Rubber Band Travel: Implementation

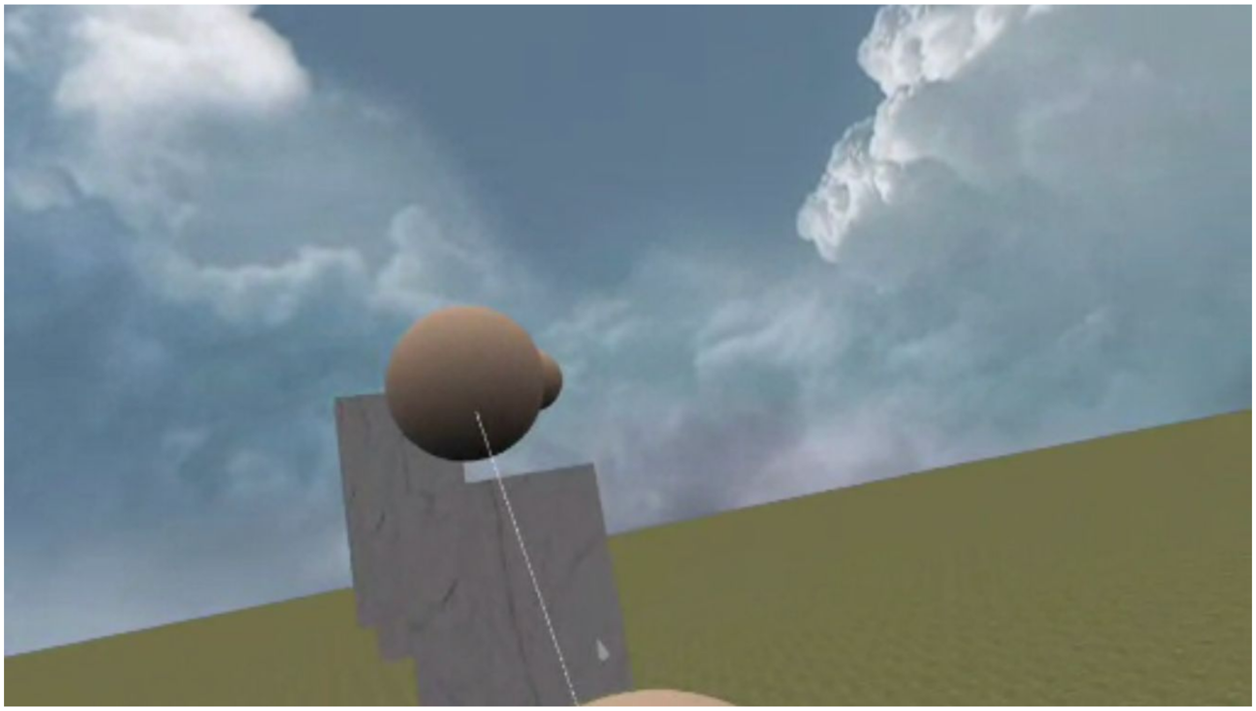
By Chris Obando and Teddy Chen

Development Environment

Rubber band travel was developed for web platforms using Typescript, with Visual Studio Code as our editor of choice. Node package manager managed third party libraries, with webpack as the bundler and development server. We used Git for version control and ngrok for port forwarding, in order to test our implementation on the Oculus Quest.

Implementation Details

The application we developed includes a full implementation of rubber band travel. When the user presses both squeeze buttons, a dashed line appears, connecting both hands. While pressed, the user can stretch and contract this “rubber band.” Upon the release of one of the buttons, a force is applied to the user, causing the user to fly to his or her destination. This force is proportional to the amount of stretching of the band and in direction of the band’s release.

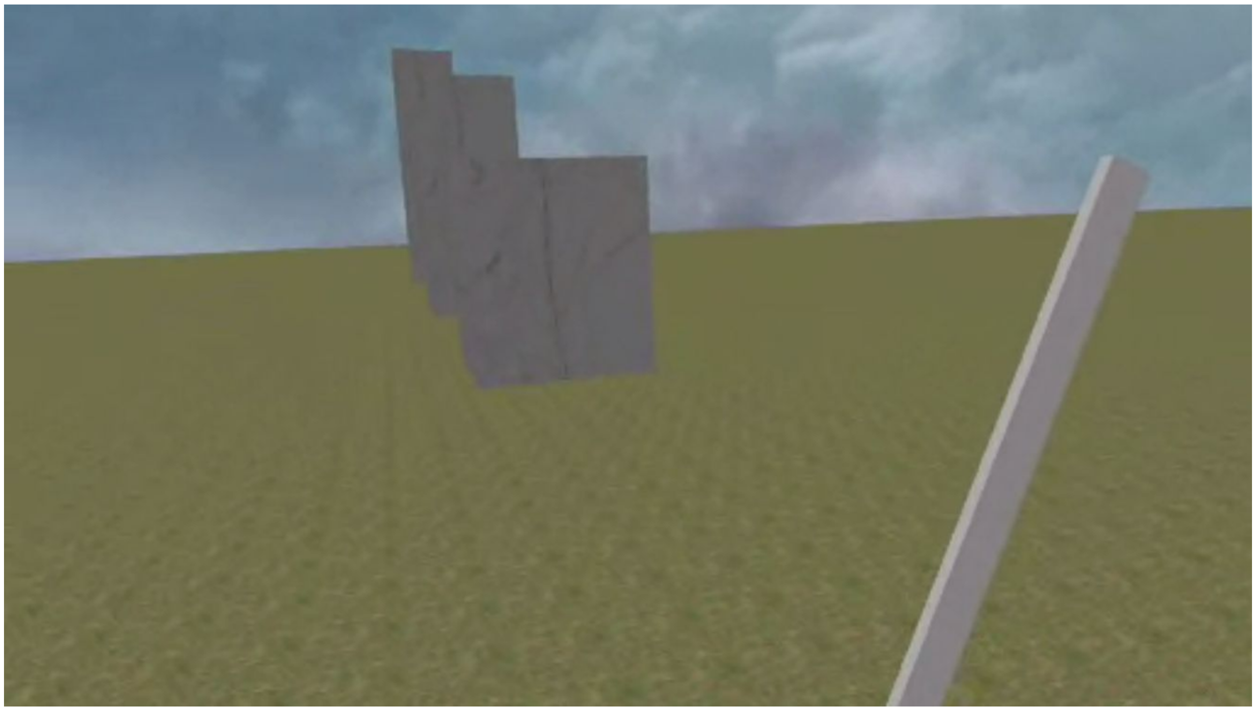


While in the air, the user can repeat the stretching action to alter his or her flight trajectory and duration slightly. However, the magnitude of the force mid flight will be dampened. Gravity in the scene will eventually pull the user back to the ground.

There were a few deviations from the original proposal of rubber band travel. One of them was dispensing with the requirement that the user must bring his or her hands close in order to initiate a rubber band. We decided this motion would get tedious and repetitive, especially in time-constrained environments. Also, we decided not to utilize a third person perspective during travel, because the user may have to slice enemies or do fine movements to manage inventory, and having an out-of-body perspective would make it harder to aim the hands. We did add an extra feature: a GUI to adjust gravity and the rubber band's force in the scene. This is so the user can try out different parameter settings to see what settings best fit their application.

To highlight the use cases of rubber band travel, we created an obstacle course in Blender, and imported it into our scene. We were able to import it onto the map (see commented code on index file), however, the physics imposter for meshes did not seem to work well, so we just used box imposters.

The map has various platforms and targets to rubber band onto and off of. But that’s not all — we also gave the user a sword and sprinkled in a few training dummies, to showcase how the user can work with his or her hands during navigation.



Overall, we created a neat demo environment to allow people to explore the possibilities available with rubber band travel. In the future, this method of locomotion would pair well with fast paced futuristic shooters like Halo or Apex Legends where your character is constantly moving and where there are maps with several floors that your character jumps between to juggle several enemies at once.

Third Party Libraries Used

We used BabylonJS (https://www.babylonjs.com/) for 3D XR rendering, and the AmmonJS physics engine (https://github.com/kripken/ammo.js/) for forces and gravity collisions.

Assets

We used grass, skin, and sky textures from Google Images, which are protected by Fair Use. When applying the skin textures to the hands, Chris found that both the ground and skin textures being applied with diffuse texture were very shiny, so he found this discussion: https://www.html5gamedevs.com/topic/18589-not-everything-is-shiny-eg-human-skin/, which he referenced for setting specular color to 0.