05/10/17

GENERAL IDEA / PROBLEMS WITH EXISTING TOOLS

As has been covered the general idea of the project is to make VR development as non-technical developer friendly as possible. No coding required if possible. Drag and drop this script here and you're set.

Despite tools like VRTK claiming to be user friendly and easy to use they still use a lot of Unity terminology, require lots of configuration to get up and running and often a lot of digging into code to get things running. That isn't not artist friendly.

Newton VR was apparently HORRIBLE especially to work with for Dr. Williamson and D&D.
The code base is supposedly a mess. They had multiple versions of the project working concurrently where a feature was implemented in on and tested to make sure everything was still working and if it was then transfer it over to the main project. Even then you still would randomly fall through parts of the floor because some other object was configure with this object and that had this effect on that object other there. Oh look the floor is rising. Not non-technical friendly.

Let's simplify this process to require as little technical input from the artists and developers as possible. Ideally no coding would be required on their part. Code free. Instead of configuring multiple things how about we configure or set it up once and all that other work is taken care of on the back end It all just works and is configured for you behind the scenes. How about we just drag and drop a pre-made script and that setups up the feature that we want Dragging and dropping scripts. Development via a user interface and a few clicks Think about the time required and number of steps to add an object and make it interactable - let's make that as few steps and little time as possible.

ESTABLISHING DEFAULTS

Now the technical purists might scream at this but think of it as automatically setting up an object with a pre-established default (which I will determined). If they want more control then dig into the code and change the gravity variable value or whatever it is they want to change. But let's follow that idea of making it as code free as possible to setup the scene or desired feature with this pre-established default set of values and settings. "It all works and here is what it does"

Example: Think about enabling the player's jump in a video game. Now in Halo the gravity set for the players jump is different and more floaty / moon-like than the gravity on the other objects in the game. Now the default "enable gravity" setting for an object would be your standard item in the game world which if enabled for the player would give them that gravity. In Halo this default gravity value would be tweaked by the developer within the code to give the floaty gravity the character has within the Halo games. For most VR experiences a single gravity setting is probably fine but should creator wish to alter the player's gravity then they can by changing the code. That's a bit of stretch as an example but it's an example I used to describe it to someone and I'm going to reuse it now.

So we're hiding all that config from the user and just giving them a default setup of "Here's your item setup to work with in this VR environment". It's less customisable but its solid and it works

every time. There's none of the one-off issues you might face where something just doesn't work because of the order you configure the items or because some other item is configure in a certain way, etc. I do all the heavy lifting on the backend and provide a nice, clean, well designed UI to work with on the front end.

SOUND LIBRARIES

Another key thing to look into is the sound / audio library. VRTK might not have this and the one in Newton didn't work at all for Dr. Williamson's project. Either way there are quirks with the existing systems and mine will work properly. That can also be a key part of the project.

Example: When a ball makes contact with a surface it makes a noise. Why can't this be a drag and drop of an audio file and set under what conditions it plays?

I'll need to look into that some more but it sounds feasible. At least the simple case of attaching a sound file for the ball to make noise when it collides. Can't be that hard. And even a simple system which works is better than the competition's which is complex and broken. That complexity just makes it harder to fix and setup. My system is user friendly.

END PRODUCT

Dr. Williamson also confirmed what I initially thought about the project and that is if good enough I get to release it on the Unity asset store, as an open source project, etc. The other tools are... well I won't use the word she used but are too clunky and technical for your average artist or non-technical developer. VR is in its infancy again (try to forget its first infancy in the 90s)
The tech is finally at a level where it's not a blocky, Tron-like world. We can make that but think about the difference between the original Tron and Tron Legacy. So I'm shipping the base, core model of a Virtual Reality Development Toolkit for Unity in 6-7 months.

8/10/17

LOADING BETWEEN SCENES

One of the issues highlighted by Dr. Williamson with their application’s development was loading between scenes. One scene ends by triggering some event which transitions into a loading screen (developed by the user) which when the scene has loaded then transitions into the next scene. Simple right? Well not exactly. You shouldn't use the typical approach for scene transitions in Unity because the frame rate is at risk (kill the frame rate at a loading screen because it doesn't really matter and the resources are better spent elsewhere) SteamVR instead has a method to load between scenes which transitions briefly to the default loading space (which happens to be the SteamVR home space or something). Dr. Williamson couldn't get this to go straight to their own loading screen without first going here for a few seconds. She combat this problem by just making the entire room black. So you saw a few seconds of complete darkness before the actual loading screen kicked in. It would be better to go straight to the loading screen. I need to see how to change the default space used by SteamVR and make it easily changed if it isn't. Asynchronous loading - look into that

GUI FRONT END TOOL DEVELOPMENT

The user should be able to interact with the scripts via a GUI menu (VRTK has its own menu component like the Unity Inspection menu).
I need to figure out how to create some simple example scripts which can be interacted with via this GUI.
I've the idea of "established defaults" and so it would be great to be able to change the values in a menu opposed to digging through a script.
I essentially need to learn how to make a Unity GUI tool and get it so it's able to change some established default value of the script its working with.

CAMERA POSITIONING

Another issue I'm going to have to look at more is how the Unity camera works (in VR and non-VR applications). One problem Dr. Williamson, Debbie and Dennis faced when developing their application was they wanted to start user with the camera facing the other way after a scene transitions. This was apparently a lot harder and more complex than it should have been. This also was an example of some of the challenges faced with the general camera controls, camera placement and camera movement. So this idea of changing a camera position / perspective both between scenes and during scene setup. I'm going to have to look into Unity cameras quite a bit.

SKYBOXES

Yet another challenge faced by Dr. Williamson was changing the skybox of the scene. Some of the reasoning for this is explained below (TL;DR: Hard coded values and stub methods). The best she could manage was to change one of the skyboxes to a nice solid red. Pity it wasn't meant to be red. I'll need to look into skyboxes some more.

EVENT HANDLING & OBJECT INTERACTION

Another thing to know about is event handling and object interaction. Working with actions which are associated with objects and interactions. Interacting with objects with more precision - giant cubes are very precise. The need for more complex and detailed / fine grained object interaction. The notion of interacting with an object to tigger some event (triggering events based off some user interaction).
Example - Push a button which start and stop a timer
Example - Push a button to spawn some items in the world
Example - Make it so when to specific objects collide that a certain particle display occurs
Example - Use a key to unlock a door

MISC DEVELOPMENT THINGS

Avatar movement - teleporting and walking around a space. Rendering - just rendering in general is something I'll need to know more about (fully understand the purpose of and how to use Valve's The Labratory renderer)

TAG LINE

"It's everything SteamVR doesn't do"
Part of the problems faced with skyboxes, scene loading, etc. is that SteamVR contains some stub methods and hard coded values. In theory it should be possible to do what we are attempting to do its just that the method hasn't actually been implemented. Or we can't change that colour because despite passing in the colour we want its actually hard coded so our passed in value is

never used. Apparently a lot of the problems faced are due to lack of or poor implementation. It's my job to either fix that or make it much, much simpler (although perhaps slightly less flexible)

01/11/17

STATE MACHINE

I asked about the State machine that Dr. John Williamson produced as part of the Intern project Its as pure a developers state machine as it gets. I must stress that in its current form is a developer's state machine. Essentially it’s a scene transition state machine in that the developer sets it up so that one scene transitions to another given some event. So you've got a but on scenes and you set up the conditions and events to transition from one to the next. This would require setting up a GUI interface where the user simply sets the conditions to transfer from one scene to another. It’s an interesting idea though I fear it will not be implemented due to the time constraints put on the project. For now will be considered a "Could Do" goal in the MoSCoW planning. If not implemented can be one of the "Well they system could be improved / extended to include..." types of things for the dissertation.