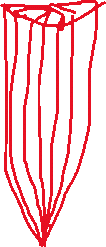
 I started by creating a bullet using a lathe function I made in ITC320. The red line in the image is the line that’s defined, and the blue mesh is generated from it by rotating it around the center.



I already had a first-person camera from Assignment 1. I used the forward vector generated by this camera’s rotation to work out the direction to shoot a projectile in. The bullet’s rotation updates to match the bullet’s direction. I made the bullet quite slow because it’s more interesting to watch:

GIF1

I had a lot more trouble with the achievement task, mainly because it’s a confusing problem. I achieved it by transforming the ray representing where the bullet is about to move into local space using the inverse world matrix of mesh instance, then, , testing the ray against each plane of the bounding box in local space, then transforming the result (if there is one) back to world space.

There was some tricky math involved including a matrix \* vec4 transform function, a vector to euler angles function and the ray cast itself. While debugging this I wrote some tests in math.tests.js (uncomment the part at the bottom to run them). I attempted the method described in (Szauer, 2017) but it didn’t quite work reliably, so I assume my implementation is incorrect (I left it in rayCastAABB() in math.js).

I added a boolean for when meshes are destructible for the cubes (so bullets will hit buildings, etc but not destroy them). I also made a physics class to handle raycasts and any other physics that come up throughout the course.

# References

Szauer, G. (2017). Raycast Axis Aligned Bounding Box. In G. Szauer, *Game Physics Cookbook.* Packt.