



Unity VR Desktop Bootcamp Part 2

Moderate - Advanced

Sam Levine

July 15, 2017

You are...

- Familiar with Unity!
- Comfortable with:
 - Prefabs
 - Scripting
 - VR Basics
- Owner of a Vive or Oculus CV1

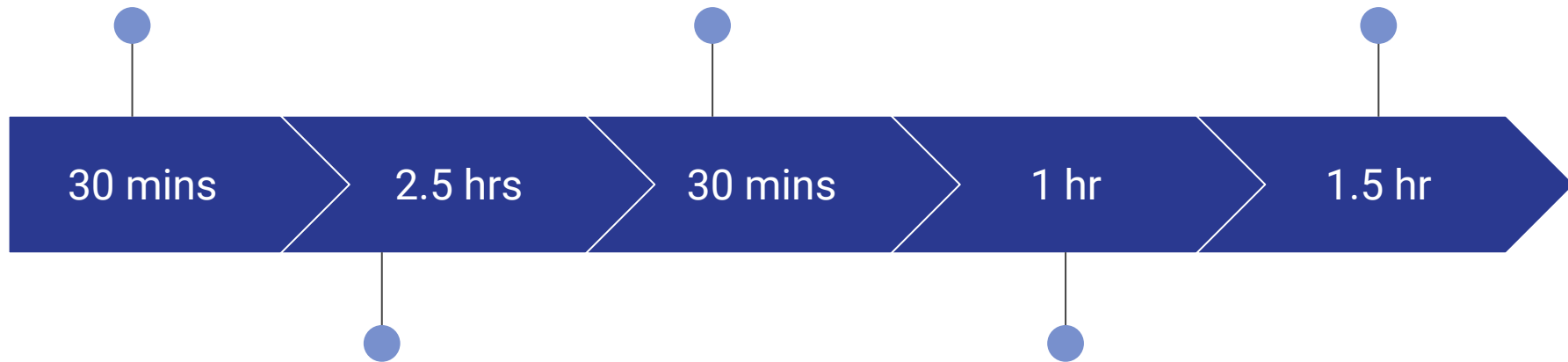
Topics

- Hardware Overview
- Hardware Setup
- SDK / Libraries Rundown
- Interaction
- Movement / Locomotion
- VR-Specific UI/UX
- Scene Transitions
- Guns and Shooting
- Optimization and Polish

1p - Introduction,
hardware overview &
setup (this part)

4p - Break! Get in
headset demo time

5:30p - Catch up,
freeform dev, or review
and wrap up



1:30p - SDKs, Intro to
VRTK, Interaction,
Movement

4:30p - General UI/UX,
Scene Transitions,
Shooting, Optimization

About Me



Hardware Overview



Hardware Setup



SDKs and Libraries

**Oculus Utilities
(OVR)**

SteamVR

**UnityEngine.VR +
OpenVR**

**Avatar Utilities
(OVRAvatar)**

VRTK

**Unity VR Sample
Scenes**

**Oculus Sample
Framework**

**ViveSoftware
(HTC.UnityPlugin)**




SDKs and Libraries: Valve

SteamVR

- Required for Vive; technically supports Rift
- "Interactions_Example" is fantastic for Vive dev

ViveSoftware Plugins (HTC.UnityPlugin)

- Get from Asset Store by searching "vivesoftware"
 - No Oculus support... :(
 - Check out instant drag-and-drop mirrors and portals (!)
- 

SDKs and Libraries: Oculus

NOTE: Oculus just announced that they're merging all of this.

Oculus Utilities for Unity (OVR)

- Required for Oculus store, can also ship to Vive (e.g. Raw Data)

Oculus Avatar Utilities (OVRAvatar)

- Required for Touch controllers
- Useful scenes - free controller models, Local Avatar mirroring & recording
- Requires copy-pasting App Id from developer.oculus.com

Oculus Sample Framework for Unity

- Great Oculus-only scenes to borrow from



SDKs and Libraries: Neutral

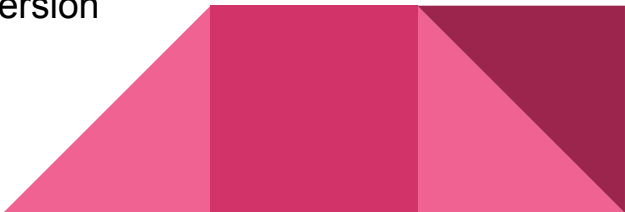
UnityEngine.VR

- Built-in, just add “using UnityEngine.VR”, make sure Virtual Reality Supported is checked

Unity VR Sample Scenes (VRStandardAssets)

- NO hand controller support, but OK for mobile-only
- Useful for borrowing elegant minimalist assets - text/GUI, gaze control, fades, cursors - scenes left in our project for perusal

VRTK

- Supports Vive, Oculus, and a **Simulator**, with experimental Daydream and “Ximmerse”
 - We’re gonna work with this today - for your own projects, grab the version from Github, NOT the asset store
- 

SDKs and Libraries: Key Points

- The two major parties have great sample scenes and assets, use them if sticking with one headset
- Minimalist Unity-only VR is certainly possible
- For game jamming and programming, though, it's hard to beat VRTK



Quick Intro to VRTK

(finally opening the project, getting in headset)



Interaction



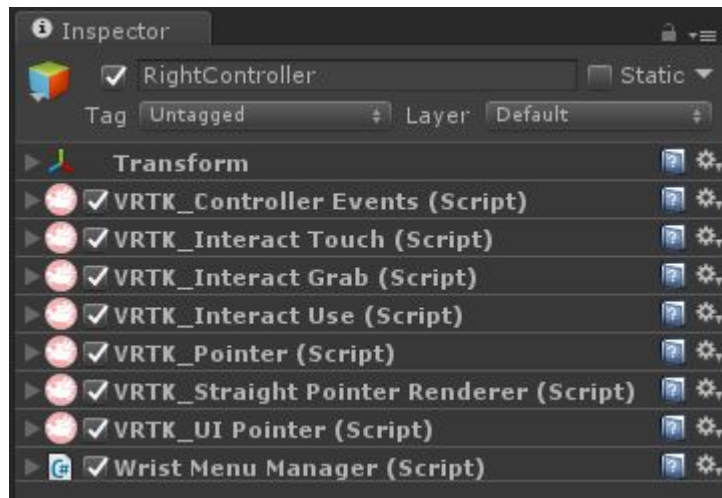
Interaction: Today's Demo

- Grabbing and using objects
- Pushing objects with hands
- Pointing with hand lasers
- Clicking with hand lasers
- Clicking with gaze control
- Wrist-attached menu controls



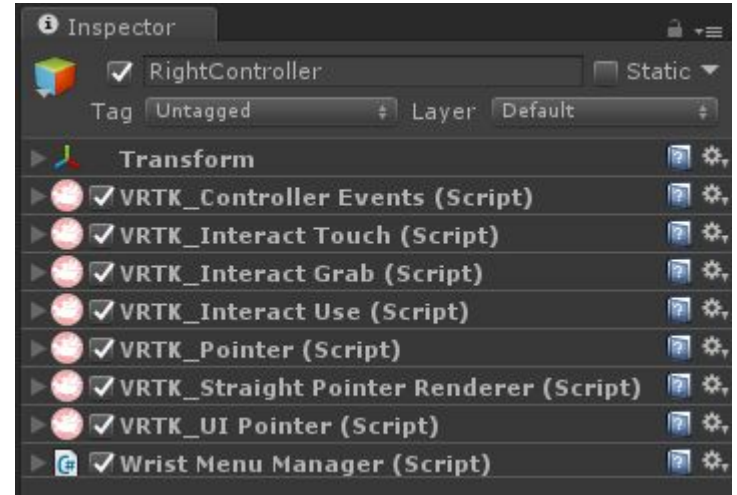
Interaction: Basic Grabbing

- Link controller objects to SDK manager
- Add first 4 VRTK scripts to each controller (Controller Events, Interact Touch, Interact Grab, and Interact Use)
- Drag out “Right Cube” prefab in /Prefabs/Interactions



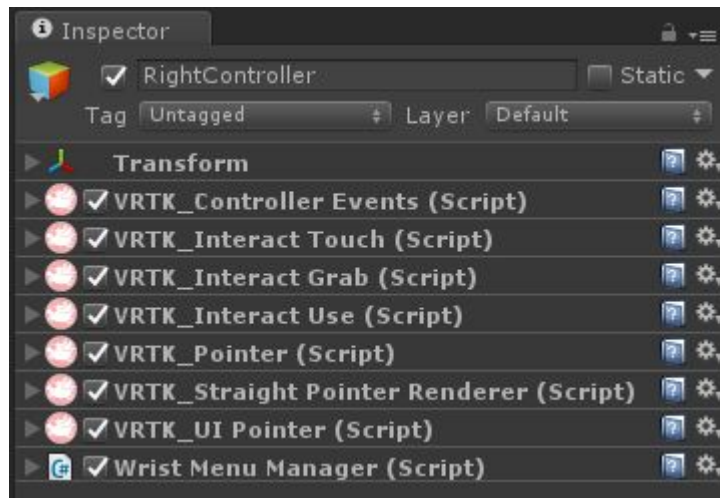
Interaction: VRTK Pushbutton

- Drag out “Left Cube” prefab from /Prefabs /Interactions and drop it in the hierarchy
- Confirm that the Button is linked to an Event Action
- Press the button...



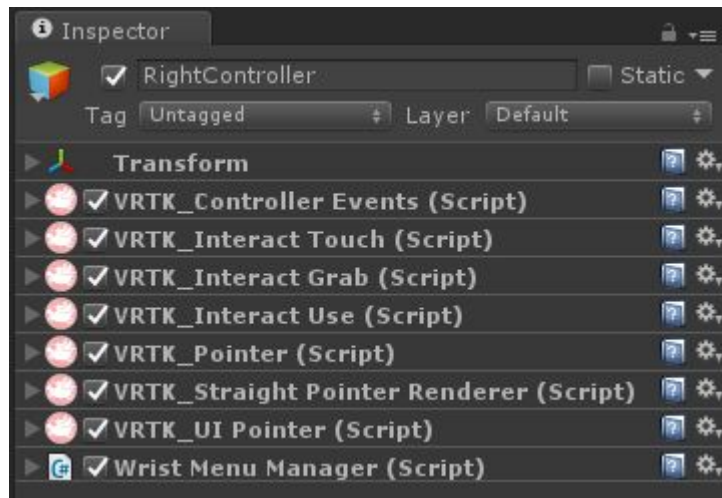
Interaction: Laser Pointing

- Add the next 3 controller scripts (Pointer/Straight Render/UI Pointer) to each hand
- Drag Straight Renderer component into the Pointer renderer variable
- Drag out HoverPanel from “Panels” into the Front Cube’s Canvas
- Assign materials to the Hover Panel (Yellow/White)



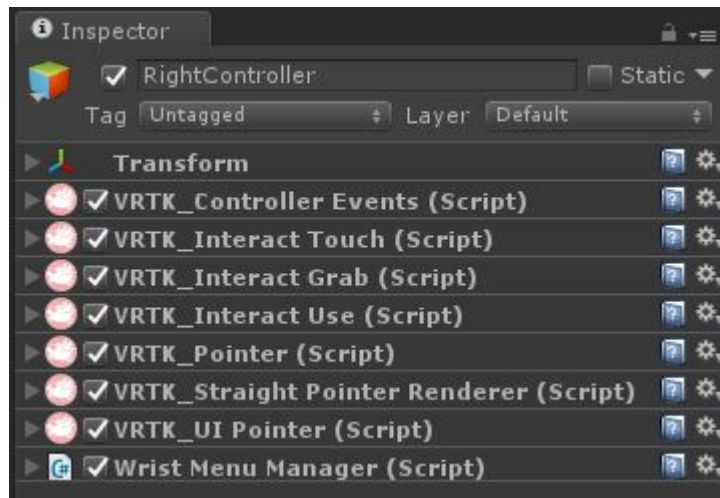
Interaction: Point and Click

- Drag out the ClickPanel prefab from “Panels” into the Front Cube’s Canvas
- Create a new Event action, drag in the KittenSpawner action, assign the Spawn Kitten method
- Click the button, and guess what happens



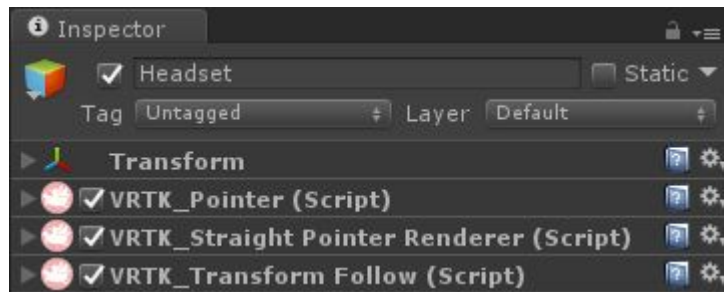
Interaction: Shooting

- Drag out the ShootPanel prefab from “Panels” into the Front Cube’s Canvas
- Toggle “Interact with Objects” in both Controllers’ VRTK_Controller Events (Script)
- Assign the KittenSpawner event again
- Shoot panel, get kittens



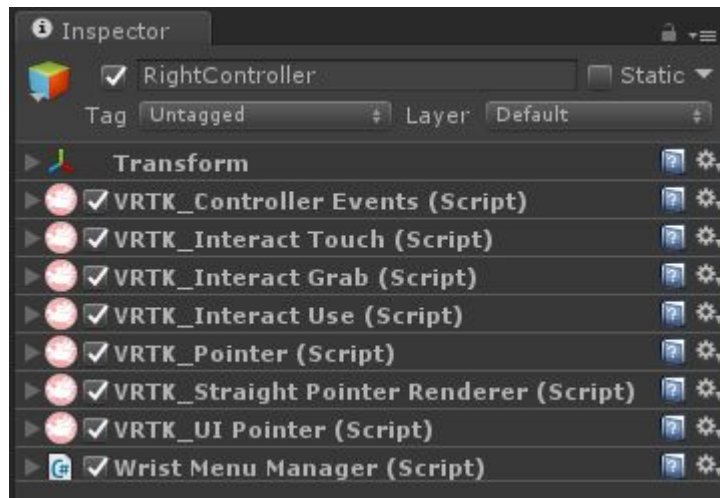
Interaction: Gaze

- create HeadsetFollower with SSK Object Alias
- attach 3 headset scripts to new Headset, have that headset follow follower (show head beams)
- In Headset Pointer, toggle activate on enable, toggle select on press, assign a controller
- drag out gaze panel (check layer)
- change straight renderer ignore layers to everything BUT gazeonly
- , change straight renderer tracer visibility to Off and Cursor to On
- , finally assign kitten event

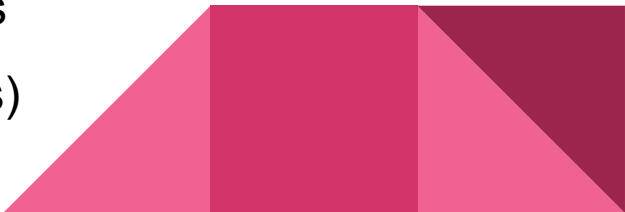


Interaction: Wrist Menu

- Finally assign the Wrist Menu Manager to one or both Controller objects
- Assign the “WristMenu_Interactions” prefab
- Make sure “Right Hand” is checked appropriately
- Hold wrist to camera, then test the button... surprise!



Interaction: What did we miss?

- Other controllers: Clicker, Gamepad, Exercise Bike...
 - Head collider (for heading soccer balls, or eating)
 - More hand control-stuff:
 - “Hitting” with objects (stick, flail, people)
 - DROP ZONES
 - Avatar-mimicked hands to allow grabbing outside of the play space
 - Mapping 3D hand position to planes/lines/circles
 - Touchpad/thumbstick radial menus (and buttons)
- 

Movement



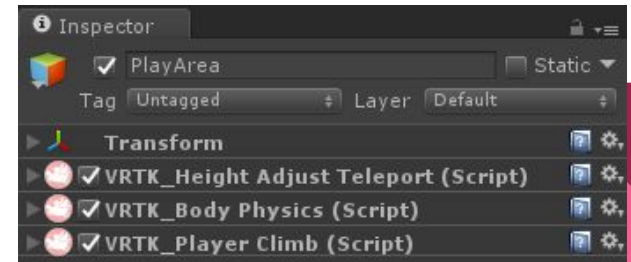
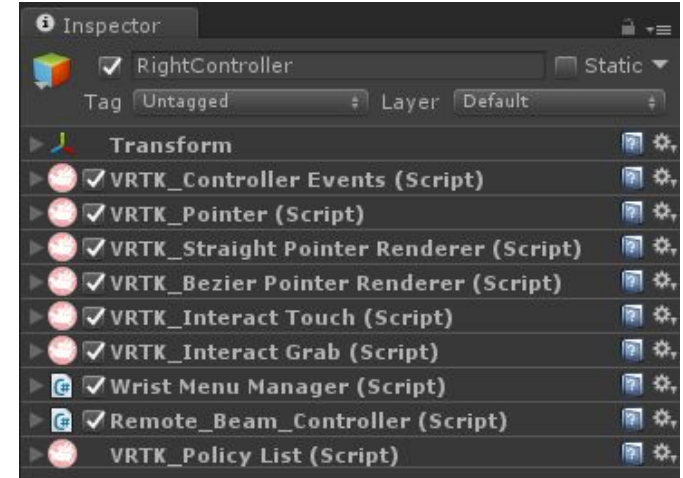
Movement: Today's Demo

- Laser pointer teleport
- Bezier arc teleport
- Wrist menu mode selection
- Remote beam
- Restricting to nodes
- Climbing / grabbing



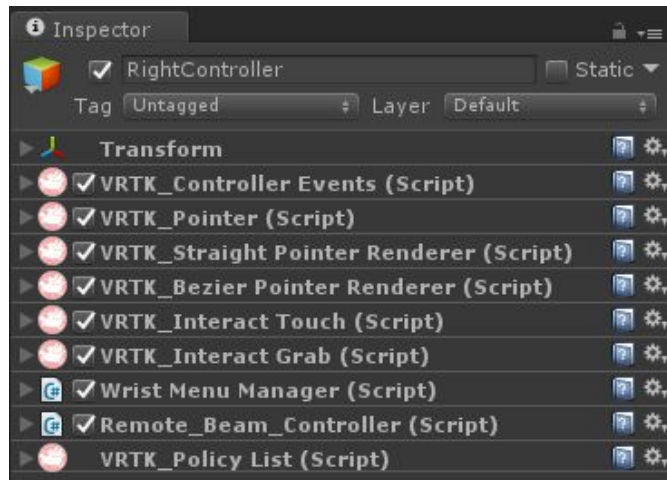
Movement: Laser Teleport

- Add first 3 VRTK scripts to RIGHT controller (Controller Events, Pointer, Straight Renderer)
- Assign the SPR to Pointer as before
- Create or find a PlayArea gameobject
- Add Height Adjust Teleport to PlayArea
- Press right touchpad/thumbstick, aim, teleport



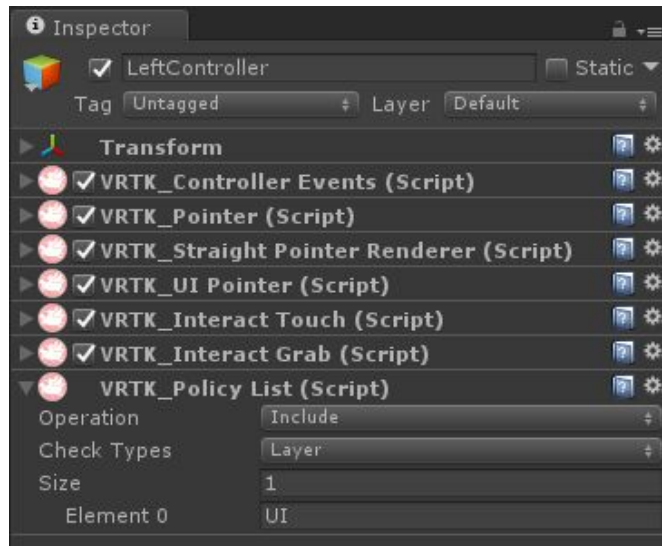
Movement: Arc Teleport

- Add Bezier Renderer to RIGHT controller
- Swap out the old Straight Renderer for the new Bezier hotness
- Press right touchpad/thumbstick, aim, teleport



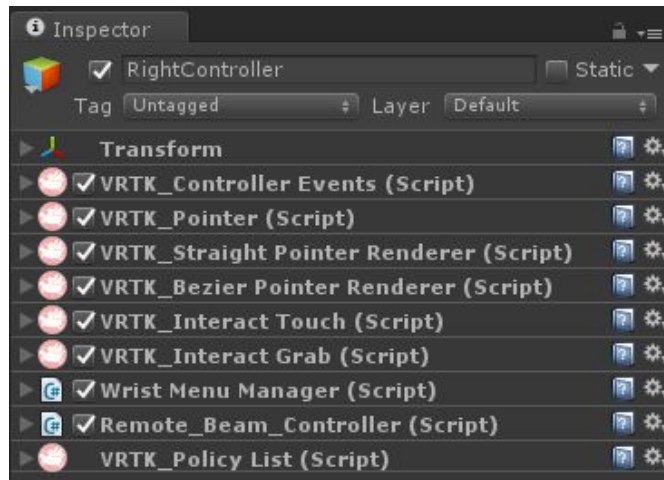
Movement: Wrist Menu Selection

- Add all components to LEFT controller, disable teleport, assign SPR, assign Policy List
- Set up the Policy list as shown (Include Layer UI)
- In RIGHT controller, add Wrist Menu Manager (with prefab WristMenu_Movement in Bootcamp folder)
- Set RIGHT pointer Renderer to null to start with nothing
- Wrist buttons should now work (except for remote beam and toggle)



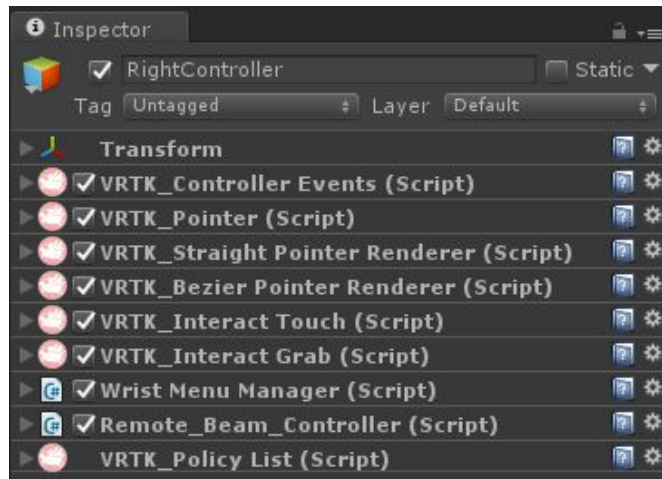
Movement: Remote Beam

- Add Remote Beam Controller script to RIGHT controller
- Assign the Remote Beam (SCRIPT, which is on Holder... fix the code if there's time) to the RBController. That script can be found on the BeamGeb object at RemoteBeam/Holder/BeamGen.
- Assign the RIGHT ControllerEvents to the Pointer on BeamGen.
- Click “Remote Beam” in the wrist menu, see remote teleport in all its crappy glory



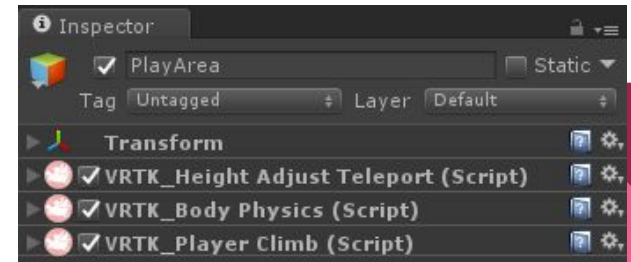
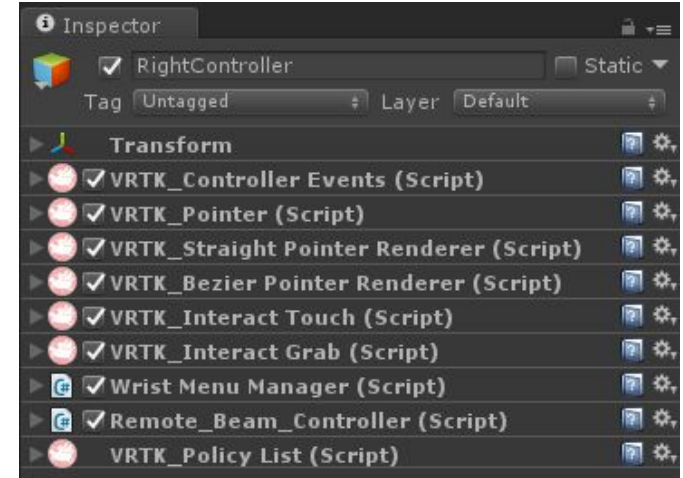
Movement: Restricting to Nodes

- Add VRTK policy list to RightController (change to Ignore, Layer, add “MazeWalls”)
- drag policy to both Remote Beam pointer and Right Hand pointer - this should stop them from teleporting into MazeWalls
- Start demo, click Toggle in Wrist Menu, see how pointers only work on the spawned objects (check out how in WristMenuMovementScript.cs for how to edit policy in runtime)



Movement: Climb & Grab

- Finally, add last scripts to BOTH hand controllers (Interact Touch, Interact Grab)...
- ...and to PlayArea (Body Physics, Player Climb)
- Drag out the Ladder and Zipline prefabs into the hierarchy.
- demo climbing + zipline (warning: nausea)



Movement: What did we miss?

- “Model village”
- “Driveable ghost”
- Head position teleport / camera swap
- Hitching a ride / rails
- Traditional thumbstick/touchpad movement
- “Armswinger”
 - PocketStrafe
- Room expansion



Break!



VR-Specific UI/UX



VR UI/UX - Frame Rate

- Traditional desktop VR dogma: 90fps or gtfo
- With modern platforms, 45fps might be an absolute minimum on Desktop
- FPS is tricky to test in Editor unless you use a reliable tool
 - don't use `Time.frameCount` (probably)



VR UI/UX - Nausea

- Nausea is a BFD, makes many (most?) VR experiences unplayable
- Physiological cause: disconnect between info received from eyes and vestibular system means evolutionary pressure to vomit to remove poison
 - Limit nausea by limiting that disconnect: minimize head rotation, player movement, and any disconnect between the visuals and movement
- We can't rely on "VR legs"

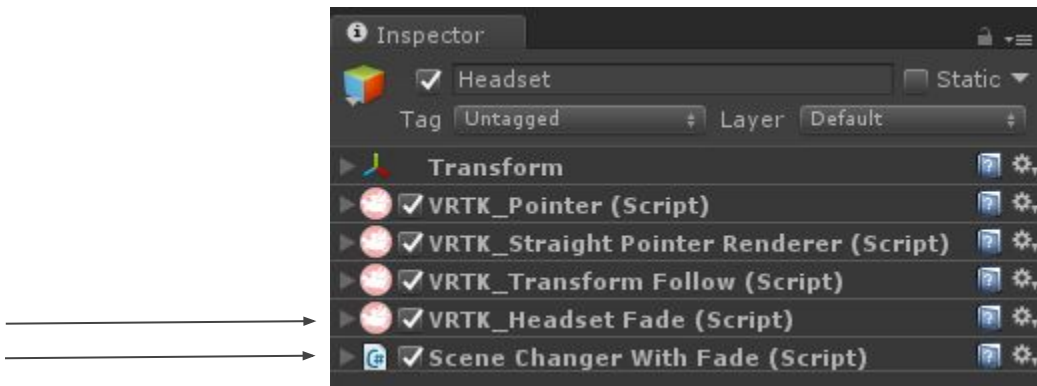


VR UI/UX - Other

- Player directed pacing- that means rest breaks, speed options, pause buttons
- Don't plaster stuff to your face - need world UI, think Dead Space
- Curved mesh surfaces are nicer than flat, can still use Unity Canvas
- Lots of points in [Sam's favorite youtube](#), but best part is the discussion of content zones based on office chair research (5:33-9:08)
- Consider fade-to-black when player head is in a mesh, rather than letting it render reality-breaking views (VRTK 011_Camera_HeadSetCollisionFading)
- For anything involving flying objects - half-gravity is apparently way more pleasant than real gravity (Project Settings -> Physics -> Gravity)



Scene Transitions



Guns and Shooting



Optimization and Polish



Thanks!

@sjklevine

Also, thanks to our hosts!

@babycastles
babycastles.com

