Preamble Note

The following is a rough transcript / collection of notes from a "requirements gathering" meeting with VR artists Debbie and Dennis (dennisanddebbie.club).

This session with them began with a brief explanation of my project to them before I was given a chance to enter the Intern experience they developed as they talked generally about its development. This was followed up with a code walkthrough of the source files as they outlined and discussed the difficulties they experienced during development. Throughout this process I would ask questions regarding issues and topics which arose.

This write up contains a general summary for each subject of discussion alongside my initial reaction and thoughts regarding each topic. These additional notes include potential ideas for the project, general thoughts and comments, and ideas for example scenes which my toolkit should be able to rapidly create and may be created as the included example scenes for publishing on the Unity asset store.

This meeting alongside additional research, and some leeway for feature discovery, formed the requirements specification for my dissertation project.

General Preamble

Started by giving an overview of their work which involved types of projects which they have worked on thus far. Varying age ranges. I did ask about the Kurt Cobain porch experiment. That was not one of their more child friendly experiences. Interactive movies and experiences as they've moved away from the more installation type of thing.

Dr Williamson gave a brief overview of what my project is and the attempt to minimise clicks. She said 5 as a number. Dennis seemed impressed in that "how on earth will that be possible" sort of way.

Excellent.

They were a little worried about the different versions of Unity but it all seemed to work correctly. Dennis started getting PTSD as he started to look at the source code though Debbie had used it slightly more recently as she tried to implement a puzzle. They hope to release the Intern experience on Steam but need to make it a bit more game-like.

As things loaded Dr Williamson told Debbie and Dennis her plans for the VR workshop room. She wants to turn one of the walls into a giant green screen. By that I mean she wants to paint the wall entirely green and then have a curtain you can pull over it to make the room entirely black. She needs the curtain because she wants to make an entirely black room for some other VR experiments. Debbie & Dennis have a green screen in their kitchen. They also did this incredible thing where Debbie struggled to find the right English word then shouted "Dennis help me [some German word]" when he then filled in the word.

Dr Williamson had to sign into Steam. She was worried about if she could remember her password or not. She got it first time though. She also brought up SUPERHOT! which is incredible.

SUPERHOT! is a game with an amazing and an extremely interesting concept for a gameplay mechanic. SUPER! HOT! Also I noticed that when I talk about video games there's a subtle change in my tone in an "excitable, passionate, child-like" sort of way. That's interesting.
Dr Williamson also plans to get rid of some of the workbenches in the workshop. Make a VR lab and pull the curtain over her door and hide. Also they have a new "fancy" coffee machine in the computer science department. Dr Williamson is excited about this new machine. Debbie and Dennis were in Germany yesterday / recently and expressed at how bad German coffee is. The bakeries there have this unbelievably strong coffee that tastes like it has been brewing for 5 days.

I wouldn't know, I don't like coffee so it's all relative to me. I did try and "like" coffee, I've been told it grows on you after a while, during the summer at my internship. The office had a coffee machine and it was free. I still don't like coffee.

Dr Williamson went for coffee as I entered the Vive.

Joseph Enters The Intern Experience

One of the interesting things they did related to difference in the user and viewers perception of the game. This was specifically regarding the user's hands. In the game the user sees both of the Vive controllers as is typical with many VR games. The viewer, people watching the feed of the game via the outputted video however see hands rather than controllers. Just this idea of the onlooker seeing a different model of the hands than the player does in-game. They did this to combat the idea that for the people viewing the experience from outside of the headset that seeing the standard controllers was a bit boring so instead opted to make onlookers see different hands opposed the player ingame.

That's an interesting idea. Being able to change the models of not just the controllers but also to specify that you want the in-game model to be this and the video output model to be something different. In regards to changing the models of the controllers I'll need to think about the many different ways to do this and issues that surrounding this task. For example do you always show the controller when you pick up an item or do you go for the Job Simulator route and make the hands disappear when the user picks up an item. In generally I'll need to look into issues surrounding controllers. Other issues such as mapping the buttons - how easy is it to set up and do that.

The experience Intern opens with an intro scene which they put in for the players to become familiar with the environment and using the Vive in this setting. It essentially amounted to an empty room with some displayed text to read and an object to pick up which would transition you into the next scene. Other methods they have used for introductory reasons are playing an audio file though this they've found must be able to loop as players occasionally miss or ignore the audio the first time they enter into the world. As a result they do not hear a complete set of instructions.

There was a unicorn to grab and when you pick it up fire starts shooting out of it. It is as awesome as that sounds.

You then enter into a room which you can explore. The brush was difficult to pick up but the bucket and many of the other items were not as difficult to pick up. There was a bug with the bucket in that you couldn't put anything inside of it rather it would just float on an invisible plane on top of the bucket. So you'd go to put something inside of it and it would just float in midair.

Another general bug they experienced was with collision between objects. That is if you pushed one object on top of / into another the physics would break and the object go flying off in a random direction. This also occurred if you sandwiched an object between two objects that you are holding. So when I got two boxes and pressed them into another box a table the middle box would go flying around. Or when you push the box into the table and let go a similar thing occasionally happened. It also happened when you'd use one box to push another box into the table, the box being pushed would go flying around.

Within the square room that you go to before heading into the oval office there's a cabinet with a cupboard and some drawers. The hinges worked well in this square room as they would easily move along the rectangular plane of the room but when put into the oval office or not up against a square wall the drawers became a lot less responsive and didn't function correctly. You'd lose grip of them as you pushed them and they moved around quite jittery.

The drawers in the square room worked out ok because they are aligned alongside the world's x and y axis. Once you rotate the drawers away from this global axis it becomes a lot harder for the drawers to function correctly and control. This is very annoying. A circular world doesn't have the same ease with which to align the drawers on a axis hence the somewhat laggy nature of them. This is even more of a problem.

I'll need to look into Unity hinges. We spoke about this topic later on as well.

The collisions between items were a bit of a problem for them during development. They found that a Mesh collider would have been preferred to a Box or Sphere collider in terms of making the objects interact correctly but using a Mesh collider made everything jitter due to the way that the Mesh collider works.

There were some cardboard boxes in the room which they hoped to make openable. Unfortunately this was cut (in part because you have to fill the boxes with content and items) and it was going to be too time consuming for them. They would also have to handle and deal with testing the collision worked with that as well.

I asked if they found it difficult to place items and things in the world exactly where they wanted. It seems that this could be an issue for developers to get the object placement just right as the sense of depth can be challenging to judge within the game. That is are you making the item too big or small in comparison to the size of the player in-game.

Debbie found that to be quite troublesome. In Blender there is a snap tool so that items can snap in place but in Unity the only tools serving this purpose are paid. Debbie would very much like the ability to snap items to something or places. There solution was to create the entire scene in Blender and handle all of the positioning there and then import the entire scene into Unity. This

saves the origins of the objects though is not an ideal development practice for Unity. This is a workaround.

This is an issue when they are attempting to make something modular. So when they're working with basic geometry they just want some stuff to snap. So you've got parts of geometry like or example if you make a staircase you would want to be able to reuse that staircase. Not being able to snap that staircase into your geometry is really annoying.

Unreal does this snap functionality hence it's a lot easier to do little worlds in Unreal than it is in Unity, in part because of this snap functionality.

I'll need to experiment with Snapmap and all that and see if it's possible to recreate that in Unity. Maybe also look into the preferred / professional / recommended methods for setting up a Unity scene with the background, play area and all of the objects as well.

That above idea is more just for me.

Dr. Williamson entered the moment I loaded into the oval office. It was timing perfection.

They have a second camera setup to show where the player is within the oval office. With this there is a similar mesh / model with the player's head that the viewer can see on the output but the player cannot within the game.

This project came together in 2 months which was impressive. They didn't sleep much. A solid 2 months of crunch right as another project came to a close. Ahh… the artist life.
They complained about the finicky nature of hierarchies and boxes and items within boxes.

One of the problems they brought up again was getting a model from Blender into Unity because of the way Unity likes everything on its global axis.

Dr Williamson and Debbie / Dennis complained about how you'd think it would be so easy to make a door or a drawer in VR just work. It's one hinge.

It isn't.

I walked into the wall. Dr. Williamson suggested I watch out for the boundary wall. I do try. They spoke about green screens some more. I hope she gets her green screen.

Hinges were a real problem that they faced.

I left the VR space as we all sat down for a code walkthrough. My chair was one of the musical chairs Dr. Williamson leftover from a student's summer project.

The Code Review

So you've got the main drawer parent and underneath that was each of the sides of the drawer. They did it that way so that they could just provide a box collider with each part. Because the box collider is always allied according to the geometry. If you make the entire thing a single box collider it requires going in and tweaking some additional things so from a workflow perspective it is easier and preferred for them to instead when creating the models make each part separately and construct the entire object. They didn't use a mesh collider here because it would have cost more. Also as a mesh collider they were fitting too tightly and would either fly out or start glitching out if upon being touched. There seems to be some Unity thing which recommends using non convex colliders aren't allowed with this and mesh colliders are convex colliders. Something like that.

Look into box colliders, mesh colliders and all that in Unity. Also that whole convex system.

For VRTK there isn't a gravity associated with the drawer so if it was just the drawer by itself then the drawer would sit in mid air. They then went over the 3 scripts associated with the item to make it move like a drawer.

Look over how VRTK handles interactive objects again.

One makes it interactive, highlighting when you move a controller over it and what happens when you grab it. There are two boxes associated with the controllers (what does the left controller do / what does the right controller do) and what you can do with them. You slot the scripts of how you want the controllers to work into here. They find this gets complicated very, very quickly.

They quickly went over how you define the interactions with the controllers. So using a mouse as an example and interacting with it by picking it up. You could just pick up the item in the position that the user picks up the item. Alternatively you could make the controller disappear (like what happens when you pick up an item in Job Simulator). Or you might snap to a specific point on the object. So you pick up a sword but immediately snap to the handle rather than the blade. These are the types of controller interactions you get associated with the object and then slot in. Remember this is just for one drawer.

Also why have different scripts. Why not a pick up script and then a checkbox for whether the object replaces the controller or not. It might be nice though if you want the controller to disappear on all interactions to have a single override that changes the default from see controller to invisible controller. This could be set on an individual controller basis. So I want the right controller to disappear in all scenarios but the left to stay visible in all but the case where you interact with that single lever.

The snap position could also be an optional point on the object that you set up an empty gameobject or something on. That is suppose you want it to snap to the handle of the sword. Then signify that position on the sword with a gameobject or something (child of the object you are picking up) then drag that object into the position (similar to the vive button example that I did with PtA and PtB) that says if the user picks up this object then we'll snap here.

So we import the sword model, and select the checkbox saying to replace or make the controller invisible. Then we create child object / highlight the snap pick up point in some way and then tell the pickup script on the sword about this point. Now when the user picks up the sword the controller disappears and it snaps to the highlighted area.

I should probably find some better models to play with rather than simple cubes and stuff. I should test it on both to be fair.

Actually there's probably an interesting simple puzzle game that could be made purely using pulling levers. Like that puzzle in Mario + Rabbids towards the end of the game where there are 4 levers to pull and a grid of 4 squares (each with a maze of pipes) in front of you. Pulling a lever rotates the associated square 90 degrees. You rotate each of the squares until the pipes line up correctly and lead to the exit.

When this is part of the workflow this just becomes rote routine but as Dr. Williamson said I need to be thinking about reducing the coupling that's involved in this process and the invisible coupling that's involved. So we've got all the scripts for our controllers setup for this drawer but suppose we accidentally delete one of the controllers. That has a major effect on the rest of the system and breaks a lot of things but Unity doesn't tell us what scripts are missing and from where due to that one controller on that drawer being deleted.

So there's a of overheads and multiple configuration points which if part of it gets deleted there's no way of knowing if part of the system is missing due to the part being deleted. There's just no way of knowing where the missing scripts were deleted from.

Dennis exclaimed that this is especially apparent when you consider that most of the time your hands are going to be doing the same thing over and over again. A lot of the functionality is the same so make the exception that if you want it to do something else then there you go rather than the current setup.

Dr. Williamson then brought up that a lot of the you need to setup the left and and the right hand separately so you set up the left hand with all its scripts then you do the exact same thing for the right hand to get left hand, right hand functionality. Why not just have it default to left hand and right hand by default. Again the notion of establishing defaults. Why not just have one setup window (insert scripts here, etc.) with a checkbox that defaults to left hand right hand same functionality. If you want a single hand then check that checkbox. If you want both hands different functionality then check another checkbox which brings up an additional script insertion point so you have one for each of the hands.

But yeah this notion of slotting things in multiple times, why not make the default initialisation that it's both hands and have the option to configure it for each hand independent of one another.

Also very random thought but should there be a lead right handed / lead left handed for people who are right and left handed? That's just a strange VR thing I get to think about right now.

Dennis then highlighted the dense amount of options that are associated with each interactive object. “Is useable, use only when grabbed, hold button to use” “A lot of these are not necessary... because it’s just a drawer it’s not a lightsabre you can click on or something”
Debbie then highlighted that “is grabbable” is off by default for interactive objects. They found this should be reverse and you should be able to interact and pick it up by default and state that it is not grabable if you are not supposed to be able to pick the item up.

Why not have a grabbable script opposed to a very general intractable script?

Its essentially set up that way by default for buttons, so you can interact with a keyboard but not pick up all of the buttons on the keyboard. Or pick up a lightswitch rather than simply putting it on.

There’s just a general issue with gravity and buttons in VRTK. They had a lot of trouble where the button on the desk would just float away. Giving it gravity however simply made the button constantly be pressed down and turned on. They ended up using a Unity joint script that was resulted in a little joint that when you zoom in to interact with it keeps on getting smaller. Angular X, Y, Z motion is just a nightmare and how it tells you to attempt to work with it. Furthermore changing to certain perspectives which you would like to work with it at made it disappear completely. Remember the little arrow you are meant to interact with. That seemed like a nightmare to work with. Also because it was at an angle you couldn’t look at it from the side so in Orthographic view it just went to. This also related to the hinges on the drawer.

Unity configurable joint - look it up
Buttons - fix them

So as Dr. Williamson suggested what is needed here is a drawer asset that one script gets dropped onto it and you specify the front. A little refinement to this would to specify the start and end, pt A and pt B. But the rest should just be taken care of. Because a configurable joint can be a lot of different things but a drawer is a drawer and it’s common enough and does a simple enough task that you should be able to just drop one script on it and specify the front and then be able to interact with it and pull it. And then you just place it where it would go. But yeah setup the start and end points, attach a script and all you have to do is place the drawer in world and it pulls along that line.

In some ways it might be best if the drawer is pulled along the line of the two specified points. So if the points are (1, 12) and (5, 12) it will be pulled in a straight line along that path, whereas (1, 12) and (5, 17) will be pulled along at an angle.

Back to buttons remember the hand dispenser. When you pick it up and move it around the water starts spraying everywhere. What they were attempting to make was that the water would only come out when you pushed the button down. However because of the nightmare setup they had with buttons moving the item would move the button and it would register the movement of the button going down as the button being pushed and the water spraying out from it.

Shaking buttons make them press rather than being pressed down.

They liked VRTK opposed to other methods.

The other desk button would play a little gif animation and some sound would play. Dennis said that the events system that is tied to the buttons is actually quite nice to have. That is you can tie an event to occur upon the button being pressed. So when the button or some other interaction occurs you can trigger an event (your own code or thing to do) which is a “basically perfect” system that they like. This system ties into the way Unity does things as well very nicely. They found this quite useful and not really in need of being changed.

Look into the VRTK event system with buttons and all that.

So you’ve got the events menu and in there you can set what event is triggered “on push”, “on touch”, etc. which they found to be useful and handy. For example if you have a door you can easily set it up so that when you hover over the handle that a locked sound effect plays or make it so that when you hover over something an animation plays. This is more intuitive and simpler than the hierarchy system native to Unity.

The animation was done within Unity and it was set up so that the button sets active to true and Unity by default then runs the animation. So by just turning it on it becomes visible and starts running. You aren’t declaring that it needs to run rather you’re just setting it to be active.

There are two sets of animations. One is the starting animation which is set active at the beginning so the Earth appears. Then there is a little delay which tells the animation to go from one animation to the next. So once you press the button the animation appears and plays out. Then we exit that animation again so that the animation just plays once, which can be declared within the animation. It also specifies the duration of the animation.

Another problem with the button is that it has an activation distance. Dennis believes that the button is designed to put on walls, the floor, flat on a desk, etc. so it ignores where its normal position is and then it assigns an axis X-Y and the direction activation distance then declares how far you have to push it in for it to do something. The main problem with this is that you can just shake the button and the activation direction thinks that you are instead pressing the button and so the button activates.

Trying to have a button on the hand dispenser didn’t work as shaking the item when it was picked up would trigger the button opposed to only triggering the button when it is pressed. When it's fixed to geometry and you cannot lift the button or the item on which the button is placed it’s fine but once it becomes grabbable the activation distance becomes basically cheated by moving the button or the object onto which it is attached to closer to the button rather than pushing the button further into the direction of the event, activation trigger.

Another example was the radio that was located on one of the shelves as well. It had a play button located on the radio but they wanted to the player to be able to pick up the radio and walk around with it. Again when they tried this the radio just kept turning on and off.

Dr. Williamson then brought up that really buttons are designed in this context to be anchored into the environment opposed to be attached to an object which the player can pick up and move around.

Dennis then expressed that buttons are such a common thing (like drawers) in environments in general. He suggested this idea of having classes of buttons. That way you could have a separate one that's designed to be put on a wall (like a light switch). So that button is always expected to be placed on the wall. Similarly one that's designed to be put flat on the surface of a desk (like a table button) and then one that's designed to be anchored onto objects which the user is able to pick up and move around. This one is more challenging as it requires that moving the object in the button's activation direction won't activate the button and trigger the event.

So this object based switch would be expected so that is on an object and does something for that object. Like a lamp switch, a play button on a radio, the button to turn on a lightsabre or something like that. Because when the button is attached to an object we want to be able to carry the object without constantly triggering the button.

Dr. Williamson suggested the default of anchoring the button to whatever parent object the button has and if there is no parent then maybe it just configures it to the world. But for the radio it should be able to check and say "hey I'm anchored to some object so I'm anchoring myself here and always going to use my position relative to this object as my sense of the world" [45.05 CHECK].

I need to think of example scenes to showcase each feature. For example a room of buttons might include:

- A light switch
- A set of floor (piano) buttons like in the movie Big
- A table button which spawns an item
- A button on a radio to turn the radio on (play Lockdown?)
- A button on a lightsabre

Time and number of clicks to floor piano or lightsabre is a pretty cool metric.

I need to see if everything needs to be featured in the Unity sample scenes to get approval on the asset store.

Example items to be able to easily pick up and use the button with:

- Hand Dispenser
- Radio (on-off button)

Direction activation distance and buttons VRTK

Another thing highlighted by Dr. Williamson, Debbie and Dennis was that VRTK comes with so many sample scenes and examples. One thing they discovered was that making a prefab out of the sample drawer and importing it into their scene made the drawer work a lot better and smoother. Taking over the elements of the prefab doesn't work. Recreating the drawer inside of their scene just resulted in the aforementioned problems. Dr. Williamson then mentioned that there's some

hidden level of complexity going on here where the developers have fine tuned the sample drawer with some settings that are not visible or easy to find when attempting to make or use the item for yourself.

So as Dr. Williamson suggested I need to be thinking along the lines of “Great here’s the prefab and the sample scene but how many steps does it take to recreate that from scratch”

Another major time killer during the development was testing the functionality and trying things out. Because they had to physically get up and put the equipment on, go over to the item in world and then test the functionality or change out. Then you’d have to take off the equipment, go back to your desk and make another slight change and then retest the change. This is a very time consuming process.

Sometimes it works better with two people but that’s also a major time killer as one person is coding and making slight changes and tweaks to the functionality and the other person is standing in the world, testing the change then standing around waiting for the change to be made. With three people it’s slightly better as one person can focus entirely on the code, one in the world testing it and the other person watching the in-game views closely to see what is happening. Again this a major time killer and also is very annoying. Especially with the Vive given you have to stand up and put on all the equipment just to check a 5 second thing.

Dennis expressed that this also quite exhausting because you're constantly going back and forth between virtual and real and constantly having to get up and move back and forth between the equipment and your desk.

Dr. Williamson suggested (though I had thought of it) the ability to record the motions and then play that over and over again. Actually VRTK has a stack with a VR simulator. Rather than a VR simulator could you incorporate recorded, unit test like, ghosts of yourself rather than run the HMD via the Vive. So you just record yourself doing the action of opening the drawer and then play it back as a test to see what happens.

The solution they used to the problem for the buttons was to spawn an item on top of the button that immediately dropped onto it and activated the button when the scene was started. They pressed a simply “test button when pushed” option would save a lot of time.

So ways of testing iterable objects either through pre-recorded input or a simplified version of the VR simulator that VRTK uses which you can just say “push this button” or “does this drawer work”.

There’s a lot of interesting ideas to be had here. How does script testing in VR. For example a button simply needs to be pressed so why can’t you simulate a hand pressing the button. Similarly some objects just need to be pulled so why can’t you simulate pulling the drawer via scripts. Ideally the way this would work would be to essentially record a copy of yourself making the motions and performing the tasks in-game and then being able to play those back rather than testing the functionality for yourself. Record your movements and then play them back. Actually recording in-game movement is an interesting idea for bug recreation. Simply find the bug once and then play back the recording to simulate the bug again. A find once test many strategy. How do unit tests

and automated testing work in Unity, games, VR. VR is obviously the task at hand but in general that is an interesting idea and concept.

I'll need to look into how VR testing is being done properly.

Another thing about buttons was the timings in general. So if you want the button to do something like an animation or play an audio clip or whatever one thing we have to keep in mind is that people will just spam pressing the button. So you need to make it so you can't queue up repetitions of the event being played and instead block the user from chaining together repeated events. Rather they push it once and let it play out and pushing the button while the animation plays out does nothing. These delays need to be timed as well which they set up manually.

In regards to the delays why not if it's attached to an animation have it so that it automatically detects the duration of the animation and locks the user from interacting with the button for that window. Have the option to easily configure the timing manually via some window option. Must also thing about other events though such as a unlocking and locking a door. Perhaps a press once only button (you can unlock the door but never lock it - once the button is pressed it may never be pressed again)

Option for the button to stay stuck down when pressed until event duration is over.

Scene idea: Button on the wall to unlock a door. Unlock the door and then walk through into the other room.

Dr. Williamson then brought up the issue of loading screens which result in that sigh of disgust, pain and annoyance. You know the one I mean. You could hear it in Dennis and Debbie's voice. Anyway yeah the whole loading screen issue. Debbie said they spent a very long time on this troublesome issue.

Debbie said they used a Steam plugin but I believe she meant the SteamVR script which handles loading scenes (the one I've played around with a little). She believes that this is meant to be able to go into the settings but it just doesn't. Their solution to this was to put this script with all of those commands for loading in it, directly within the scene (thinking well maybe it needs to be in the scene for some reason) because it has public variables and yet it still didn't work as expected.

Again this is the whole public variables are actually hard coded variables or being overwritten, the whole stub method thing, etc.
As Dr. Williamson stated a loading screen should be a trivial thing. A single drag and drop loading screen that takes in the parameters for the skybox, etc. and gets rid of the plane that's in there.

The idea is just to go to the compositor transition screen and it would look like something normal. Make it so that it's easy to set up the sky box and replace the texture on the floor / get rid of all the the SteamVR compositor stuff and configure it with your own stuff.

So you'd have your script similar to the SteamVR script and there's a bunch of boxes for you to drag and drop to change the default compositor stuff to make it look differently. Maybe there's a one

and done option that changes the entire scene. Probably start with “this one changes the skybox”, “this one changes the floor”, “this one replaces the Steam display thing”, etc.

Regarding what they would do differently with their next project, as they currently do not have a VR project active, would be to do more rapid prototyping. They lacked the time to do that on this project but believe that the suggested changes would allow for more rapid prototyping and experimentation which would feed into a better development experience and end product.

Prototyping would help test if a concept worked or not and being able to do this rapidly would allow for development to progress faster and be easier to make changes in the long term. Currently they hope to publish the Intern experience they have made but need to make it a bit more game-like before uploading it onto Steam. However to make additional changes and content for it now they expect would take a lot of work and time as there is no easy means to prototype an idea to see if it works and if it fits.

Additionally they would rather the loading screens be handled properly and easily as loading screens are essential to every project. Because these are the transitions and its this issue of how you transition from one scene to the next.

So this idea getting things to the point of being tested very quickly and taking care of some of the general things that are part of every project such as loading screens and testing.

Talked about the state machine. It would be nice to do something with the state machine. Dr. John Williamson wrote most of it and it's the ideal, absolutely most abstract, not developer friendly. It could be useful to use. I’ll need to look into this and ask for more details about this.

Debbie says it would have been really helpful to use the state machine as they had a timing event they wanted something to show up on and a state machine would have solved this very easily but unfortunately they couldn’t get it to work in time.

I’ll need to look into Unity state machines somewhat

Another thing Dennis suggested was how to enter and exit a scene. Individual solutions to this, so does the scene fade in and figuring out how you’d want it to fade and making this customisable. F

I think my favourite moment of the day was when Dennis sort of just spoke his thoughts aloud of “So yeah that’s a lot. It’s a lot. It’s not an easy task.” and gave me a look pure sympathy.
They liked VRTK but it quite dense.

Oh there was the issue of teleporting as well. Not just in the teleporting on top of everything sense but in the way the user interacts and moves about the space. One system uses an arc to show where the player will move to. Another uses a straight line instead. Apparently first time users had trouble with a straight line and using it to move around so they instead chose the arc solution. Just make it so you can switch back and forth. Default to one. VRTK uses point at the ground method.

Handling where the user is able to move is another interesting idea. In order to be able to not teleport on top on objects you have to attach a script. You only have to do that if the object has a collider attached to it though. For VRTK there is an exclude teleport option which means you don't teleport on top of an object. For Newton you tag the area you want to teleport onto so they had a tag "floor" which meant you could teleport onto that area of the world. You might be able to do it with a combination of VRTK and Unity to make the ground walkable but they weren't entirely sure about that.

One fun Dennis had occurred when he forgot to exclude teleport on the radio item which meant you could pick it up with one hand and then teleport onto it with the other and fly it around like a magic carpet. You don't get bugs like that in most video games these days.

In general though investigate teleportation and that in Newton, VRTK and SteamVR. Remember that if you can teleport anywhere in the floor that you shouldn't be able to teleport inside of objects placed on the floor. In general just read into teleporting and moving around. Also NPC walking paths and stuff like that.

Another of the general difficulties they face is to gauge if the scene is big enough or if it is too big. Getting the scale right has always been a big problem for them. One of their projects used gigantic scenes but ended up only making a small area to interact in. That worked out for them in the end as the way rendering and lighting works in Unity favours a very small scene.

Just figuring out scale and getting it consistent across scenes is very difficult for them. They use one Blender file to construct the scene and import that across which can be difficult to judge the scale and sense of size in two separate scenes as you transition from one to the other and ensuring it is kept consistent.
When they started with Intern the first made the room with the first attempt at the room feeling too big so they had to downscale it. After that they started importing the assets and working on the interactions.

A random problem with getting objects from Blender and Unity that they want me to fix, which I don't get a degree if I don't fix for them, is that it doesn't have the same axis when you take it from Blender to Unity. Again because Unity likes it to have its world axis. This problem might actually be a good one to solve (during non-vr work) and upload to the asset store to test that whole process of getting something published on the asset store. I'll look into that. It was worse in the past but at least now Unity tries to match and convert it.