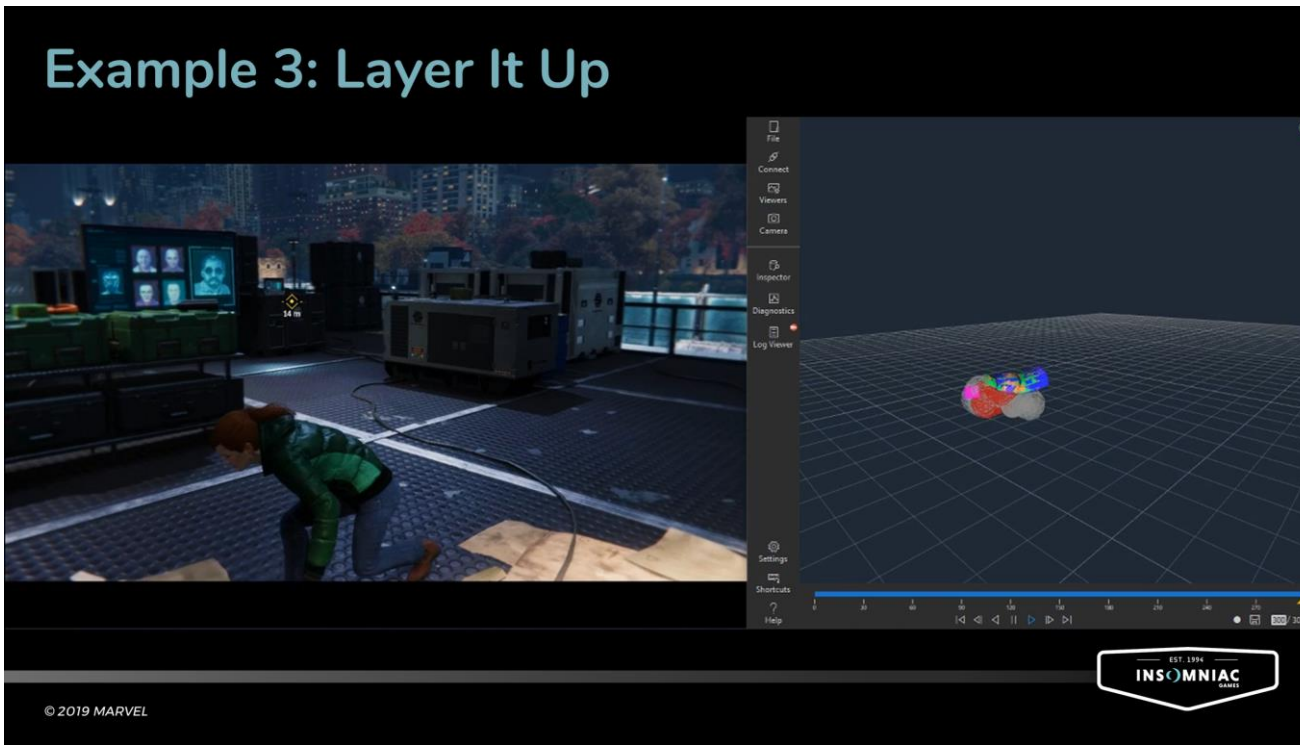
Finally, we had to have controls added for animators to have poses that would 'coax' the cloth into the general position. Due to the nature of the cloth, it would move into these poses and then relax if the movement so this was a tricky balance.

The scarf overall was a fairly simply setup. However, we had to make sure to limit the side-to-side motion on it to make sure that it didn't collide through the jacket (as cloth-to-cloth collision wasn't possible). To do this, we added small colliders parented to MJ's chest to keep it within a small range. However, we didn't reduce the overall range as this would mean upon leaning forward the scarf would reach a wall.

Havok currently can't limit side to side motion so we had to resort to 'capturing it' with capsules.

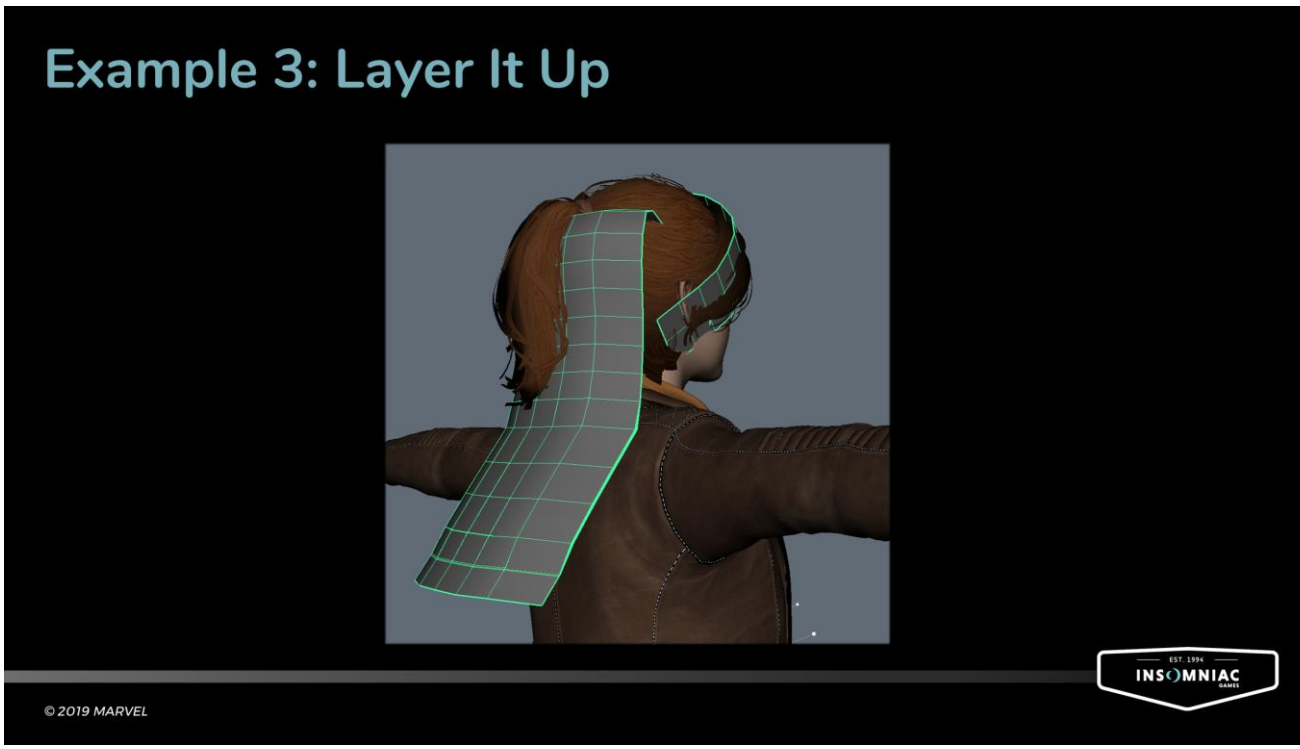


Here’s a small screen capture of the kind of ‘distance’ limits we put on the scarf within the content in Maya. We’re generally paint a very soft gradient from top to bottom, making sure the top two edges (the top edge was ‘pinned’) had no distance to prevent shearing when the mesh transitions from skinning to sim.



The ponytail was complicated as MJ wore it in a variety of outfits. However, it would have to work with each outfit. We didn't want to double up the geo/sim by duplicating this setup per outfit, so we tried to 'coax' it gentle within a range with a use of different colliders across the outfits.

We encountered a similar problem to before where we needed to limit side to side motion but were unable to do so with constraints. However, in this case – we couldn't 'capture' with colliders as MJ's head moved side to side a lot and colliders would create an invisible wall. Eventually, we ended up having to stiffen up the hair sim a ton, and lengthen and widen it so reduce the range of movement.

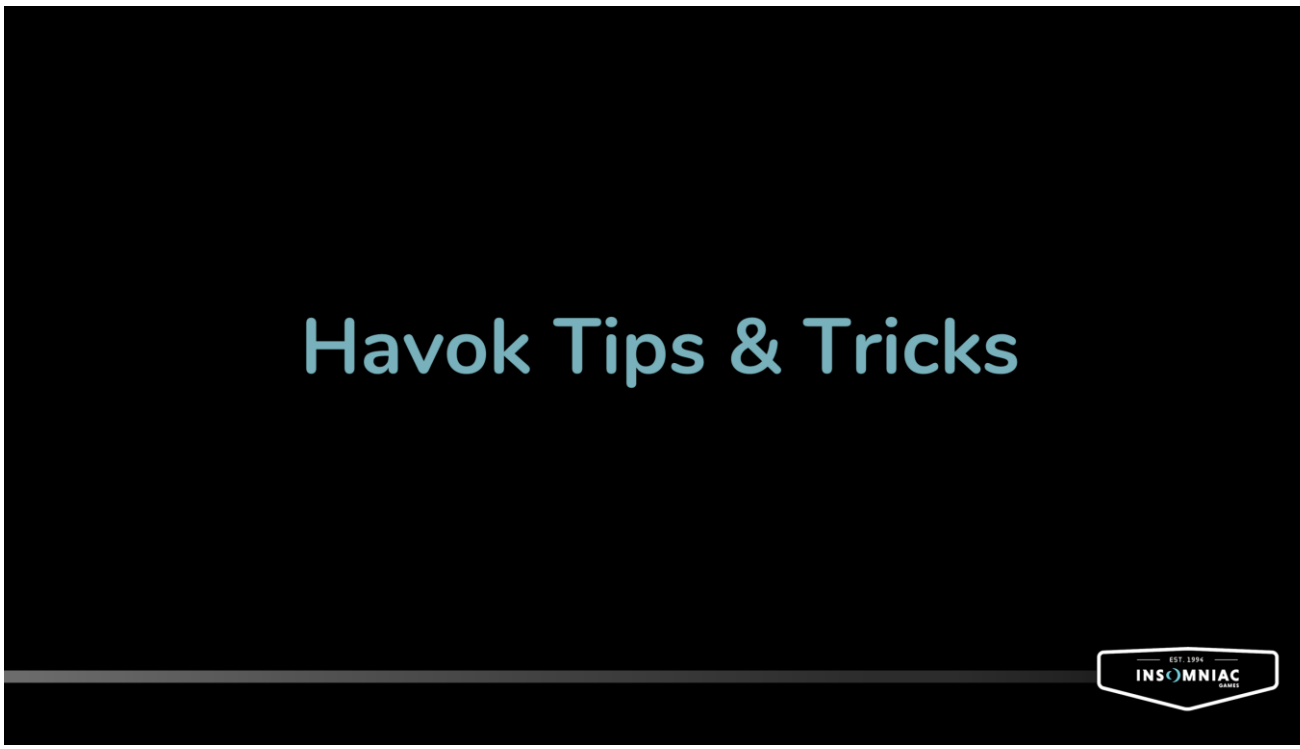


Here’s what’s the sim mesh looked like in Maya…

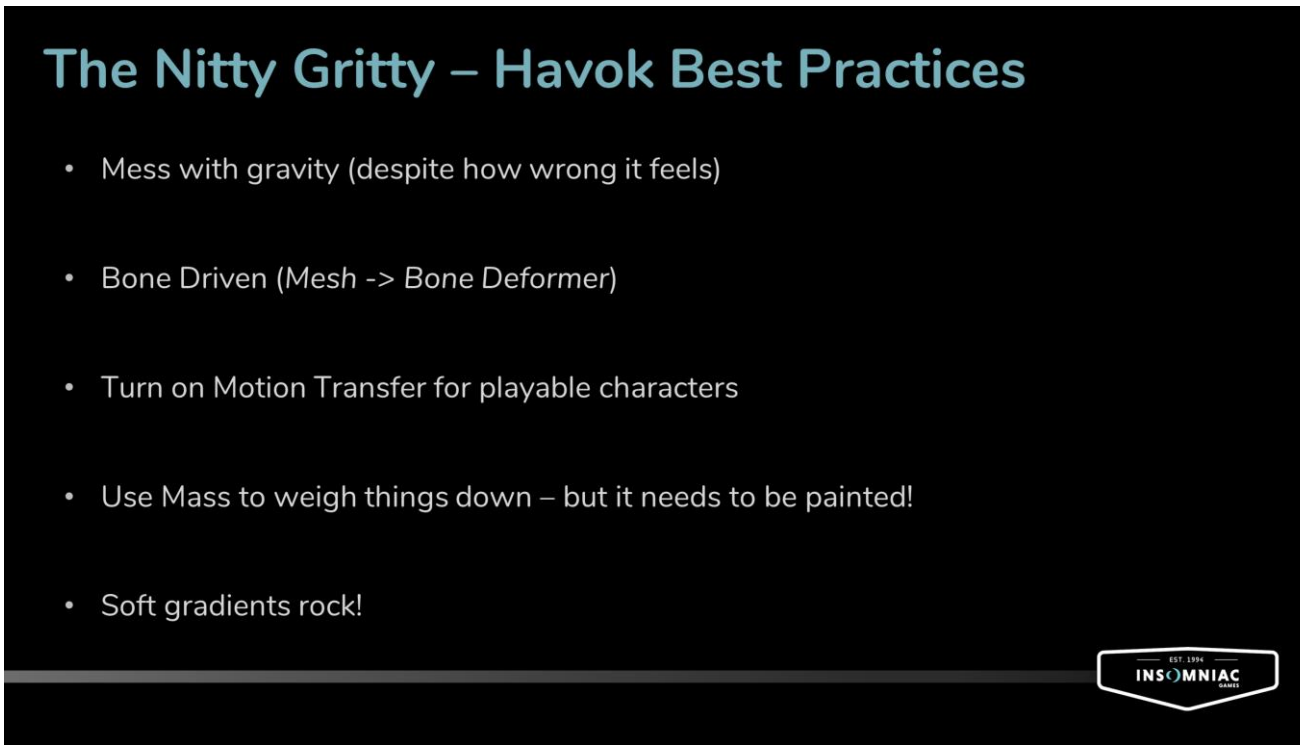
To get the behavior we wanted – we needed to extend the sim way past the end of the ponytail and the width of the sim had to be widened a lot.

It’s also hard to see, but we had 3 hair sim running on top of each other with VERY slight adjustments to get some variation in the hair .

Finally – there were some things that didn't make sense or were too time-challenging to do with real-time cloth and had to be done by animators. Because we had added (optional) controls to MJ's cloth, the animators could technically turn off cloth and animate by hand. For a scene such as the one above, we had to rely on entirely keyframe animation to get the job done. We had more animators then character TDs and by now we knew enough about the cloth that making a new setup for a couple of scenes was just too expensive. This is because we would have to duplicate the setup and the geo to get different behavior, but also every time we R&D'd a new behavior it would require a lot of time from the TechAnim department.



So, hopefully that previous video gave you a little taste of what our cloth looked like on Spider-Man. But before we dive into our pipeline and solutions, let's talk a little bit about the methods available to us…



This is more specific to Havok itself, but since this is what we authoring with, it would be good to share the knowledge here.

There was a lot of learning curves for us – one of the main ones being that it is okay to play with gravity. We know gravity to be a pretty fixed and static number for physics, but in Havok it had the effect of making everything look heavier or lighter with a one number cheat change.

As mentioned earlier we used Havok Mesh-Bone (bone driven cloth) cloth deformer as their Mesh-Mesh(vertex cloth) deformer would have been too expensive for us and

didn’t allow us to do depth on the cloth.

For player characters, we made use of their Motion Transfer system, which limited the speed at which cloth could accelerate and stopped cloth from ‘snapping’ into position.

We used the Mass to weight down cloth, for example to make the start or end of a setup feel heavier. For the Mass to work, the you needed to paint custom masks with different values across the mesh – a constant Mass value of 1 would act as tho Mass were disabled.

Also – soft gradients rocked! If you painted very specific information into your mesh – if there was any large movement you would get anomalies like bumps or jittering.

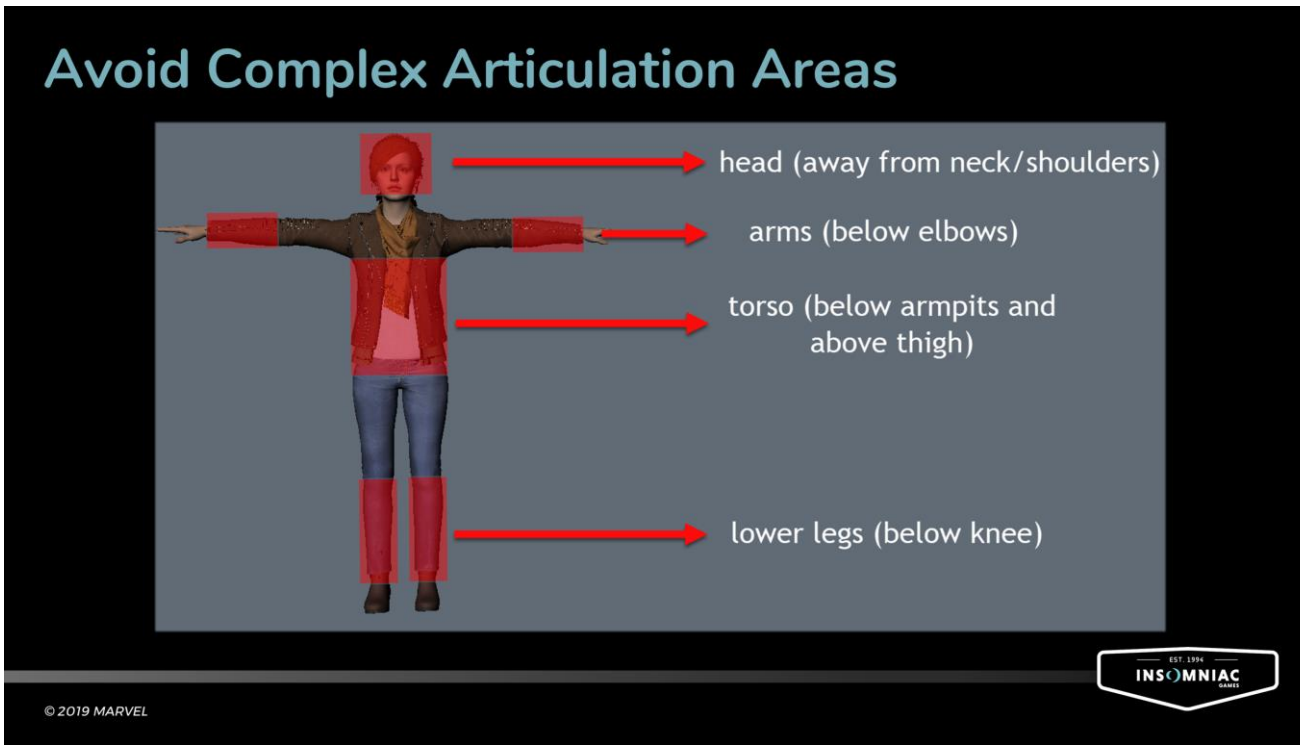
The Nitty Gritty – Havok Best Practices

- ***Standard Link*** & ***Stretch Link*** Constraints were always in our setups
- ***Bend Link*** and ***Bend Stiffness*** would add stiffness (in slightly different ways)
- ***Volume Constraint*** was overall expensive and we tended to avoid it
- ***Bone Particles Angles*** and ***Bone Global Plane*** could be used to constrain a sim within a pre-defined area (a lot like distance)

Finally, here highlighted in yellow are some useful Constraints I used frequently for our cloth setups.

Standard Link was the basic/standard constraint and Stretch stopped stretching. These were always there unless you just wanted your cloth to look and behave like spaghetti!

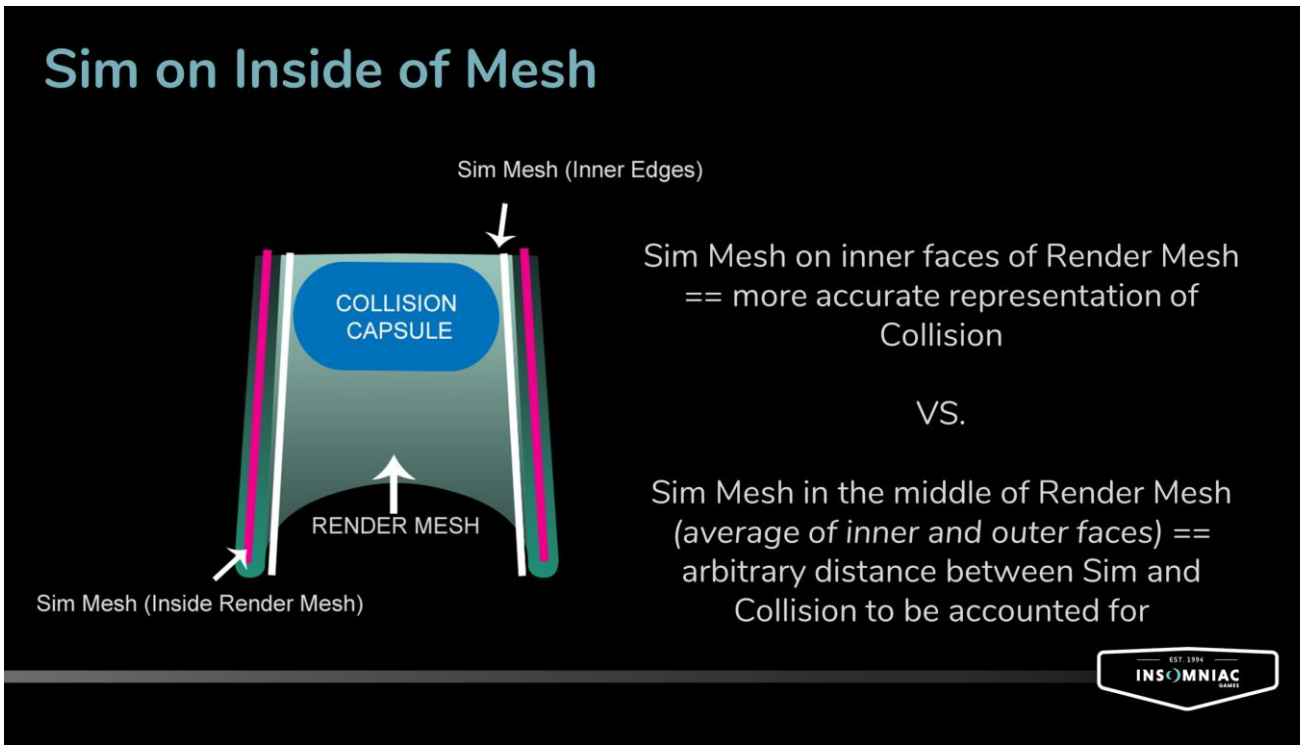
We would add Bend-Link and Bend Stiffness to create more rigid behavior in different ways. Volume Constraint tries to keep the shape of your original mesh – but I found I rarely used it.



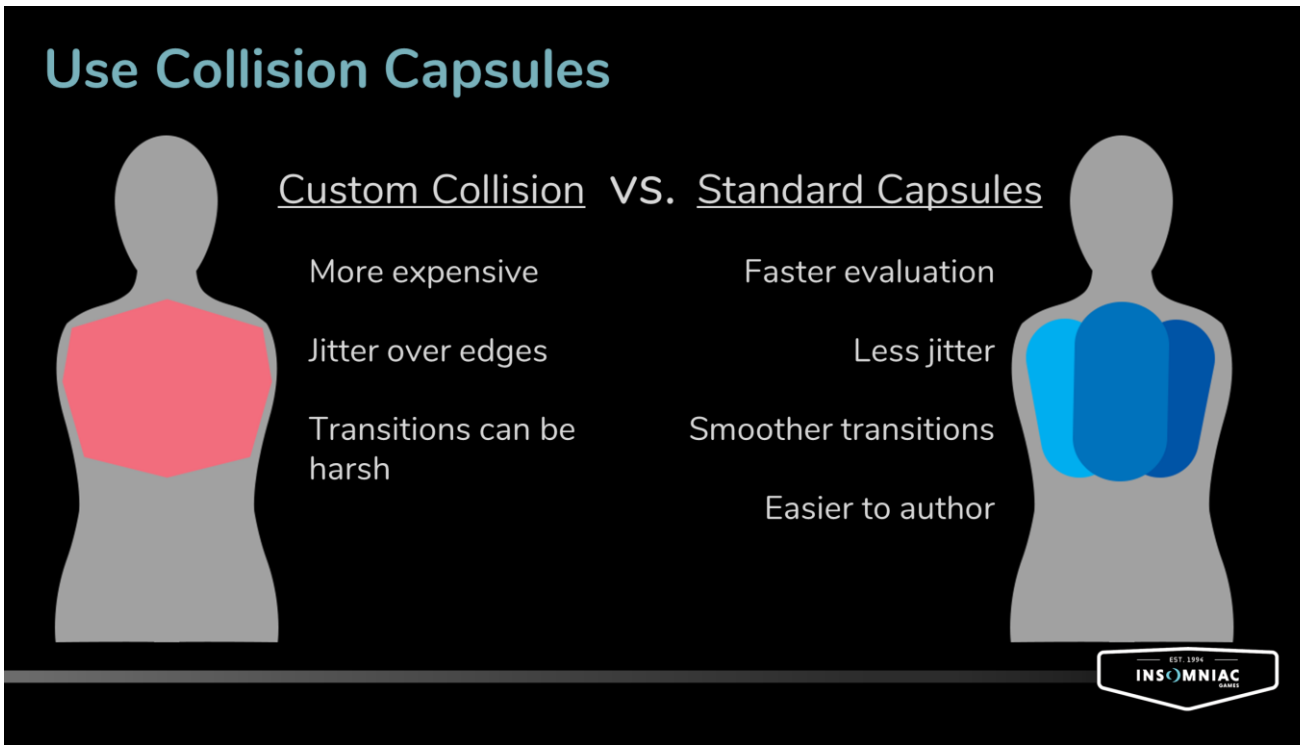
When it came to where to put cloth – we avoided any complex areas of articulation. There were just so many factors in these areas to account for that it wasn't worth the trade off.

We tended to stay clear of the armpits/shoulders, as these areas would add complication and simulated mostly from the armpits down. If a design came in with particularly complex cloth concept we would work with character to modify it in way that wouldn't compromise the design but also simplified the cloth (no point spending a bunch of time on something that could be avoided).

The areas marked in red here are what I considered ‘safe zones’ and were clear enough from complex areas. Not to say you can’t break these rules – but pick and choose your battles!



Our sim meshes had uniform polys as much as possible and kept as close to the character bind pose. We would often use the inside of the mesh if there was a thickness, so that there wasn't any guesswork when it came to collidables. If you used the middle of the mesh (an average position of the thickness) – there would be an arbitrary distance between your cloth and collison capsules that was challenging to account for. Remove the guesswork – put the sim on the inside of the mesh. It'll make your life easier!



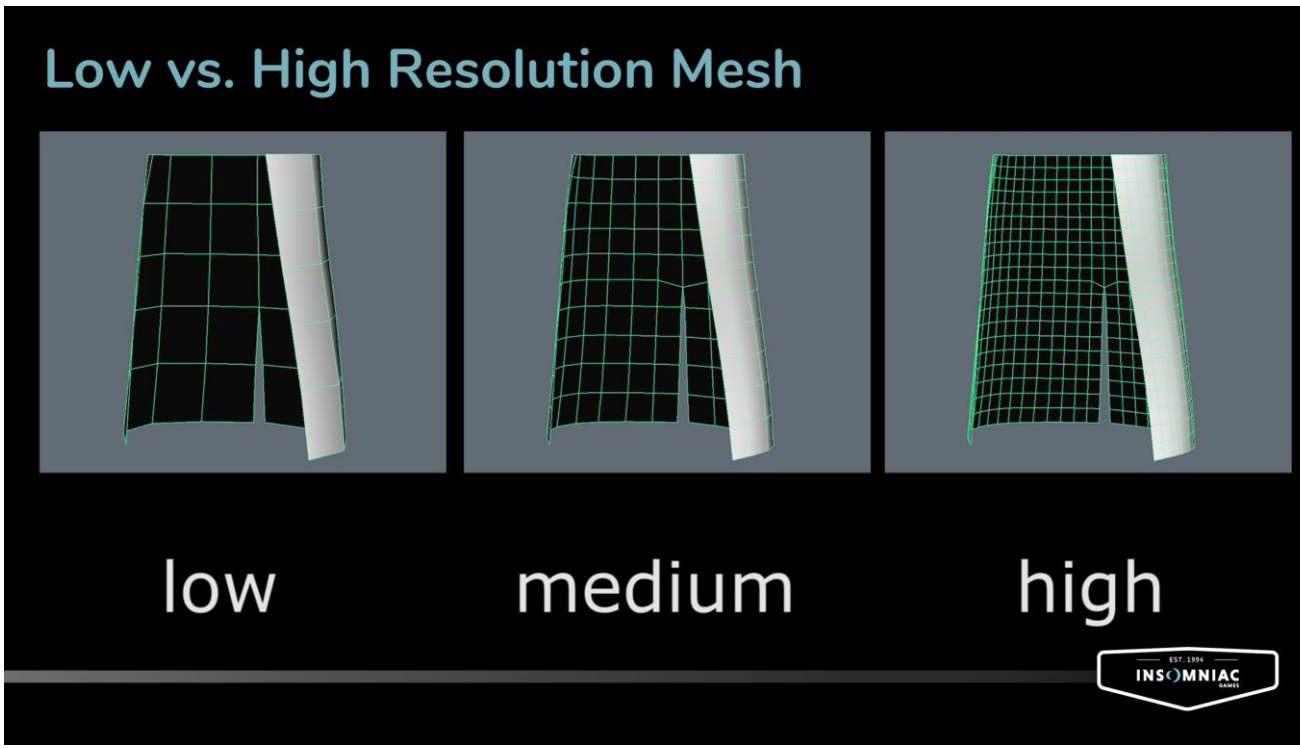
So… custom collision…

Versus…

Standard capsules….

I found we got better and smoother results layering up collision capsules versus using custom collision objects. Some very rough napkin math made it seem like it took up to 6 collision capsules to make up the cost of one single custom collision shape. Also, I found that jittering was more likely to occur on custom shapes. It was also harder to make smoother transitions between the capsules associated with

joints if they weren’t capsules. The rounder they were, the less likely it was to create ‘lumps’ in the sim.



We tried to be conservative in our cloth setup by creating low-poly sim meshes – but this also had a behavioural link to our cloth. Lower poly cloth created a stiffer feeling, where adding more resolution caused the cloth to behave more silkily.

Here’s an example:

Same settings… totally different behavior. (for the record is it out-of-the-box Havok behavior on the cloth here and I have not modified it at all for this example).

Try and nail down your sim mesh resolution before you start

doing iterations on it – otherwise you'll be causing a ton of rework for yourself later down the line.

Now that we got some tips and tricks covered… what else can you do? Well, if you are lucky enough to have an in-house engine team – you might want to get yourself some extra features!

Out of the box, Havok was able to get us to achieve a lot – but we still needed extra engine support. Luckily – this is where our Core team came in!

(As a side note – having an in-house engine is great as you can get to directly interface with your engine team and get feedback, answers to questions or even feature requests. Here's a nod to our core folks as they enable us to do awesome stuff everyday.)

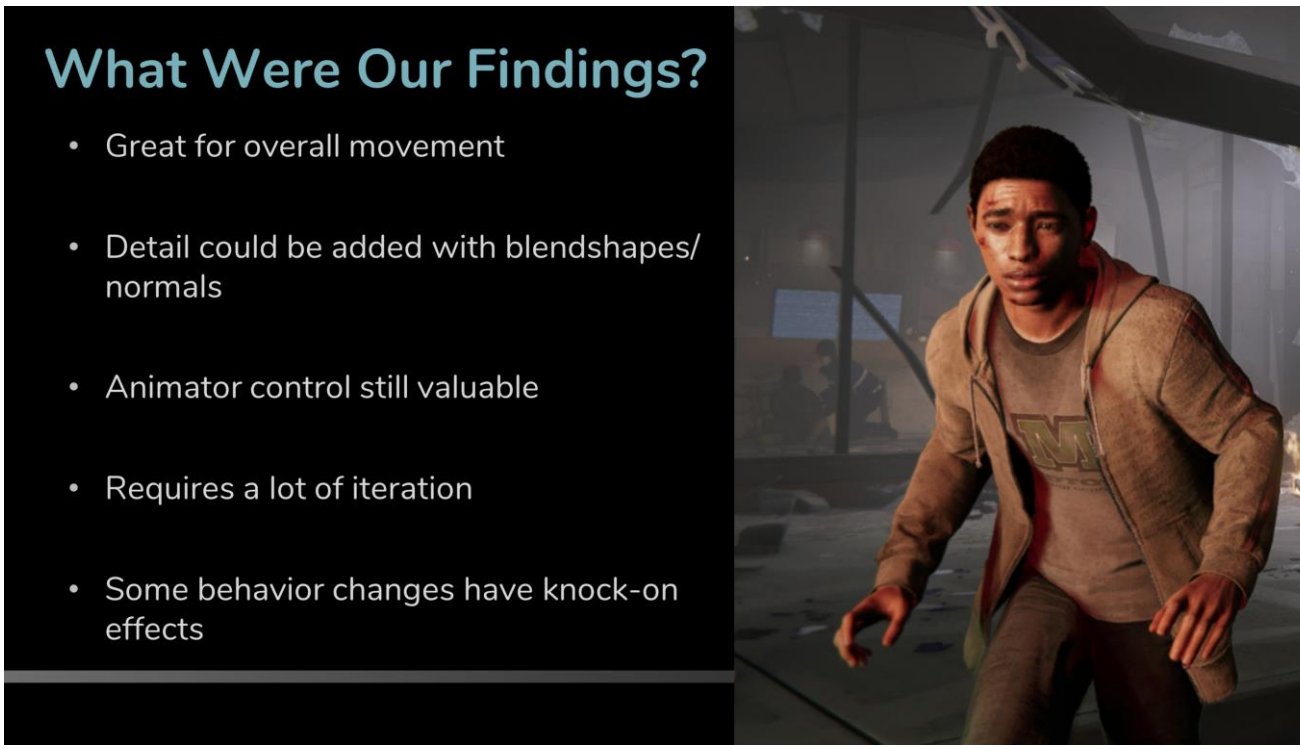
So, what did we get in terms of support?

Firstly, it came in the removal of cinematic cloth pop. What's

that? Ever seen a game where the camera cuts and all the cloth settles on a frame? Our core/engine team worked extremely hard to remove that – which was no easy feat. Shout-out to Reddy at Insomniac for his tireless work on this.

Next, came the creation of events that let us trigger cloth on and off, as well as alter some values in cinematics. These events were gameplay triggers that we could assign in various ways in the game. In some cases, where we needed a more art-directed approach to the cloth, and the ability to disable it per shot was utilized to get the look just right. As mentioned earlier, we would then swap to the animator controls and the animator would animate the cloth accordingly. We did this in shots that were overly specific for the look/feel we needed as it was often more time effective than trying to develop a new setup.

We also added some events for adjusting gravity and dampening on the fly. We needed it in some very specific circumstances that helped solve some particularly tricky problems.



So, from our work on Spider-Man - we developed a good understanding of what real-time cloth was good at and where we had to sprinkle on a little extra magic.

Real-time cloth was great for overall flow and larger shapes. Nothing else could get us that immediate dynamic feedback and follow-through that it gave us – especially in gameplay. It also saved us a ton of time by using it in cinematics too – that allowed us to iterate without destroying the sim.

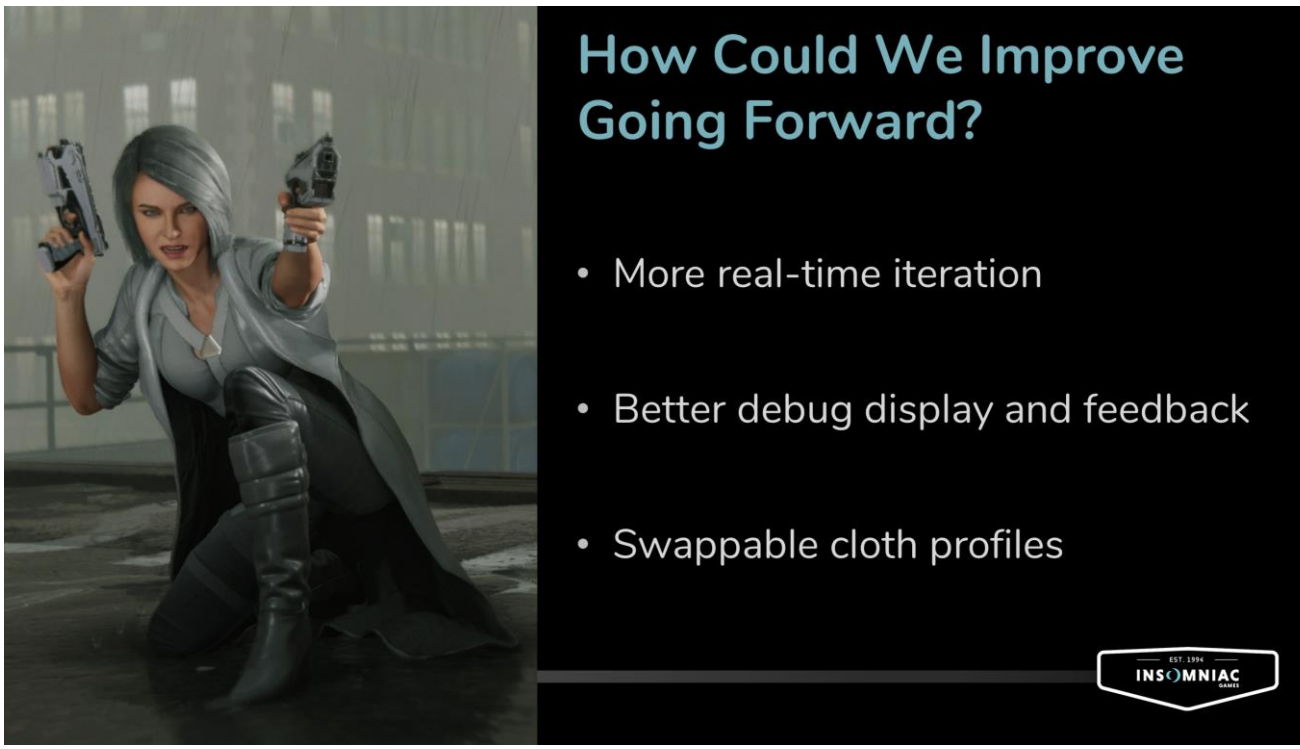
When we need more detail – this is where we would start using normal maps or blendshapes. This would let us add the more granular visual information that we couldn't afford

to do so with the sim alone.

Despite my initial philosophy of wanting to keep cloth controls as far aware from animators as possible so they could focus on performance – we could we still needed them. Whether it was to massage the sim into position or complete override and control in some instances – it was something that animators wouldn't have to touch 90% of the time – but was really needed when they did.

Most of my time with real-time cloth was spent on polish and iteration. Nudging a collision capsule here, subtracting a radius value by 0.001 there… To avoid clipping there was a LOT of noodling of the numbers. Don't underestimate the amount of time this takes – cause I sure did!

And with that… behavior changes that might seem trivial (like subdividing the sim mesh so it doesn't pass through collision) were actually massive and had huge knock on effects. Unfortunately, there were times where we spent a lot of time iterating on a particular setup before coming to terms that wasn't going to achieve what we needed it to. Changing some components of it requires all other attributes to be rebalanced – which basically ended up a re-do. And as previously mentioned… iteration took a LONG time! Try to make sure you're happy with your sim mesh as much as possible before jumping into polish.



More real-time iteration. We'd love the ability to be able to adjust values of gravity, dampening, mass, etc in real-time. Each time we altered this, it would cause a rebuild of the asset and export times. While we did lots of work to minimize this, the best solution would be immediate feedback. This would also allow us more instant debugging, as we could alter properties at will to see what effect they are having on the behavior.

Better visual display. It would be ideal to add the ability to see our debug draw within our engine – to remove the disconnect.

Another thing I'd love is the ability to swap cloth profiles on the fly. Our current system is to duplicate parts of the model and sim. It would be nice if we could alter these real-time so we could have different behavior setups for different scenarios. It would also mean that if we got a performance we liked on one scene, we could keep that intact and adjust it on other scenes without worrying about breaking it.

So let's sum this all up!

Did we deliver on our promise: to create a believable world and cloth that services that? I believe we did! Not only was it belieable – but it was also abundant! Cloth was everywhere – in the enemies you fought, the npcs you talked to and in our vast cast of characters. A cursory glance at our depot and we had around ~70 different assets with active real-time cloth – and that doesn't include the separate setups inside the models themselves! And we did this all without our small and versatile rigging team – no one person was dedicated to cloth!

And while we were at it – we improved our cloth tools and pipeline A LOT. Our engine team worked with us tirelessly to give us the best results. Whether it was removing cloth pops – to better and quicker loads – to developing new ways to filter it on and off in our engine… from start to the end of the project the pipeline was transformed. I also helped streamline the content side as I removed as much room for error as I could and made our content pipeline much more straightforward.

Finally, and perhaps this is a subjective opinion, but I believe we made some of the best real-time cloth developed ever – especially in such a large scale open world game. We pushed the bar and tech and for that I'm proud of us and our entire team at Insomniac.