VR Modeling with hand gesture detection

Jeff Gyldenbrand,

e-mail:s202790@student.dtu.dk

Department of Computer Science,

Technical University of Denmark (DTU)

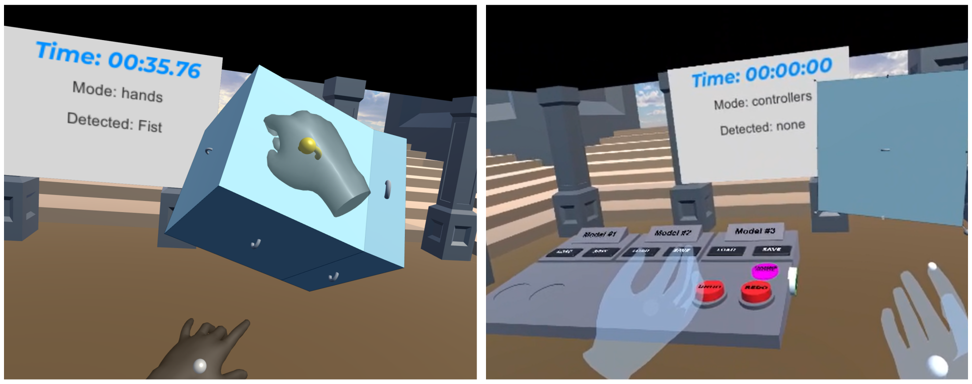


Figure 1: *Hands vs Controllers*

Abstract

3D modeling of objects in VR is largely done using tools such as controllers, where an object is shaped by holding the button while the controller is moving, I propose doing this solely using the hands so that one, has a more direct and free, and hopefully, better way to model the object.

Keywords: 3D Modeling, VR, Hand gesture detection

1 Introduction

This is an educational project done for the course *Graphics Lab* at *The Technical University of Denmark (DTU)*. The project is built upon the existing VR modeling project [Bærentzen-Frisvad-Singh 2019]. With the Oculus headset and controllers, in the original project, one can model a cube by grapping on to its vertex or face-handles. Some of the different actions that can be done includes: Single face extrusion, multi-face extrusion, face rotation and translation, vertex movement, and as seen in figure 2, face extrusion with tape measure.

This project introduces a way of replacing the controllers with hands. This is obtained by hand-tracking and handgesture detection. The motivation for using hands only is that intuitively, it must be easier to model an object with the hands rather than a tool (the controllers), as this already seems natural to us in reality.

To compare between the two modes; hands vs controllers, a way of toggling between the two is implemented. The original project has a worktable with helping functionalities such as redo- and undo actions, saving and loading of models. In this project the worktable is extended with a button that toggles between these two modes. In order to bring forward the worktable one has to rotate the left controller 180 degrees, then the user can tap the ‘hands-mode’ button.

2 Related Work and Background

This project is directly based off the project and paper [Bærentzen-Frisvad-Singh 2019]. They propose a direct signifier-based approach to the way 3D modeling is usually done. a signifier is to be understood as, e.g., a door-handle, in which has a push/pull function that the user interacts with. In their project the signifiers are face- and vertex-handles, when grapped the user performs a sequence of extrude operations, as seen in figure 2, where a face-handle is grapped and extruded towards the user.

Great inspiration is drawn from the work of ValemVR[ValemVR]. Hes work on how to recognize and record a gesture is used in this project with modifications.

3 Implementation

This project is developed using the Unity Game Engine [Unity b] version 2019.4.18f1 and the Oculus Quest 2 VR Headset[Oculus b].

When running this project in unity make sure to be in the scene: VRScene. If not, the scene is found at *Assets/Scenes/VRScene*. Scripts of particular interest are: ToggleModes.cs and GestureDetector.cs found at the Assets top level and GrapControl.cs found at Assets/Scripts/Controls/controller

The original project is using the OVR Camera Rig which comes from the Oculus integration SDK[Oculus a], this SDK comes with both controllers and hands support. In order to start tracking the hands its as simple as deactivating the controllers from the original project, then importing and activating the hands.

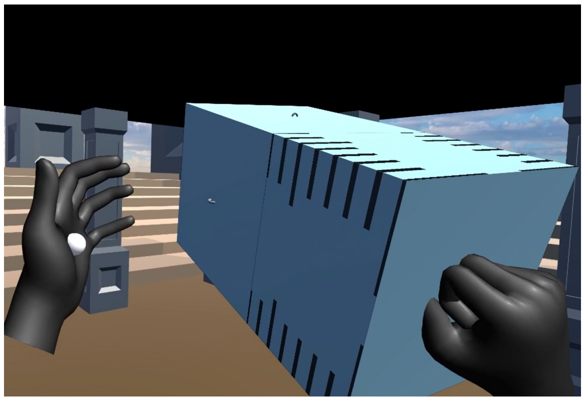


Figure 2: *Sequence of face extrusions*

In order to interact and model the cube we need some way of recognizing when we are grapping and releasing the handles of the cube. The OVR SDK provides us with alot of different information, such as the name, id, transform, etc., of all the bones in the hands. As ValemVR[ValemVR] proposes, we can take a snapshot of a gesture during runtime and save the co-ordinates of the bones, as seen in figure 3. Also note from the image, that a function to switch modes are invoked here.

There are many ways one could approach how to detect the fingers, like finding the local position or rotation of the bones. Here we are comparing the position of the fingers compared to the root of the hands, this allows for recognizing a gestures regardless of the position or rotation of the hands. In other words, a gesture like ‘open hand’ would be recognized whether its facing inwards, outwards, upside down or normal, in relation to the camera.

So initially different gestures like: a closed fist, thumps up, ok sign, etc., is recorded, named and stored as coordinates in a vector3 list, such that the *GestureDetector*-script can listen for whenever a gesture matches. A small treshold is added to allow for some slacking. Otherwise the hand gestures have to match 100% to be recognised.

The way the system broadcast the handgestures is by event delegates[Unity a] defined in the GestureDetector script which is attached to an empty gameobject named GestureDetection. Any class can then subscribe to these delegate events if needed. A GrapControl script is implemented in the original project, which handles the way a controller interacts and grap hold of the objects handles. This script is now extended to listen for which mode the system currently is in; hands or controllers, and listen for gestures as well.

Switching between modes is handled by the ToggleMode script, which is attached the an empty gameobject named Switch Mode. What the script does is in all its simplicity to deactivate and activate controllers and the hands respectively to which mode the system is in.

4 Hand Gestures

The system can recognize the following gestures right now:
Right Hand: Closed fist, Open hand, Thumps up, OK-sign.
Left Hand: Closed fist, Open hand.

If you want to create your own gestures, then follow these steps: Press play $\longrightarrow$ Switch to hands mode $\longrightarrow$ Make a gesture with your left or right hand $\longrightarrow$ Press ’Space’ with your other hand

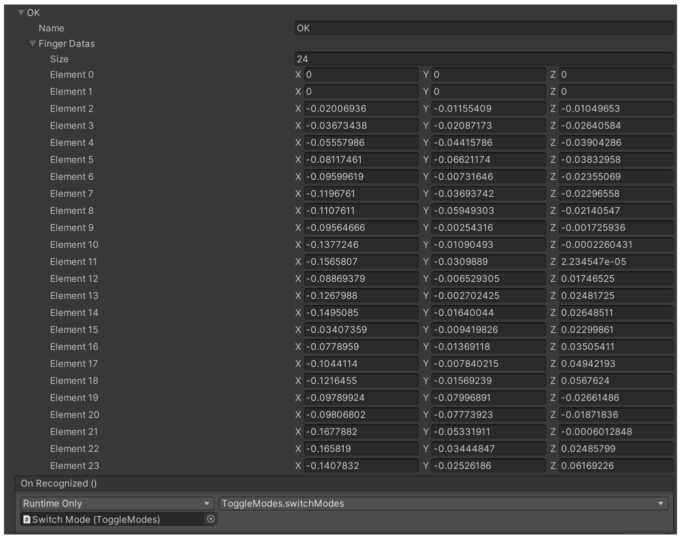
The gesture is now recorded and visible under the GestureDetection gameobjects inspector tab. Remember to rename it and copy its values before stopping application.

5 Discussion

Which approach is better, modelling with the controllers or the hands? this has been put to the test and found that the controllers are far superior still. One big challenge with hand gesture detection is that the headset needs a decent amount of light. If its too dark, it gets very hard to do anything with the hands. With a good light source the handgesture detection works okay, however still needs improvements.

One problem seems to be, when extruding the face with the tape measure, it quikly snaps into face rotation, thus making it harder to model. This is properbly due to the limitations of the hardware on which the tests was performed, and still needs to be properly testet. If the problem consist, a possible solution could be to implement a condition, that one hand is, with a gesture, locking the extrusion to being in tape measure mode as long as the other hand is grapping and extruding.

Another minor issue is that, when you are grapping a handle to perform a modeling action, right now, you need to open your hands completely in order to let go of the handle. When you open the palm of your hands, usually you have a tendenecy to slighlty move your whole hand back a little, this combined with the possibility of lagging can make it hard to model precisely. One improvement regards to this would to make a method that recognizes when a grapping hand is opening up, perhaps by measering the speed of the hand opening up, such that is releases the grap instantly.



OK
Name: OK
Finger Datas
Size: 24

	X	Y	Z
Element 0	0	0	0
Element 1	0	0	0
Element 2	-0.02006936	-0.01155409	-0.01049653
Element 3	-0.03673438	-0.02087173	-0.02640584
Element 4	-0.05557986	-0.04415786	-0.03904286
Element 5	-0.08117461	-0.06621174	-0.03832958
Element 6	-0.09599619	-0.00731646	-0.02355069
Element 7	-0.1196761	-0.03693742	-0.02296558
Element 8	-0.1107611	-0.05949303	-0.02140547
Element 9	-0.09564666	-0.00254316	-0.001725936
Element 10	-0.1377246	-0.01090493	-0.0002260431
Element 11	-0.1565807	-0.0309889	2.234547e-05
Element 12	-0.08869379	-0.006529305	0.01746525
Element 13	-0.1267988	-0.002702425	0.02481725
Element 14	-0.1495085	-0.01640044	0.02648511
Element 15	-0.03407359	-0.009419826	0.02299861
Element 16	-0.0778959	-0.01369118	0.03505411
Element 17	-0.1044114	-0.007840215	0.04942193
Element 18	-0.1216455	-0.01569239	0.0567624
Element 19	-0.09789924	-0.07996891	-0.02661486
Element 20	-0.09806802	-0.07773923	-0.01871836
Element 21	-0.1677882	-0.05331911	-0.0006012848
Element 22	-0.165819	-0.03444847	0.02485799
Element 23	-0.1407832	-0.02526186	0.06169226

On Recognized ()
Runtime Only — ToggleModes.switchModes
Switch Mode (ToggleModes)

Figure 3: *Storing coordinates of a the OK-gesture*

As suggested by Jeppe Frisvad, the hands are made transparent for better visibility to when the user is actually grapping a handle. This is not shown in the video because it was recorded before this change.

6 Conclusion

From the original VR modeling project, a way of handtracking and handgesture detection has been implemented succesfully. It is possible to model the object with the hands, further more switching between controller and hands during runtime.

Acknowledgements

The author would like to thank the advisors of this project, Jeppe Frisvad, Andreas Bærentzen and Mark Bo Jensen, and the other students of the graphics lab course for questions / feedback during

the weekly meetings

References

BREANNAN SMITH, CHENGLEI WU, H. W. P. P. Y. S. J. H. T. S. 2020. Constraining dense hand surface tracking with elasticity. *Facebook research, web: https://research.fb.com/publications/constraining-dense-hand-surface-tracking-with-elasticity*.

BÆRENTZEN-FRISVAD-SINGH. 2019. Signifier-based immersive and interactive 3d modeling. *web: https://dl-acm-org.proxy.findit.dtu.dk/doi/abs/10.1145/3359996.3364257*.

OCULUS. integration sdk for unity. *web: https://assetstore.unity.com/packages/tools/integration/oculus-integration-82022*.

OCULUS. Vr headset. *web: https://www.oculus.com/*.

UNITY. Event system. *https://docs.unity3d.com/2017.2/Documentation/ScriptReference/EventSystems.EventSystem.html*.

UNITY. Game engine, 2019. *web: https://unity.com/*.

VALEMVR. Youtube series on vr. *web: https://www.youtube.com/c/ValemVR*.