Team Members

Joseph Tibog

Tanish Arora

Jiahui Huang

Ran Liu

CS135 Project

Subway Runners

Abstract (1 point, ~1 paragraph)

This project is inspired by the original Subway Surfers which was an endless mobile runner game co-developed by Kiloo and SYBO games. Subway Surfers was the best game to play when we were bored, had nothing to do, and didn't want to invest in a serious story. Bringing this into the VR setting while retaining the basic gameplay and adding more VR mechanics such as dodging, shooting objects etc., will make the player physically active rather than just sitting in bed all day and doing nothing.

Introduction (4 points, ~1 page)

This project is about reinventing the popular mobile app Subway Surfers so that it can be played in VR. It is pretty much our reinterpretation of the popular game and the VR application attempts to reinvent the same mechanics of the original game but with better functionalities. The game uses the Oculus Player controller so that instead of using buttons and swiping movements like on the mobile version, we simply track the player based on their position in the real world. What made this project interesting was that instead of being constrained to the usual three lanes that the game provides, the unique capability of position tracking is that it doesn't need to be held down by this constraint which allows us to place obstacles in more realistic positions. I believe the project is interesting because it utilizes the infinite generation of the map which the original game uses but because we accidentally utilized depth perception, we actually were able to make the railway tracks a de facto mipmap because the resolution changed accordingly based on the player's position so that the resolution would be clearer once the player approaches a certain position. This makes movement more intuitive and it indirectly allowed us to avoid experiencing vection since it appears that the game is running at constant velocity. Even though the velocity changes over time, the utilization of the de facto mipmap made the discomfort of changing velocity disappear. What we also found interesting was that the player didn't actually have to move right away like they had to in Subway Surfers and since the room appeared big based on the player controller's perception of the map, we were able to make it much easier to dodge obstacles despite the fact that they are not in fixed lanes. This VR experience makes the game feel big due to the scaling and so it makes it more immersive as each obstacle in the game is easy to track down. Movement in the real world is also slow because of the nature of how obstacles look like as the player moves closer to it and therefore, safer to play even with a lot of obstacles in the real world. The nature of how the world was designed also means that this game is welcoming to new players and it allows them to see the world in a much more realistic size. The thing with Subway Surfers was that the world felt small and that the kid running in the game was too tall when taking the train's height into account. The VR experience makes the player feel bigger and so it seems like they are actually dodging obstacles in the real world. This therefore increases immersion. Overall, the casual difficulty and the design of the game makes

VR Subway Runners an ideal type of game for those who aren’t trying to work out and just want to stretch their body without having to move too fast.

Related Works (2 point, ~1 page)

Our project was inspired by a number of different games ranging from Beat Saber to Roller coaster Simulations, but the main source of inspiration came from the classic game Subway Surfers. Both games require user to dodge various obstacles coming their way, but in our VR immersed game, user gets to shoot the object coming their way and thus increase their score for every obstacle they shoot/ and every second they dodge.

There was a Youtube video where someone simply ran the original Subway Surfers on the Oculus Rift. However, since the original game isn't a perfectly straight map, it looked like there was a small angle of rotation in constant velocity and even just watching the Youtube video itself, I felt vection.

We saw a tutorial on Youtube which was a spinoff of the original game without the VR implementation but they provided some of the basic code for the foundation of the game's constant motion of the map and obstacles. The concept of an HP bar instead of simply making the player trip and the pursuing aggressor from getting closer to them was also inspired by a tutorial about how to shrink the HP bar for every obstacle hit by scaling down a certain axis.

During the demo, we also saw someone create their own version of subway surfers. The major difference was that instead of placing a fixed set of obstacles and a fixed velocity, they create a randomization script which randomized the spawn of the obstacles and the game sped up the player's speed instantaneously instead of over time which avoided vection and made it easier to show that the difficulty was increasing. Lastly, they mapped the player movement with a joystick instead of tracking the player's position in the real world.

Regardless, both our take on Subway Surfers and the other group's take on Subway Surfers was unique in its own way where we probably deviated more from the original concept of the game to make it feel like a different type of game thanks to the gun our player has and where their game feels more like the actual original game thanks to the randomizer script.

Design (4 points, ~2 pages)

The game starts off with a six walled room that has a space-like background except the floor in order to avoid tripping out the player. The game welcomes the player with the text "Start" and a track that they will start on. There is no explanation for how the Start button works but the player will find that they already have a gun on their right hand. We believed that players would be able to figure this out on their own that in order to start the game, they must shoot the start button down. The Start button has a cube which detects if the gun is currently pointing at it and if the player shoots it down, the game immediately starts by moving the Start button away from the player's field of view and by beginning continuous movement of the track and the obstacles backwards towards the player.

The game itself uses a single railroad track as it invokes similarities with the original Subway Surfers. The graphics that define the train track uses a de facto mipmap since the resolution is clearer when it's closer to the player and lower when it's farther from the player. The game is also confined by four gray walls on the top, left, right, and bottom of the map so that the player isn't distracted by any background on the outside. There are also invisible walls on both sides of the ground so that the player doesn't fall off if they move too far from the middle. Once the ground moves back, the obstacles also move back at the same speed as the ground, giving the illusion that the player is moving forward. However, there are two things on the left side of the player, around their 10:30 position. There is a score that iterates over time that is displayed in white font. Below the white font, there is also a gray health bar in the shape of a cylinder which tracks how much health the player has and goes down whenever the player hits an obstacle. There is a collision detection and a tag used to detect that the player hit an obstacle. The health bar shrinks by scaling down the Y position and by changing the color of the HP bar to red so that it's easier for the player to see that they just lost health and once the scale is 0, the health bar disappears and so once that happens, the player gets teleported into another room which is a six walled room that has a game over prompt. The speed increases over time as the player dodges obstacles or shoots it down. When it comes to the generation of obstacles, we will simply be loading in easy patterns that are pre-generated in a way we believed was easy to dodge in the beginning but harder to dodge as the player progressed throughout the game. Since the

speed of the player is actually constant, loading in harder to dodge patterns seemed sufficient enough in giving the illusion that the game was going faster when it was actually not. To make it harder still, we accidentally made the obstacles appear only when they were really close to the player so they would have less time to react. In order to maintain the idea that this game is very welcoming to newcomers, we decided to simply give infinite ammo to the gun so that the player could memorize patterns to dodge later by just shooting down the obstacles and then once the player was confident enough to challenge themselves after restarting, they could try to dodge everything without shooting, which is significantly harder than shooting down some obstacles at least but is possible regardless.

Inside the second six walled room, the player encounters six blue walls which mimic the color of the blue screen of death for Windows 10. There exists a text on the front, left, right, and back side of the walls which all say ":(Your character ran into an obstacle and needs to restart. We're just collecting your score, and then you'll restart for you. (NaN complete) If you'd like to know more, you can practice to avoid this error: OBSTACLE_DODGE_FAILED. The reason why this was added was to add humor to the game over screen so that the player doesn't feel so bad for losing the game. On the roof, the player will find encounter two buttons that say "Restart" and "End It All". When the player shoots the "Restart" button, the player teleports back to the very beginning of the game where the track starts in which the player must shoot the "Start" button again to actually begin the game again. The "End It All" button was supposed to exit the game and the choice of words brings humor to edgy kids who always say they want to die. However, it didn't work and sometimes, it also functions as a duplicate restart button. At this point also, while the player is in the game over room, the ground should stop moving and the score should stop increasing.

Overall, we decided that all functionalities of the game will be using the Oculus Rift controls because we wanted it to feel as VR as possible. The position of the player in the real world maps the game and the trigger button is used for starting the game and shooting the obstacles. The only button used in the entire game is the trigger button on the right hand. Our design conforms to the Best Practices guide by avoiding vection and utilizing intuitive controls. The design works best in the scaling of the room and the game since the world looks really big

while the design is worst where there are no walls next to the track the player is on during the game. In our design, we added bright colors in the game over room since the player doesn't have to focus on the game. In the game itself, we use mostly gray colors so that the player isn't distracted by the background and so that the player can focus more on the game itself.


05


Start
Welcome t
Runner Game


Start

Implementation (2 points, ~1 page)

Our first objective/milestone was to make the moving scene, how to move the player through the game. Thankfully, we were able to find necessary assets in the Assets store to help us with the design of the room. We were also facing some issue with the ground but we were able to find texture.png, and then extend it to the track and then move the track backward to make the player feel like it's moving forward. We implement that by giving the ground the velocity for moving backward.

Our second milestone was How to make the object collisions and dodging work, so we implemented by assigning different tags to the cubes or obstacles as objects. And used the physics ray cast in order to shoot those objects, and then adding every obstacle shot or dodged to increase the score. If the gun's line of sight is on the object that is tagged with cube and the player is pressing the trigger button, we destroy that particular object. Although sometimes the tags did not work and collision sometimes became disabled, we solved those glitchy problems by positioning the player on the track immediately after opening the game.

Our third milestone was figuring out how to track the health of the player and what to do once the player's health bar was zero. We simply scaled down the size of the bar on its y-axis by 0.3 to shrink it and then assign the color to red, regardless of whether it is still gray or if it's already red. If the scale of the health bar reached 0, we set the player's position to a six-walled room and disable the movement of the map and the objects. If the player wants to restart, we set a flag that resets the game's position to true. At this point, we also re-create any obstacles that were destroyed.

The last milestone was simply testing whether or not the obstacles were realistically dodgeable and so objects were cloned and pasted in the most appropriate positions we felt were realistic enough to dodge.

All obstacles were placed as a child of the ground object so that the obstacles would follow the movement of the ground as well. The player, the health bar, and the score is not a child of the ground object so they don't move at all.

We downloaded and utilized a train track asset, train assets, and a wooden asset set of objects. The objects we have that are not assets were the health bar, score, the six-walled room

using planes, the buttons, and the 3D text. We also used the OVR player controller to track the player's position but we limited one degree of freedom so that the player cannot move forward or backwards.

Lessons Learned (2 points, ~1 page)

We learned that there were certain undefined behaviors that occured when using multiple scripts at once. In the beginning, we actually had a start room as well but since it messed up the game by removing the collision tags of the player and obstacles, causing the player to just go through the obstacles, we decided to remove the start room altogether and just have the start button in the game. This removed the problem but we still don't know why it fixed the problem.

Another thing we learned which was definitely very important was version compatibility of Unity. Since we only used lab time to plan our game on paper and used our own desktop and oculus rift outside the classroom, we didn't realize that the different versions of Unity actually significantly affected our game. The desktop we used had version 2018 of Unity installed and every prefab, script, and asset was working fine there. But when we copied and pasted the files used onto the classroom desktop for demo, we found out that it was using version 2017 of Unity and this disabled around half the prefabs we used and generated a lot of compiler errors. We fixed this by creating an executable and copying and pasting the scripts onto the classroom desktop. We were able to run the game and we no longer had to worry about version compatibility. We realized that it was better to do our work on the classroom desktop first and make sure that transferring back and forth from personal computer to classroom desktop shouldn't cause any problems.

Users who tested our project mostly reacted by asking why the player was able to shoot down a train using a simple glock and why the gun had unlimited ammo but it was asked in a humorous way and it was mostly friendly camaraderie. Everyone had an easy time learning how the game worked and as expected, those who had a hard time dodging chose to shoot down all the obstacles while those who wanted a challenge dodged everything and the game was challenging enough for them to have fun. I believe the only thing people disliked were the little glitches that occurred occasionally where sometimes the player fell off the map for some unknown reason or the restart button couldn't be shot down. We would probably improve our game by adding more graphics and randomizing the obstacles so that the game feels more realistic and immersive. We would also add an explosion animation so that the train doesn't just abruptly disappear after being shot down.

The workload was mostly distributed evenly. Joseph and Tanish focused on the design of the game from the assets downloaded to the objects and components added on the map. Mike and Ran were focused on coding the movement of the map, the HP bar, the buttons, and pretty much any scripts needed to make the game work.