**Part 1**

**Question 1.1**

**Oculus Go + Netflix app**

1. *How does the app present information to the user?*

Simulated living room with a TV in front of user, where the Netflix app environment runs.

1. *How does the app allow the user to interact with it?*

Point and click system, while stationary sitting position is assumed. User can look around the room, but cannot move.

1. *Do you feel immersed?  Why or why not?*

I’d say the experience is more immersive than watching Netflix on a laptop, but less then watching movies in the cinema. The images aren’t very sharp, it ruins the feel of the movies and isn’t too comfortable to watch.

1. *Did you feel any negative physiological responses to using the application (e.g., nausea, headache, dizziness,...)?  If so, do was there something particular about the app that you think caused it?  Note: If you do feel these things you should stop using the headset for a bit and/or sit down while using them.*

No, but after some time, I maybe would get a headache from the bad quality.

1. *What did the app do well, and what could be improved?*

I like the environment (living room) and idea with the TV screen in it (as opposed to floating window in YouTube app), but the headset cannot display the content in good enough quality to get a good viewing experience.

**Oculus Quest + Superhot**

1. *How does the app present information to the user?*

Not very well, but it may be the intention. It doesn’t give user much information about controls, goal of the game, etc.…

1. *How does the app allow the user to interact with it?*

VR headset allows the user to look around the rooms, while remotes serve as hands which can punch, grab weapons, use weapons. 6DOF on headset allow movement through the rooms.

1. *Do you feel immersed?  Why or why not?*

Yes, the whole vibe of the game immerses the user and the control over movement, weapon movement, time gives user real sense of danger and control.

1. *Did you feel any negative physiological responses to using the application (e.g., nausea, headache, dizziness,...)?  If so, do was there something particular about the app that you think caused it?  Note: If you do feel these things you should stop using the headset for a bit and/or sit down while using them.*

No.

1. *What did the app do well, and what could be improved?*

Sometimes when I tried to punch oncoming attackers, the attack didn’t really work and I got killed, which was quite frustrating. I’m not sure if it was my fault for maybe punching too fast or if the tracking wasn’t quite calibrated.

**HTC Vive + Tilt brush**

1. *How does the app present information to the user?*

Floating menu, point and click to choose. Inside the game, one controller has a 3-sided palette where you point and click to choose drawing method, background,…

1. *How does the app allow the user to interact with it?*

One remote is used for selecting drawing options and how the world around is shaped, second remote is used to draw. Headset has 6DOF and so do the remotes, they are very precise.

1. *Do you feel immersed?  Why or why not?*

Yes, the accuracy and the 6DOF add a lot to the feeling of being immersed. Only distraction is the cable at the back of the headset.

1. *Did you feel any negative physiological responses to using the application (e.g., nausea, headache, dizziness,...)?  If so, do was there something particular about the app that you think caused it?  Note: If you do feel these things you should stop using the headset for a bit and/or sit down while using them.*

I did not.

1. *What did the app do well, and what could be improved?*

App is easy to control, I love the menu on the secondary remote which controls painting options. The way it’s 3-sided is very smart, as it is easy to use and able to provide a lot of options without the menus being cluttered.

**Question 1.2**

1. I don’t feel that the browser requires additional 3 degrees of freedom, it need just a simple point and click tool and that is effective enough with 3 DOF. This is one app where the Oculus Go does just as good of a job as the other two systems.
2. The starter project does not necessarily need more DOF either, although it does add to the immersive feeling of the app. It is nice to have the freedom and not be limited, but effectiveness does not get affected.
3. There may be compatibility issues that arise, for example, it may not always be a good idea to create generic simple apps with support of 6 DOF, as older and simpler devices cannot run those.
4. Because apps developed for the Oculus Go will work on the other devices as well, the forwards compatibility is okay unlike the backwards one.

**Part 2**

**Question 2.1**

1. Git provides a great tool for having control over versions of a project. It keeps a history of all updates and changes and it also allows for multiple people to collaborate and work on the same project and even same files, thanks to the merging tools. With the use of its integrated continuous integration, the project can be automatically compiled and tested as well.
2. Staging area is a cache of files that we want to track and commit with git. To add files to the staging area, we ‘git add’ them. It’s important because we can fine-tune our commits, we do not need to commit everything in our working directory, sometimes there are unfinished feature updates which require more thought and work before committing them.

**Question 2.2**

1. In the Chrome browser on my laptop, I can only see one instance of the scene, only one set of squares and I’m unable to turn around and move. By using the mouse buttons, I can use the app. In VR, I can turn around and see the scene in all directions around me (displaying the same configuration), I interact by pointing and clicking the main remote. The menu remains in bottom right corner as I move (it moves with me in my field of vision).
2. The way I interact and am stationary within the room is the same.
3. The laptop interfaces is enough for this purpose, since I only need the one scene and it fits my field of vision on the computer screen just fine, it’s sufficient. The VR interface adds more of a sense of being immersed in the room, being able to look around and of course it provides interesting possibilities of expanding upon the project and moving around the room.

**Question 2.3**

With the ‘Watch’ section in the debugger, we can select variables which we’d like to keep track of as far as their values go. This comes in handy especially in situations where we’re not sure if our program is doing exactly what we think it is with a certain variable, this is an easy way to check the content or address of them.

**Question 2.5**

1. Commenting out both of the lights leaves the scene dark except for the white field of squares and grey action board, those stay the same. Adding only the point light adds a light source somewhere near the top of the scene, because we are suddenly able to see the title Bresenham’s Algorithm and the room is more dimly lit. Adding only the ambient light has a similar effect, but everything is even dimmer and quite uniform, everything has the same amount of light. Combining both light sources gives us a clear view of the title while the room is also quite well lit. The ambient light adds light evenly to all surfaces. The point light adds directional light rays from it’s position, which is set near the title, that is why we can observe the above mentioned behavior.
2. Changing the x-axis moves me to the left (positive) or right (negative) while still facing the same direction. Changing the y-axis moves me down (positive) or up (negative) while still facing the board. Changing the z-axis moves me closer to the board (positive) or further away from it (negative). When I go close enough, I can pass through the board and it appears behind me.
3. I would have to add a newly created GuiVRButton to the buttonList of the menu. The boards.map maps an action to the press of that specific button. It is in the form of a function, which is called when the button is clicked.
4. Global Listeners -> click -> handler first calls function onSelectStart(event).

**Question 2.6**

1. Member variables: brushColor, castShadow, children, clicks, collider, ctx, dim, editMode, frustumCulled, guide, layers, matrix, matrixAutoUpdate, matrixWorld, matrixWorldNeedsUpdate, n, name, parent, position, quaternion, receiveShadow, renderOrder, rotation, scale, stride, texture, type, up, userData, uuid, visible, eulerOrder, id, modelViewMatrix, normalMatrix, useQuaternion, \_\_proto\_\_
2. Line 40 calls ‘this.reset()’ which leads us to line 124 in function ‘reset’, which sets the color to white.
3. It sets the color for the square at coordinates x and y. The color is first calculated from an 3D RGB vector into decimal numeric system and then converted into a hexadecimal numeric system string and concatenated after a ‘#’.
4. It checks if the program received enough information from the user (enough clicks) for the given mode it is in and if so, it completes the given rasterization. E.g. point mode does not need any additional clicks, so it just draws the point right away. Line mode, however, needs two selected points to draw a line, so the program waits for the user to make two clicks before rasterizing the line in this mode.
5. From Object3D (<https://threejs.org/docs/#api/en/core/Object3D.position>).
6. I would create a Rasterizer class, which would be a member of the Board and would handle all the rasterization tasks. Upon construction of the Board object, its member rasterizer would be instantiated and from that point I would use it to draw on the board.

**Question 2.7**

1. Board inherits the collider member variable from the GuiVR class. This variable represents the rectangle which shows the shape of the board and is used when finding out if ‘click rays’ have intersected with the board and the board should perform adequate tasks upon that click interaction.
2. The event of clicking the mouse is being handled by the ‘onSelectStart’ function. To register the click into the GUI, ‘intersectObjects’ is called. The first element the ray collides with is found within this function and the element’s ‘collide’ function is called. Since I was in point mode, the collide function calls the point rasterization as ‘rasterizePoint’ which fills in the point with ‘writePixel’.
3. Abstract classes cannot be instantiated and they are typically used for creating subclasses. They provide common functionality across several similar classes, but they allow for different specific method implementations. The reason why GuiVR benefits from being an abstract class is that there is no reason to adjust the interface of the GuiVR for different types of hardware, it will be compatible and the implementation will be in the subclasses themselves.
4. **Extra Credit:** What would be better documentation for the Board.collide function?  Make sure to discuss the parameter's types, meaning, and the function's overall behavior