**VR Hide & Seek**

Kyle VanSteelandt

<https://youtu.be/siKKywXmck8>

**Overall Objective of the project:**

The goal of this project was to create a fun Hide & Seek game using the Oculus Quest VR headset. The project focused on the Players interaction with the virtual world through the physical actions of grabbing, throwing, crouching, crawling, walking. The Seekers in the project were to provide a fun challenge for the player while following the basic rules of Hide & Seek.

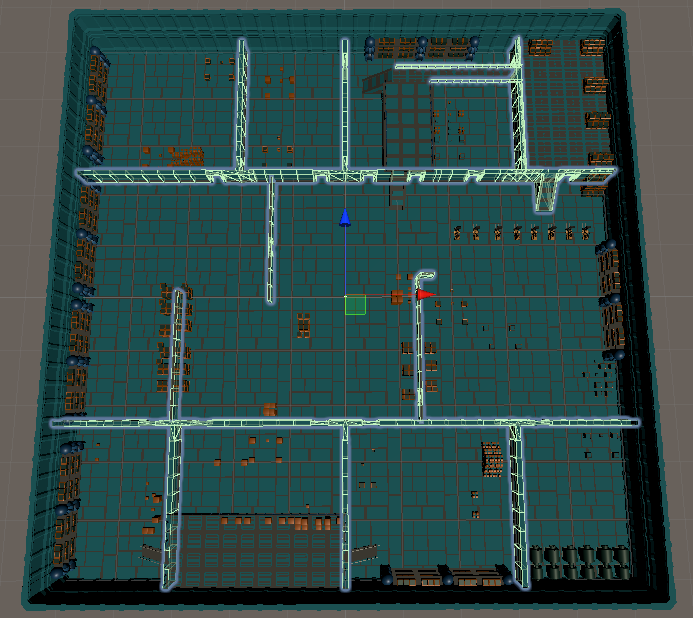


Figure 1:Warehouse, Player Starts in the center

**Technical highlights:**

This project has a better Camera constraint than the current Oculus Integration package. The ColliderFollowPlayer.cs script uses the height of the player to shrink the player collider which allows the player to get through smaller holes and escape the Seeker Bots. Players can interact with the environment by grabbing and throwing boxes. By hitting the Seekers with thrown or placed boxes the Seekers will become stunned. The player can use this to get away toa better spot or stop the Seeker from catching the player. When this collision occurs, the box is crushed and makes a loud noise which can be heard by other nearby Seekers.

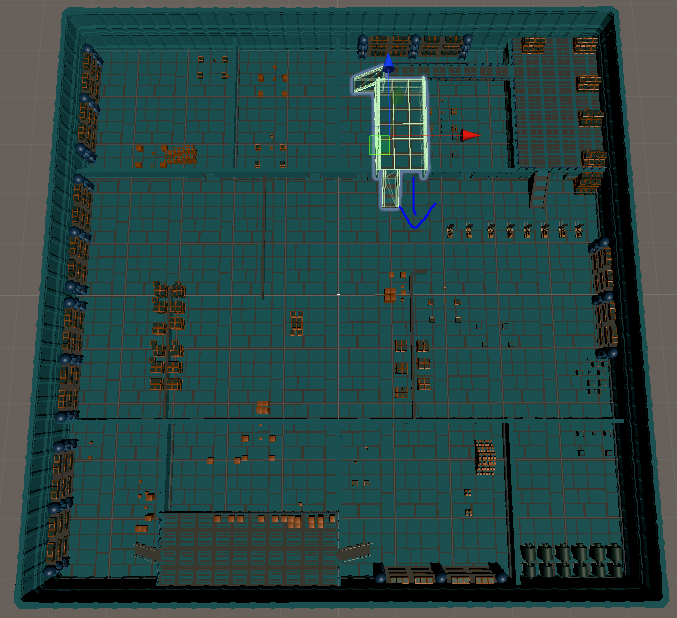
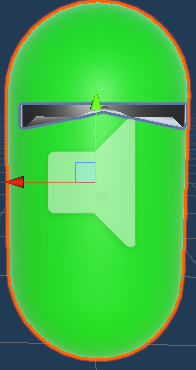


Figure 2: Bot Bay, Seekers start inside and count to 30 secs! Just like Hide and Seek!

Seekers have 6 different states that they enter once the game starts:

1. Wandering: The Seekers have no real purpose in this state they are given a large sphere and they select a point to move to at a slow speed. The Seekers will stand at the new location for a set amount of time and then they move on to the next. The Seekers will be Green at this time.



1. Chasing: The Seekers have a visual of the player the Seekers vision cone originates from the sunglasses. At high-speed the Seeker will move towards the player. This state is always active when the Seeker sees the player. The Seekers will be Red at this time. They will also shout “ACHTUNG!”



1. Searching: The Seekers lost sight of the player. They will move to last seen location at medium speed. They have a smaller sphere in which they select their next location. If they haven’t seen the player in the last 30 seconds they go back to wandering. The Seekers will be Blue at this time.



1. Stun: The Seeker has collided with a box. (This happens sometimes even if the player didn’t throw the box) The Seeker is frozen for 7 seconds and loses all “visual” and “hearing” capabilities. (i.e., the player can stand in front of a stunned Seeker and the game will not consider the player “Found”). The Seekers will be Yellow at this time.



1. Investigate: The Seeker is within the hearing radius of another Seeker getting stunned (Colliding with a box). The Seeker will go to the point the collision happened and then search around for the player. The Seekers will be White at this time. They will also shout “Kein Durchgang!”.



1. Found: The Seekers is within 2 units of the player. (This state is currently bugged as the WaitforSeconds function is not waiting at all.) The Seeker will turn Black at this time and the game will (unintentionally) instantly restart. The Seekers are supposed to laugh at this time.



**Differentiating factors compared to existing programs/previous work/commercially available programs:**

The AI of the Seekers gets much of its inspiration form the Tom Clancy’s: Splinter Cell games where the last seen location is investigated and then a small area around the location is searched.

Previous homework assignments learned skills and code used in the project:

Homework 1: Building of the level and making the model that is used for the Seekers.

Homework 2: The UI from this project was expanded on and attached to the player hands using Homework 2 as an example.

Homework 3: The BuildNavMesh script used in this assignment is also used for the Seekers, this allows them to navigate through boxes which are scattered around by the player instead of just colliding with the boxes constantly.

**How do you measure the success of your project?**

From the project proposal, the success was based off the projects ability to take the fun of real world hide and seek and transfer it to the VR realm and make the player feel suspense, surprise and involve some physical exertion.

I believe that this project was successful in all those categories as I have experienced each of those emotions. Surprise when a Seeker comes out from a room behind you, or when crossing the bridge and getting surrounded. Suspense when a Seeker comes into the room, pauses for a couple seconds and then turns around to leave, not seeing the player. Physical exertion from crouching to get through the smaller doorways and throwing box after box trying to keep the Seekers from getting too close. There were some fun tests that I had when 9 Seekers would chase me and I was throwing every box trying to get away, only for the last Seeker to be standing in the first room I enter.

**What you would do next if you had more time?**

Because of difficulties getting the project to build the android SDK in the Unreal engine it was moved over to Unity 2020.1.6f1. This required that the Seeker Bot AI be rebuilt, and a new Seeker model needed to be made. This engine change also resulted in the loss of the Unreal animations.

Using Unity 2020.1.6f1 came with another challenge which was only found and understood late in the development, new changes to the Oculus integration package broke the “OVR camera constraint” script, it no longer works properly and even after fixing the code to call the proper classes it still was causing issues as it forces the camera to be above the collider. In the end, the camera constraint script ended up needing to be completely redone. The version submitted today (December 9th) currently has the players physical movement code commented out as it still needs work and currently causes motion sickness for the player as turning the head causes the character to move faster and farther than the player physically walking. Going through this process of getting the player camera to follow the player gave a lot of insight into how to work with player interaction in VR.

If the player movement had been working properly from the start I would have like to try and add something like what was discussed during the” Improving Obstacle Awareness to Enhance Interaction in Virtual Reality” presentation, where the game (I believe the example was Fallout 4 VR) would pause and turn green in order to inform the player that they need to turn around. Also, the topic of curving the players path in order to give the player the sense that they traveled in a straight line when the player was really walking in circles. These are two things that I had in mind to tackle the issue of the player moving out of the “Guardian Space.”

The “A Comparative Analysis of 3D User Interaction: How to Move Virtual Objects in Mixed Reality” paper also has a “highlight” object feature that I would also like to add to the boxes in the warehouse. This is because it would allow me to visually show what can be interacted with and allow for more variety of objects that could be used to create a better-looking level.

**What you would do differently next time?**

If I had the opportunity to start this project again, I would have started with Unity and never bothered with trying to make bipedal robot animations for walking, running, standing, and sprinting. I also would not have started with the camera constraint if I had known from the start that it was not working as intended. I had created the dynamic height setting after noticing that it was no longer available, but I missed the fact that the collider doesn’t follow the player because a lot of the initial testing was done at my computer not in the large area I had made for the project. Once all the Seeker AI was handled, I started testing in the large area and then it became evident that it wasn’t following. Next time I develop in VR I would also like to spend some time making a “Developer Mode” UI that would print out my debug statements. I think if I had that a lot of my issues would be found and solved much faster if I had created something like that during the semester.

**Assets Used & Works Cited:**

Voice lines of the Seeker Bots are all from the video game Wolfenstein 3D:

Wolfenstein 3D  
Developed by id Software in 1993

Unity Assets:

SNAPs Prototype Package

ProBuilder

ProGrids

DustyRom CasualGameSounds – Crate impact sound

NavMeshComponents

Forkloader Mesh