

Extended Reality – Lab3

Using and Programming Virtual Reality Headsets

Jean-Yves Didier

Lab goals:

- Getting hands on Virtual Reality headsets
- Being able to program and interact with a virtual environment.





Students requirements:

- Being able to write a program, either in JavaScript or C language.

Environment's requirements:

- Web browser supporting WebXR
-

Contents

1	Handling a Virtual Reality Headset	2
1.1	Package content	2
1.2	 Safety notice	2
1.2.1	Guardian setup	2
1.2.2	Comfort of use	2
1.2.3	Default account	3
1.3	 Preparing the headset	3
1.4	 Home menu	3
1.5	 Interacting with bare hands	3
2	Getting used to WebXR sessions	4
3	Developping your own WebXR application	4
4	WebXR target application – Tower of Hanoi	4
4.1	Design of your virtual reality application	4
4.1.1	Choice of navigation technique	4
4.1.2	Choice and design of selection and manipulation techniques	5
4.1.3	Specification of the virtual scene	5
4.2	Implementation of your virtual reality application	5

A	How to implement interactions	6
A.1	Hands on your application	6
A.2	Adding interaction	7
B	Debugging a browser session	7

Session's preparation

Documentation of Three.js API is at the following address:

<http://threejs.org/docs/>

For this, we will use JavaScript as a programming language. For the ones who are not comfortable with it, you can use the following resources:

<http://eloquentjavascript.net/>

<http://www.w3schools.com/js/>

1 Handling a Virtual Reality Headset

The VR headset that we will use is the Oculus Quest. This model has been launched in 2019 and runs using a customized version of the Android operating system. It comes with several interaction possibilities since you can interact with your bare hands or using touch controllers (one is dedicated to each hand). We will detail a little bit more what you can do with your headset.

1.1 Package content

- 1 VR headset
- 1 right hand controller
- 1 left hand controller
- 1 USB-C to USB-C cable
- 1 power to USB-C charger
- 1 user manual
- extra batteries

⚠ Please make sure the package is complete when you give it back at the end of the session!

1.2 **⚠** Safety notice

1.2.1 Guardian setup

Being immersed in a VR experiment cuts you from your surroundings. Be aware that you might bump into real items in your surroundings. It is very important to setup the **Guardian** boundaries to mark a play area, free of any objects, so that the headset will warn you when you reach them.

1.2.2 Comfort of use

It may be a good idea to take a break from time to time (every 20 minutes), especially if this is the first time you are using a VR headset. If you experience discomfort or starting to feel sick, you should stop immediately.

1.2.3 Default account

Please do not change the settings for the default account. It would make the headset unusable for the next group of students using it.

1.3 🖱️ Preparing the headset

1. Power up the headset. The power button is located just in front of the right attach.
2. Setup the guardian area
3. Connect to wifi

1.4 🏠 Home menu

Once the guardian is set up, your Oculus Quest headset will display a default environment (**home**), which is the starting point to use your headset. In your environment, you should be able to locate a menu which looks like the one presented in figure 1.

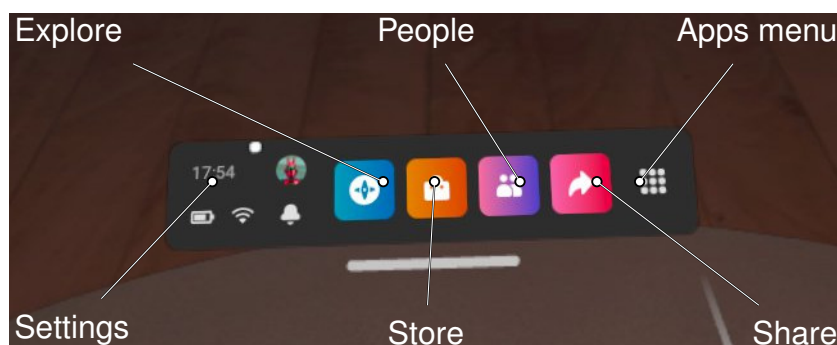



Figure 1: Oculus menu

We will not use all options from the menu. However, here is a short description of all of them:

- **Settings** gives you quick access to the settings of your headset. This way, you can reset your guardian, setup wifi and perform many other actions.
- **Explore** lets you explore the possibilities offered by your headset.
- **Store** is used to install official apps from the Oculus store.
- **People** is dedicated to network games.
- **Share** lets you record videos from your headset as well as take screenshots. They will be available to your computer if you connect your headset using the USB-C cable.
- **Apps** lets you access to all installed applications on your headset (including settings).

1.5 🖐️ Interacting with bare hands


The Oculus Quest detects both position and orientation of your hands. By default, it casts a ray coming from your hand to the virtual scene. In order to **click**, you must perform a **pinch move** with your thumb and index.

If you are lost in an immersive app with your bare hands, you can rotate your palm as if you were staring at it. A  icon should appear next to it. Then close the hand (**palm pinch**) and wait for the white circle to completely surround the icon. Then you will be **back home**. The same icon is engraved on top of a button of the right controller and has the same use when you press it.

2 Getting used to WebXR sessions

The first thing you will do is to get used to the mechanisms of WebXR sessions.

To do

1. Open the apps.
2. Start the browser (*navigateur*). The headset has its own browser simply named **Oculus browser**. If the headset asks you to update the browser, feel free to do it; the newer, the better.
3. Got to the **Three.js** website and, in the examples, select the **WebXR** keyword. All VR examples can be started inside your headset.
4. For each example, to start the WebXR mode, you will have to press the **Enter VR** button at the bottom of the webpage. Trying them will give you ideas and an overlook of the headset capabilities.
5. On your classical browser, have also a look at the source code of each example, by clicking on the  button in the bottom right corner of the webpage displaying an example.

3 Developping your own WebXR application

4 WebXR target application – Tower of Hanoi

The idea, for this session, will be to develop a virtual *tower of Hanoi* puzzle game. It consists in two pegs on which several disks of varying sizes are stacked up as you can see on figure 2. The idea is to move the tower from one initial position to a final position while respecting three simple rules:

1. Only one disk can be moved at one time
2. Each move consists in taking the top disk from one peg and moving it on top of another peg
3. No disk can be placed on top of a disk smaller than it

4.1 Design of your virtual reality application

Since this lab session has only a 4 hours duration, it is possible that you will not be able to reach the final goal. However, the idea is to put into practice many notions that you have seen so far. Before rushing to the code and an implementation of your solution, it might be a good idea to express the ideas you have in order to design your virtual reality application. This is aim of this first part that will also serve as a backbone for your lab report.

4.1.1 Choice of navigation technique

So far, you have applied a navigation technique in your VR experiments. The idea is to characterized according to what you have seen so far in the lecture track.

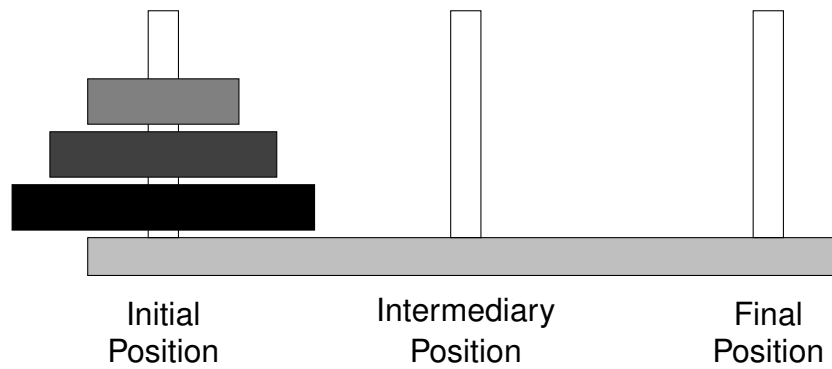


Figure 2: Tower of Hanoi in its initial configuration

➤ **To do** You will have to answer to this question: what is the navigation technique that you have previously used and that you will probably use during this lab session?

4.1.2 Choice and design of selection and manipulation techniques

In order to play the *tower of Hanoi* game, it means that basically the user has to perform many times elementary tasks that are to move one disk from one peg to another. It involves:

1. Selecting a disk or a peg;
2. Moving the disk to another peg (manipulation technique) or selecting the other peg;
3. Place the disk to its new peg.

➤ **To do** Here, you have to describe:

- The interaction techniques you want to use;
- The manipulation techniques (if you use one) you want to use;
- The feedbacks that you may put in place so that the user will be able to interact more easily (visual or aural cues, object highlighting, bounding box display, etc)

4.1.3 Specification of the virtual scene

In this part, the idea is to describe and specify what you will exactly need for your virtual scene.

➤ **To do**

- List the elements you need for your virtual scene
- Draw the scenegraph for your scene

4.2 Implementation of your virtual reality application

Once, all ideas are expressed, you can start to implement your virtual reality application.

➤ To do

For this part, the proposal is to use an incremental way to develop your application and to implement it using small steps:

1. Set up your scene graph using simple geometries (for example cylinders for pegs and disks, boxes for room and basis of your towers)
2. Implement some interaction with the mouse in order to emulate and test your selection and object manipulation
3. Setup the WebXR environment and finish the VR application prototype by introducing interaction with bare hands or controllers
4. If you have some time, include eye candy features (textures, complex 3D objects, tons of feedback,...)

📄 Notes

- Last lab session we used the simple virtual hand as an interaction technique. However, this time you can also use, if you prefer, the virtual raycasting technique. Such way of interacting with objects is described in the *three.js* example named `vr / cubes`
- Interestingly, raycasting technique can also be used outside of a virtual reality headset. For example, when you click on the scene with your mouse it can be seen as if you are projecting a ray emitted by the camera and passing through the pixel you clicked with the mouse. An example of such a thing can found in the documentation of the *three.js* prototype named *Raycaster*.

The idea, for the session, will be to develop, through the codelab¹ on your desktop browser and then test your simulation inside the Oculus browser.

A How to implement interactions

A.1 Hands on your application

The Oculus Quest headset is able to detect and track hands. It is now the time to try to interact with bare hands in your virtual environment.

To do

1. Study the source code of the example named `vr / handinput`
2. Display your hands inside the virtual environment

Notes

- Scripts to handle hands and controllers used in Three.js WebXR examples are provided inside the `js` folder of your project
- Several hand models can be used to display them. The `createHandModel()` method takes two parameters, `controller` and `profile` (by default this parameters equals to `spheres`). Possible values for `profile` are :

¹<https://deptgi.iup.univ-evry.fr/codelab>

- `spheres`: all hand joints are displayed using spheres
- `boxes`: all hand joints are displayed using boxes
- `mesh`: hand is displayed in a realistic manner

A.2 Adding interaction

Now that we can display our hands, we would like to add some interaction with the scene. The interaction we will program is the simple virtual hand. We will add an object and the idea will be to change the color of the object when it is touched by hand.

To Do

- Add an object inside your virtual scene, for example a sphere, located in a place you can reach
- Detect that the index finger tip is touching the object. To do so, you will have to detect if the position of the finger tip is inside the bounding box of the object. In order to show it, change the color of the object.
- Once, you are able to change the color of the object, we will transform it as an exit object: when you touch it, it will be the end of the WebXR session.

Notes

- Check the relationships between `Box3`, `Object3D` and `Geometry` Three.js prototypes
- Hands created by the `XRHandModelFactory` add a `joints` property behaving as an associative array in which all joints are available as independent `Object3D`. The list of available joints is provided in listing [1](#)
- Session can be accessible through the `xr` property of your renderer.

B Debugging a browser session

The Oculus browser can be accessed using Google Chrome. To do this, you must:

1. Connect the headset to your computer using a USB cable
2. Launch Google Chrome on your computer
3. Open webpage you want to debug in your headset
4. Open address `chrome://inspect#devices` in Google Chrome on your computer:
 - Then **inspect** the device you want to debug
 - Developer tools will be displayed to debug headset browser

```
const joints = [  
  'wrist',  
  'thumb-metacarpal',  
  'thumb-phalanx-proximal',  
  'thumb-phalanx-distal',  
  'thumb-tip',  
  'index-finger-metacarpal',  
  'index-finger-phalanx-proximal',  
  'index-finger-phalanx-intermediate',  
  'index-finger-phalanx-distal',  
  'index-finger-tip',  
  'middle-finger-metacarpal',  
  'middle-finger-phalanx-proximal',  
  'middle-finger-phalanx-intermediate',  
  'middle-finger-phalanx-distal',  
  'middle-finger-tip',  
  'ring-finger-metacarpal',  
  'ring-finger-phalanx-proximal',  
  'ring-finger-phalanx-intermediate',  
  'ring-finger-phalanx-distal',  
  'ring-finger-tip',  
  'pinky-finger-metacarpal',  
  'pinky-finger-phalanx-proximal',  
  'pinky-finger-phalanx-intermediate',  
  'pinky-finger-phalanx-distal',  
  'pinky-finger-tip',  
];
```

Listing 1: List of available joints described in a hand model