Full Bodied Player Research and Development

Purpose:

Detail researched, attempted, and potential methods for implementing a full-bodied player character, with fully visible / functional arms, legs, torso, etc.

Methods Attempted / Researched so Far:

1. Valem's Player Body
    a. Videos/Articles:
        i. How to make a Body in VR - PART 1
        ii. How to make a Body in VR - PART 2 : LEGS
        iii. How to make a Body in VR - PART 3 : WALK - YouTube
    b. Findings:
        i. Valem's series of video tutorials were the first items that I utilized in building the Player Body and have been heavily revisited, as they are an excellent resource. When first researching Player Body development, this was the first stop I made as they were provided by our sponsor Karan. I spent the entirety of the first semester focusing on recreating what Valem does in these tutorials. After finishing development near the end of the semester I acquired an HTC Vive to test the work done, however it became immediately apparent that some heavy debugging needed to be done. Once I finished the debugging, the Player Body still left a lot to be desired, the hands were useless as they were the hands attached to the model, the scaling was off, and the inverse kinematics being used were shoddy and unpredictable. The model's hands could be scripted to function like the default Steam player's hands, however trying to mimic that was easier done than said and most online resources didn't really offer me anything helpful. The scaling was a rather prevalent issue as the models hands would not stretch to meet the point where the users hands were, they would just go as far as they could and stop. Lastly the inverse kinematics I am using are actually an early "Preview" build within Unity, so perhaps in the future it will be updated but currently there are instances where the Player Model will do a complete 180 degree turn or the arms will shoot backwards or drop to the sides when odd movements are performed. To try and navigate around these shortcomings, the Player Model's hands were actually removed so that the default Steam hands could be locked to the models "stumps" but this didn't pan out either as they would always anchor themselves either within the wrist or too far ahead of the "stumps".
    c. Final Thoughts:
        i. Valem's videos are very helpful and in fact may very well result in a functioning Player body if some of the shortcomings I brought up are

resolved. A more competent Unity developer could most likely figure out the scaling issue, however the hands may be a bit larger problem as they would seemingly have to be done from scratch but Valem does offer a tutorial behind a paywall through his Patreon if this route is to be explored more. The inverse kinematics may be the major stopper here though as it just does not track reliably.

2. Wire Whiz Tutorials
    a. Videos/Articles:
        i. 3Point VR Body Tracking In Unity SteamVR Tutorial | Wire Whiz
        ii. VR Body Tracking Tutorial Pt. 1 | Head and Torso
        iii. VR Body Tracking Part 2 | Arms. | Wire Whiz
        iv. VR Body Tracking Part 3 | Legs
        v. How To Code Two Bone IK in Unity | Wire Whiz
        vi. How To Implement Walking in Unity SteamVR | Wire Whiz
        vii. Making Half-Life Alyx Physics Hands In Unity | Wire Whiz
        viii. Many, many others...This is an excellent source for VR Tutorials/Materials
    b. Findings:
        i. Through conducting research on how to resolve the issues discovered in Valem's Player Body, I stumbled upon this site that actually offers it's own tutorial for a Player Body. Their implementation is a bit different so I decided to give it a go, however this implementation had the same issues found in Valem's tutorials. It also did not function as well as Valem's implementation as it seemed the same issues were doubly worse (at least in terms of the inverse kinematics). By doubly worse, I meant the arm falling to the sides or swinging backwards happening much more frequently. With this in mind I decided to pivot back to focusing on Valem's Player Body.
    c. Final Thoughts:
        i. Despite the version of the player body I developed using their method being abandoned, it may very well be due to something that I missed and a more competent Unity developer could catch these errors. However this version still uses Unity's preview version of inverse kinematics, thus it may not pan out to rely solely on this method.
3. Other Helpful Sources
    a. Videos/Articles:
        i. VR Hands FP Arms SteamVR Setup - YouTube
        ii. Unity: How to parent the Vive [CameraRig] with a player model properly :: SteamVR Developer Hardware General Discussions (steamcommunity.com)
    b. Findings:
        i. These were not specific implementations but they were somewhat helpful and relevant videos/discussions for trying to develop VR hands and tying a model to the Steam VR default player. These I did not get to act on but saved for additional resources later down the road.

c. Final Thoughts:
    i. If the Inverse Kinematics kinks are worked out these resources could prove valuable in resolving the issue with the Player's hands and getting them to sync with the Player model.

Possible Future Methods / Helpful Technologies:

1. Vive Tracker:
    a. Sources:
        i. VIVE Tracker (3.0) | VIVE United States
    b. Findings:
        i. I had actually stumbled across these while conducting my own research and initially thought they functioned similarly to the Vive base stations and were just meant to provide redundancy in tracking the player's movement. However after Nevin brought up that they could potentially be used to track additional points on the users body (i.e connecting to the elbows and knees of a user) for more adequate player movement tracking, I decided to look into it further and it turns out that he was correct. These trackers can be attached to the user to provide additional points of tracking that can then be utilized in developing more accurate player movement.
    c. Final Thoughts:
        i. These trackers could subvert the need for inverse kinematics entirely, if the documentation included with them is adequate and if they can provide output to Unity throughout the duration of their use during the simulation. Which according to HTC's promotional material would seem to be the case. They are a bit pricey, at $130 per tracker for the newest model, but older models are cheaper and at a minimum to just track the arms more accurately it would only be $260 total (or $200 total for the 2.0 version).
2. TVico
    a. Sources:
        i. TVico - Interactive Android™ Box
    b. Findings:
        i. I also came across the TVico during my Player Body research, it is essentially an Xbox Kinect but it is marketed for use with VR headsets in mind and even has Unity plugins. The sensor is promoted as being able to provide complete player tracking and even has the SDKs included to do so.
    c. Final Thoughts:
        i. This option I included, since we did discuss the possibility of using an Xbox kinect and this is actually meant for VR, while I am not too sure if the Kinect actually has anything that would allow it to work with a Vive headset. However this option is more expensive than the Vive Trackers at

nearly $400 dollars for the sensor, however there is the added benefit of not having to attach more items to the user.

3. Deep Motion
    a. Sources:
        i. VR Tracking | DEEPMOTION
        ii. How To Make 3 Point Tracked Full-Body Avatars in VR (deepmotion.com)
    b. Findings:
        i. The final resource I found essentially allows one to buy a full-bodied player outright and includes everything needed to set up any model within Unity to function with their plugin. Unfortunately there is no pricing included and requires contacting their sales department for a quote.
    c. Final Thoughts:
        i. This option will basically solve the Player Body problem outright, however the cost is unknown and it could very well be the most expensive option (or it may even be the cheapest). It couldn't hurt to look into this option a bit further.

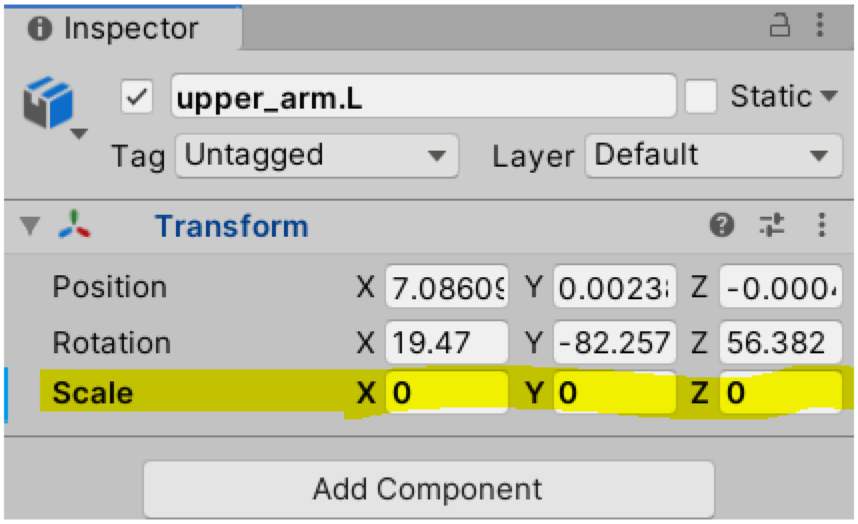
How to Utilize What is Already Implemented:

Summary:

I won't go into a super detailed breakdown of everything within Unity as it is essentially everything covered in Valem's tutorials. Thus, if a more detailed explanation is needed Valem's videos will do a far better job at explaining the ins and outs of the Player Body than I could ever do in writing. But essentially I will cover how to go about resetting the Player Model, the scripts developed, and some specifics on how to use the options that the scripts create to effectively configure a model to utilize the scripts.

1. The Player Model
    a. How to reset to original status:
        i. Currently the Player Model (Rigged Construction Player, inside Player) has been edited to remove the hands and arms as I was playing around with just using the Model without the arms. To bring the arms and hands back all that needs to be done is to change a couple of values in each limbs transform. So navigate (within SampleScene2) to Player > Rigged Construction Player > metarig > Root > hips > spine > chest > shoulder.L > upper_arm.L then change the values of X, Y, and Z for scale from 0 to 1. Repeat this with shoulder.R > upper_arm.R to do the same for the right arm as well. For the hands it is the same process but navigate a bit further, so: upper_arm.L > forearm.L > hand.L (Then of course just repeat on the right arm hand to do the same.

b. Helpful Images
   i. Image #1: Scale values within each limbs Transform to change



Inspector

upper_arm.L | Static

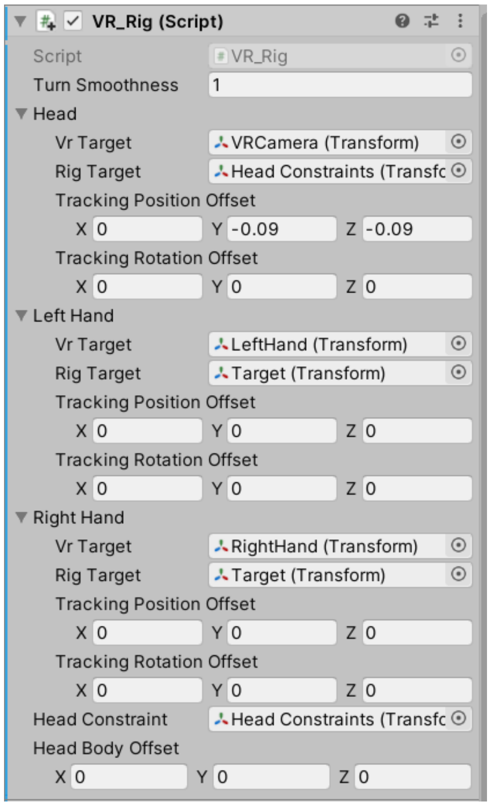
Tag Untagged | Layer Default

Transform

	X	Y	Z
Position	7.0860ç	0.0023;	-0.000.
Rotation	19.47	-82.257	56.382
Scale	0	0	0

2. VR_Rig Help
   a. Setup:
      i. As mentioned previously for a more hands-on and thorough explanation on setting up the VR_Rig script and how the back-end of the script works, I would highly recommend watching Valem's videos as they will do a far better job at explaining than I could do typing it out. I will however go over what to tweak in the front-end to get your desired results.
   b. How to utilize the scripts tracking and rotation offsets:
      i. It may be self-explanatory but the VR_Rig script includes rotation and tracking offsets, as not all Rigs are created the same and may be rendered differently than the rig used in the tutorials. Meaning if all setup is done to the letter and one begins playing, they will find that the model may be facing the wrong way or the arms are rotated slightly off, etc. Thus the rotation offsets for each limb (left hand, right hand, and head) will control its rotation, so if the head is backwards play with the rotation offsets to get it back into the correct position. The same goes for the tracking offset which is especially useful for positioning the camera as close to the model's head without going inside it (Head Body Offset), to mimic seeing from the models point of view. All that needs to be done to accomplish this is changing the X, Y, and/or Z values for tracking or rotation (use small values, 0.09 for example for tracking and degree values for rotation like 90). This script can be found on the Rigged Construction Player within the Player in SampleScene2.

c. Helpful Images:
   i. Image #1: VR_Rig View



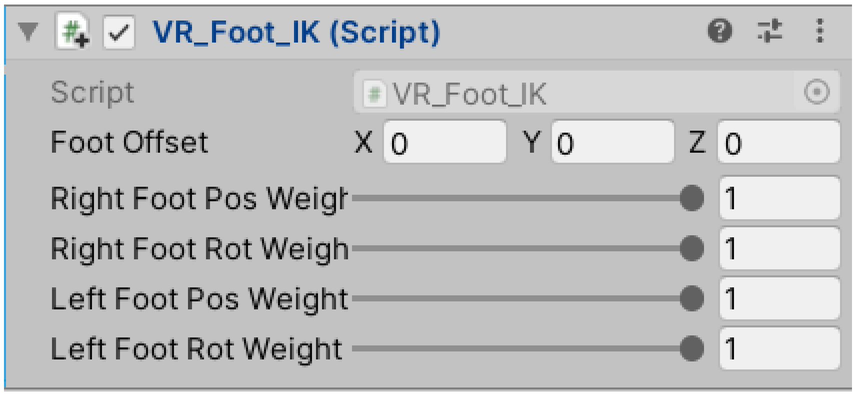
VR_Rig (Script)

Script	VR_Rig				
Turn Smoothness	1				
Head					
Vr Target	VRCamera (Transform)				
Rig Target	Head Constraints (Transfc				
Tracking Position Offset	X	0	Y -0.09	Z -0.09	
Tracking Rotation Offset	X	0	Y 0	Z 0	
Left Hand					
Vr Target	LeftHand (Transform)				
Rig Target	Target (Transform)				
Tracking Position Offset	X	0	Y 0	Z 0	
Tracking Rotation Offset	X	0	Y 0	Z 0	
Right Hand					
Vr Target	RightHand (Transform)				
Rig Target	Target (Transform)				
Tracking Position Offset	X	0	Y 0	Z 0	
Tracking Rotation Offset	X	0	Y 0	Z 0	
Head Constraint	Head Constraints (Transfc				
Head Body Offset	X	0	Y 0	Z 0	

3. Foot_IK Help
   a. Setup:
      i. Again for a more hands-on and thorough explanation on setting up the Foot_IK script and how the back-end of the script works, I would highly recommend watching Valem's videos as they will do a better job at explaining the ins and outs. I will however go over what to tweak in the front-end to get your desired results.
   b. How to utilize the scripts tracking offsets and weights:
      i. The Foot_IK script is similar to the VR_Rig script in that it has a tracking offset but it also includes weights for rotation and position. The Tracking offset functions identically to the tracking offsets in the VR_Rig script. However the weights are a little different in that they really only affect how aggressive the legs will rotate/change position (Speed and travel wise). Anything below 1 will allow the legs to bend and rotate as well as affect how far they rotate / lift. This script can be found on the Rigged Construction Player within the Player in SampleScene2.

c. Helpful Images:

i. Image #1: VR_Rig View



VR_Foot_IK (Script)

Script	VR_Foot_IK
Foot Offset	X 0 Y 0 Z 0
Right Foot Pos Weigh	1
Right Foot Rot Weigh	1
Left Foot Pos Weight	1
Left Foot Rot Weight	1