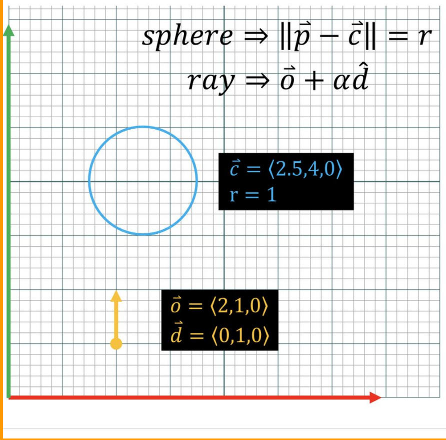
Ray intersections

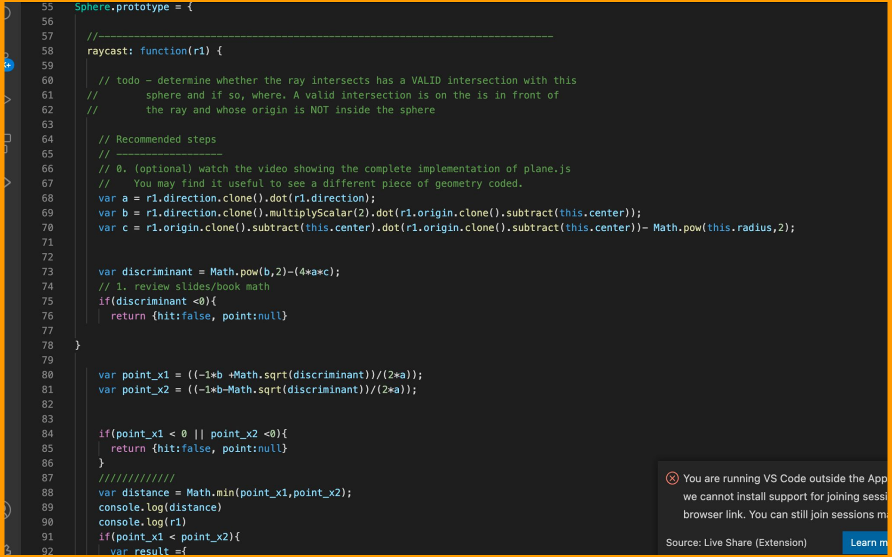
This week, I learned about Ray intersections. I definitely found it challenging to learn about them other than of course applying the quadratic formula part.

Ray intersections are when a ray and sphere intersect. This can be done by finding the point between first point and second point and dividing it by the absolute value of two points. Ray intersections are often used for bullets and laser beams which I never knew until now. I always knew that rays were in a straight line however, what I didn't know was that sometimes we use them to find a particular point on a surface that we desire such as a target on a bullseye. One perfect example of what the direction from my apartment to clases are. I could simply point my finger to the direction of my classes however, I may take a few turns, walk around the building and it may not always be a straight line like one point to a ray and another isl

When I started implementing the code into my VS system, I got the majority of the logic correct however, forgetting to put the parenthesis really changed the test results of the code. I was expecting my code to neutralize but it didn't because of something simple as that. Luckily, I found the issue with my TA and instructor and all 9 test cases succeeded. It goes to show that someiimes the most simple mistake could be the one thing that makes the program execute successfully!



$sphere \Rightarrow \|\vec{p} - \vec{c}\| = r$
$ray \Rightarrow \vec{o} + \alpha\hat{d}$
$\vec{c} = \langle 2.5,4,0 \rangle$
$r = 1$
$\vec{o} = \langle 2,1,0 \rangle$
$\vec{d} = \langle 0,1,0 \rangle$


```
Sphere.prototype = {

  //------------------------------------------------------------------------------
  raycast: function(r1) {

    // todo - determine whether the ray intersects has a VALID intersection with this
  //        sphere and if so, where. A valid intersection is on the is in front of
  //        the ray and whose origin is NOT inside the sphere

    // Recommended steps
    // ------------------
    // 0. (optional) watch the video showing the complete implementation of plane.js
    //    You may find it useful to see a different piece of geometry coded.
    var a = r1.direction.clone().dot(r1.direction);
    var b = r1.direction.clone().multiplyScalar(2).dot(r1.origin.clone().subtract(this.center));
    var c = r1.origin.clone().subtract(this.center).dot(r1.origin.clone().subtract(this.center))- Math.pow(this.radius,2);


    var discriminant = Math.pow(b,2)-(4*a*c);
    // 1. review slides/book math
    if(discriminant <0){
      return {hit:false, point:null}

}

    var point_x1 = ((-1*b +Math.sqrt(discriminant))/(2*a));
    var point_x2 = ((-1*b-Math.sqrt(discriminant))/(2*a));


    if(point_x1 < 0 || point_x2 <0){
      return {hit:false, point:null}
    }
    /////////////
    var distance = Math.min(point_x1,point_x2);
    console.log(distance)
    console.log(r1)
    if(point_x1 < point_x2){
      var result ={
```

You are running VS Code outside the App
we cannot install support for joining sessi
browser link. You can still join sessions m

Source: Live Share (Extension)