CIS367 - GFX

Winter 2023

**HW 3: Other Libraries**

**Due March 31st**

**Note: all screenshots go in the report at the end!**

**Synopsis:**

Let's get setup with ThreeJS. You're going to setup a basic scene, play with lighting and primitives, and load in a model.

Then your homework will be to extrapolate. Oh boy!

Note: most of this comes from the ThreeJS documentation and some from <https://www.briankoponen.com/three-js-introductory-tutorial/> (though this tutorial is a bit out of date at this point).

**Step -1**: Log into your EOS account and go to your /WEB\_STUDENT/<your-last-name> folder.

**This is what I'll be doing up front.**

You'll create your **hw3** folder inside of public\_html and access it via your usual CIS webpage URL.

**Step 1:** Get familiar with the ThreeJS documentation. Start here: <https://threejs.org/docs/index.html#manual/en/introduction/Installation>

**Step 2:** Make a webpage.

Assume everything we do now is within your lab folder. At its root create an index.html file. Inside that file add the following code (i.e., Install from CDN or static hosting):

<!DOCTYPE html>

<html>

<head>

<meta charset=utf-8>

<title>YOURLASTNAME - 367 Three.js demo</title>

<style>

\* {

margin: 0px;

padding: 0px;

}  
 </style>

<script async src="https://unpkg.com/es-module-shims@1.6.3/dist/es-module-shims.js"></script>

<script type="importmap">

{

"imports": {

"three": "https://unpkg.com/three@**latest**/build/three.module.js"

}

}

</script>

<script type="module">

import \* as THREE from 'three';

const scene = new THREE.Scene();

</script>

</head>

<body>

</body>

</html>

**Replace YOURLASTNAME and put a screenshot of your barebones webpage in Q1.**

**Step 3:** Setup a default scene.

Lets move over to <https://threejs.org/docs/index.html#manual/en/introduction/Creating-a-scene> now. You already have the setup in place, now lets add some content.

Add a camera and renderer (this goes under the const scene = … line):

const fov = 75;

const near = 0.1;

const far = 1000;

const camera = new THREE.PerspectiveCamera( fov, window.innerWidth / window.innerHeight, near, far );

const renderer = new THREE.WebGLRenderer();

renderer.setSize( window.innerWidth, window.innerHeight );

document.body.appendChild( renderer.domElement );

renderer.setClearColor(0xFFFFFF);

Then right on top of the </script> call put in our call to render and animate the scene, otherwise we'll just have a blank canvas the whole time:

function animate() {

requestAnimationFrame( animate );

renderer.render( scene, camera );

}

animate();

**Change the background color and put a screenshot into Q1.**

**Step 4:** Add a **mesh**.

Now, let's add a floor and a mesh. If you recall from class, a mesh is basically our object that we're modeling (comprising a geometry and a material).

Add this after the call to glRenderer = …

// Add a floor to the scene

let floorWidth = 20;

let floorHeight = 40;

let floorGeometry = new THREE.PlaneGeometry(floorWidth, floorHeight);

let floorMaterial = new THREE.MeshStandardMaterial( { color: 0x009900 } );

let floorMesh = new THREE.Mesh(floorGeometry, floorMaterial);

// A Plane is created standing vertically.

// Let's rotate it so that is lays flat.

floorMesh.position.set( 0, -1, -3 );

floorMesh.rotation.set( -Math.PI/2, 0, 0 );

scene.add(floorMesh);

// Add a box to the scene

let boxWidth = 1;

let boxHeight = 2;

let boxDepth = 1;

let boxGeometry = new THREE.BoxGeometry(boxWidth, boxHeight, boxDepth);

let boxMaterial = new THREE.MeshStandardMaterial( { color: 0x00ffff });

let boxMesh = new THREE.Mesh(boxGeometry, boxMaterial);

boxMesh.position.set(-2, 1, -5);

scene.add(boxMesh);

**Take a screenshot and add to Q1.**

Now, change a couple of parameters. Update fov to be 100 and farPlane to be 10.

**Take another screenshot and add to Q1. In Q2, describe technically what is actually happening (note: it looks different or better is not a valid answer … what does updating these two parameters actually *do).***

**Step 5:** Let's add a touch of lighting.

After the call to scene.add(boxMesh) add the following to setup a light:

// Add the Lights

// Ambient light is used to prevent the scene

// from ever being too dark.

var ambient = new THREE.AmbientLight(0x333333);

scene.add(ambient);

// A point light acts like a light bulb, sending light

// in all directions.

var lightIntensity = 1;

var pointLight = new THREE.PointLight(0xffffff, lightIntensity);

pointLight.position.set(2, 4, -2);

scene.add(pointLight);

// Enable Shadows

// The floor will only receive shadows, but the box can both

// cast and receive shadows.

renderer.shadowMap.enabled = true;

floorMesh.receiveShadow = true;

boxMesh.receiveShadow = true;

boxMesh.castShadow = true;

pointLight.castShadow = true;

**Take a screenshot and paste into Q1.**

Now, tweak some of the parameters. Change the light color/position. Turn shadows on and off. Make it look *different* purely by lighting it. **Take two screenshots and describe what you did in Q3.**

**Step 6:** ANIMATION STATION.

Let's animate! We're going to drop a few things in here now so pay attention.

Add a global variable to the top of your script block to contain a bunch of rotating boxes. Put it right on top of const fov = …

let boxes = [];

Next, change boxWidth to 0.9 and boxDepth to 0.2.

Then under the definition of boxMaterial, change the definition and position setting of boxMesh to the following:

for (let i = 0; i < 14; i++) {

let boxMesh = new THREE.Mesh(boxGeometry, boxMaterial);

boxMesh.position.set(i - 7, 1, -5);

boxMesh.receiveShadow = true;

boxMesh.castShadow = true;

scene.add(boxMesh);

boxes.push(boxMesh);

}

Almost there! If you look now you should see some unmoving boxes. Add in the following code to the top of your animate function:

function animate () {

**for(let i = 0; i < boxes.length; i++) {**

**boxes[i].rotateX(Math.PI/(100+i));**

**}**

renderer.render( scene, camera );

requestAnimationFrame( update );

}

Should look fairly familiar to what you recall! Now, just add a call to update() after init() and watch it animate!

**Take two screenshots to show me that it's animating and paste into Q1.**

**Step 7:** Texturing!

Download the following images and place them in your main lab folder:

(Originally from opengameart.org 🡪 I highly recommend checking out this site if you've never been there)

<https://www.briankoponen.com/assets/textures/wests_textures/grass1.png>

<https://www.briankoponen.com/assets/textures/wests_textures/stone%20wall%204.png>

Note: easy way to get files in linux –

wget <file>

First, we'll load in the textures and move a few things around.

Add in the following code after the call to renderer = …

// Load Textures

let floorWidth = 20;

let floorHeight = 40;

let boxWidth = 1;

let boxHeight = 2;

let boxDepth = 1;

let textureLoader = new THREE.TextureLoader();

let grass = textureLoader.load('grass1.png');

grass.wrapS = THREE.RepeatWrapping;

grass.wrapT = THREE.RepeatWrapping;

grass.repeat.set(floorWidth / 2, floorHeight / 2);

let stone = textureLoader.load('stone wall 4.png');

stone.wrapS = THREE.RepeatWrapping;

stone.wrapT = THREE.RepeatWrapping;

stone.repeat.set(boxWidth / 2, boxHeight / 2);

Then you can either comment out or delete the call to defining floorWidth/floorHeight/boxWidth/boxHeight/boxDepth later on as it was needed for the texture sizing.

Then, replace the call for creating the floorMaterial with your new texture:

var floorMaterial = new THREE.MeshStandardMaterial({

map: grass,

metalness: 0.25,

roughness: 0.75

});

And do a similar process for the boxMaterial definition:

var boxMaterial = new THREE.MeshStandardMaterial({

map: stone,

metalness: 0,

roughness: 1

});

**Take a screenshot of your textured scene and paste into Q1.**

Now, go to opengameart or itch.io and look for textures that you can use. Replace the grass and brick textures I provided with something else. Make sure it's safe for work.

**Take a screenshot of your newly textured scene and paste into Q1.**

**If you feel you want extra credit add orbit controls (note – you'll need to set the camera position if it isn't "working"). Tell me in the lab report that you did this. If you want double extra credit either create a 3D model in Blender or find one online and import it into your scene – easiest way will be to use the glb format.**

**Homework:**

**1) Add a spinning torus to the scene**. Check <https://threejs.org/docs/?q=torus#api/en/geometries/TorusGeometry> for the syntax.

You can put it anywhere, I just want to see it placed in addition to what you did in the lab.

**Take a screenshot and put it in Q1. Note, I'll check when I visit your site as well to see that it's animating.**

**2) Pick one of the following 2D libraries and implement a "thing"**

* Phaser: <https://phaser.io/>
* PixiJS: <https://www.pixijs.com/>
* **The thing:** Implement a controllable **Sprite** character. This character should be controlled by the arrow keys and bounded by the screen (the character can't leave the screen). Bonus points available if you gamify it in some way.

**3) Term Project**

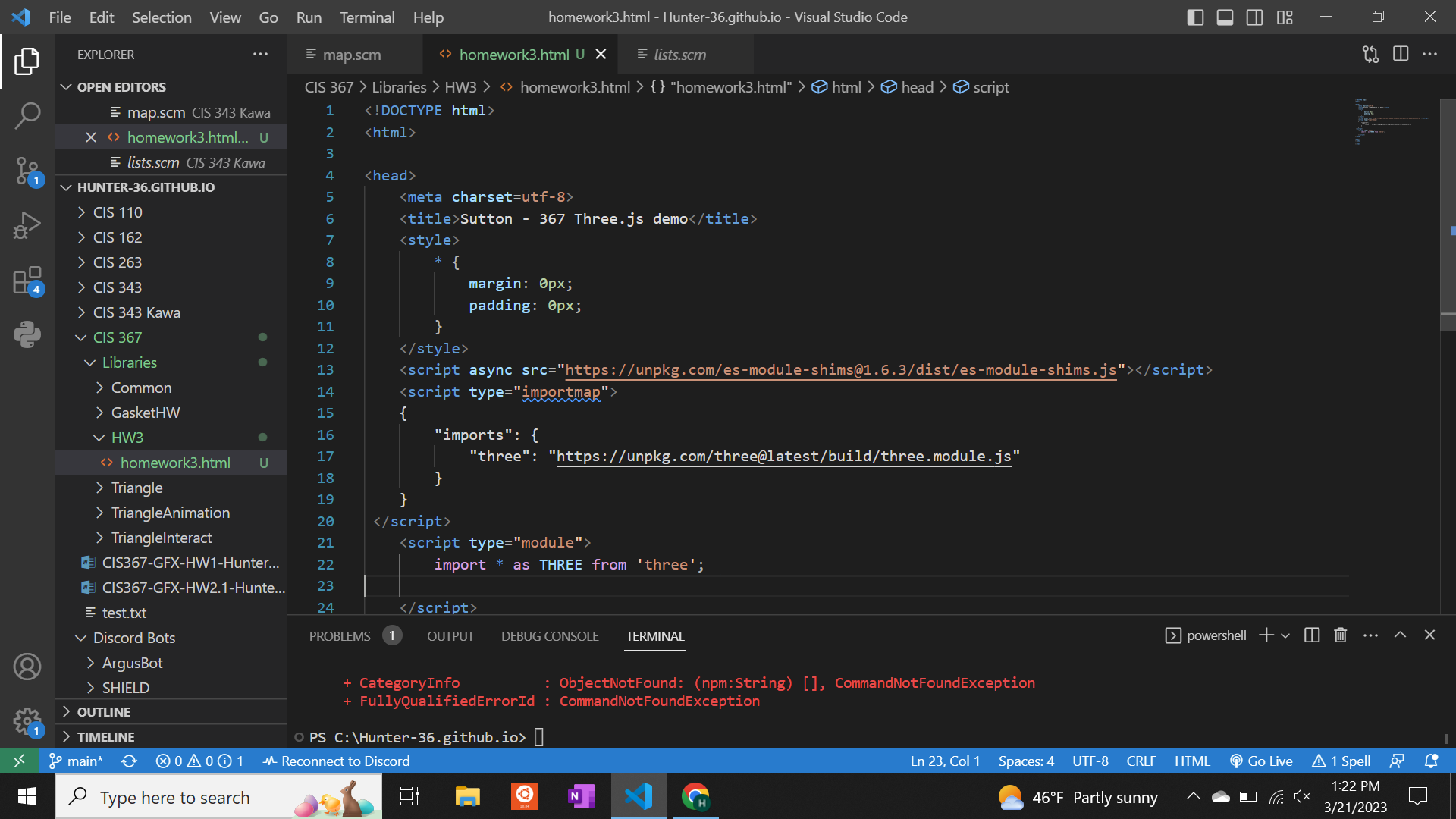
As promised, you'll have some forced progress on your projects. In the last week of class I'll have you all give presentations/demos on what you've done. However, I'll be looking for something that is approximately 2-3 homework assignments worth of effort (think of it like a prototype of a future project … the more effort you put into it the more portfolio-worthy it will be).

You have all proposed your topics already and hopefully have started thinking about them. What I would like you to do is to implement **two things** towards your goal. It could instantiating the scene, importing a music library, displaying a demo model, etc. But, provide two screenshots below of what you did.

**Lab Report / Homework:**

1. Provide all required screenshots from the lab (except the lighting ones – those go in Q3).

*Hint: there should be 9.*

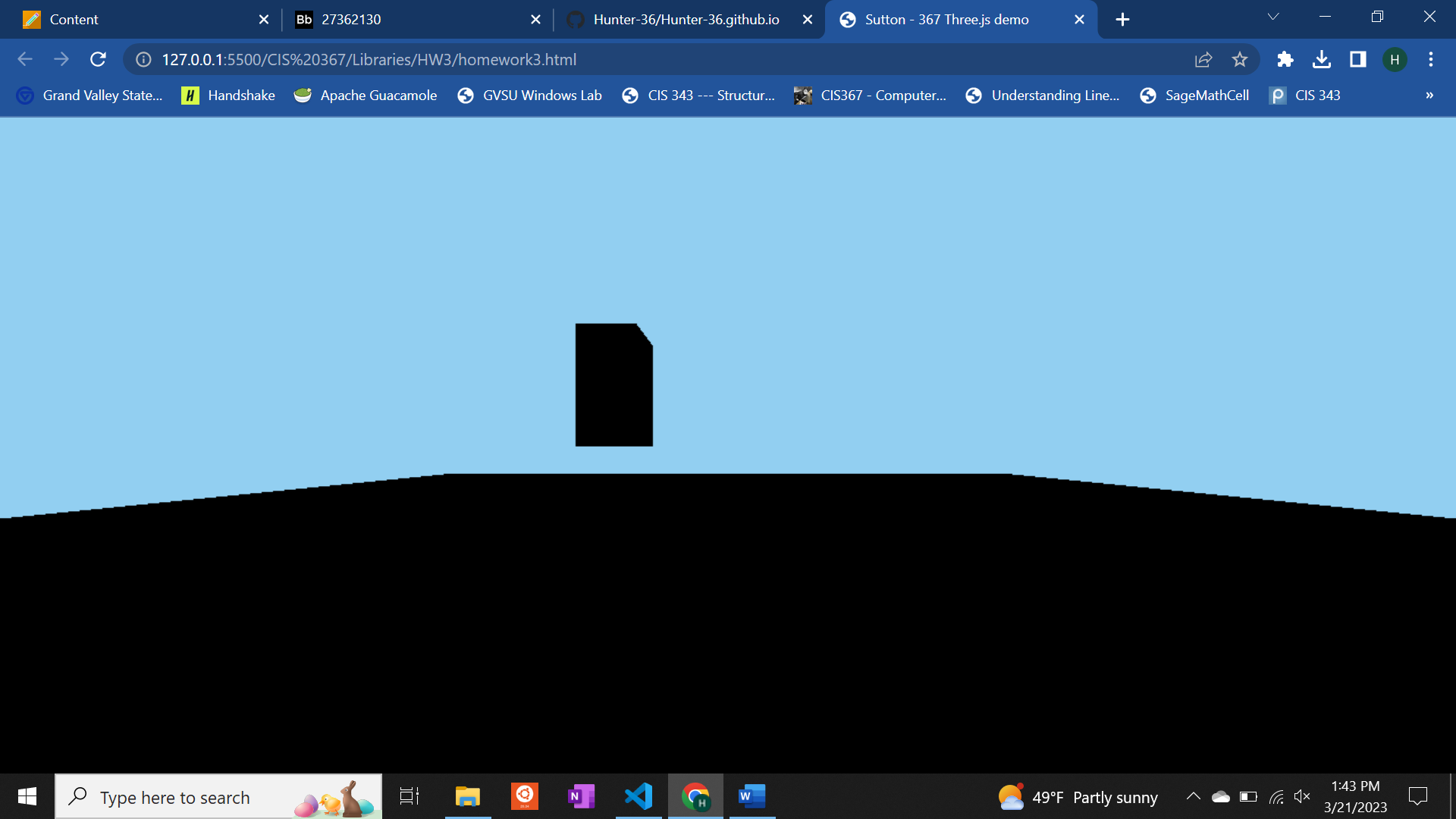


A screenshot of a computer

Description automatically generated with medium confidence

A screenshot of a computer

Description automatically generated



A screenshot of a computer

Description automatically generated with medium confidence

Chart

Description automatically generated

A screenshot of a video game

Description automatically generated

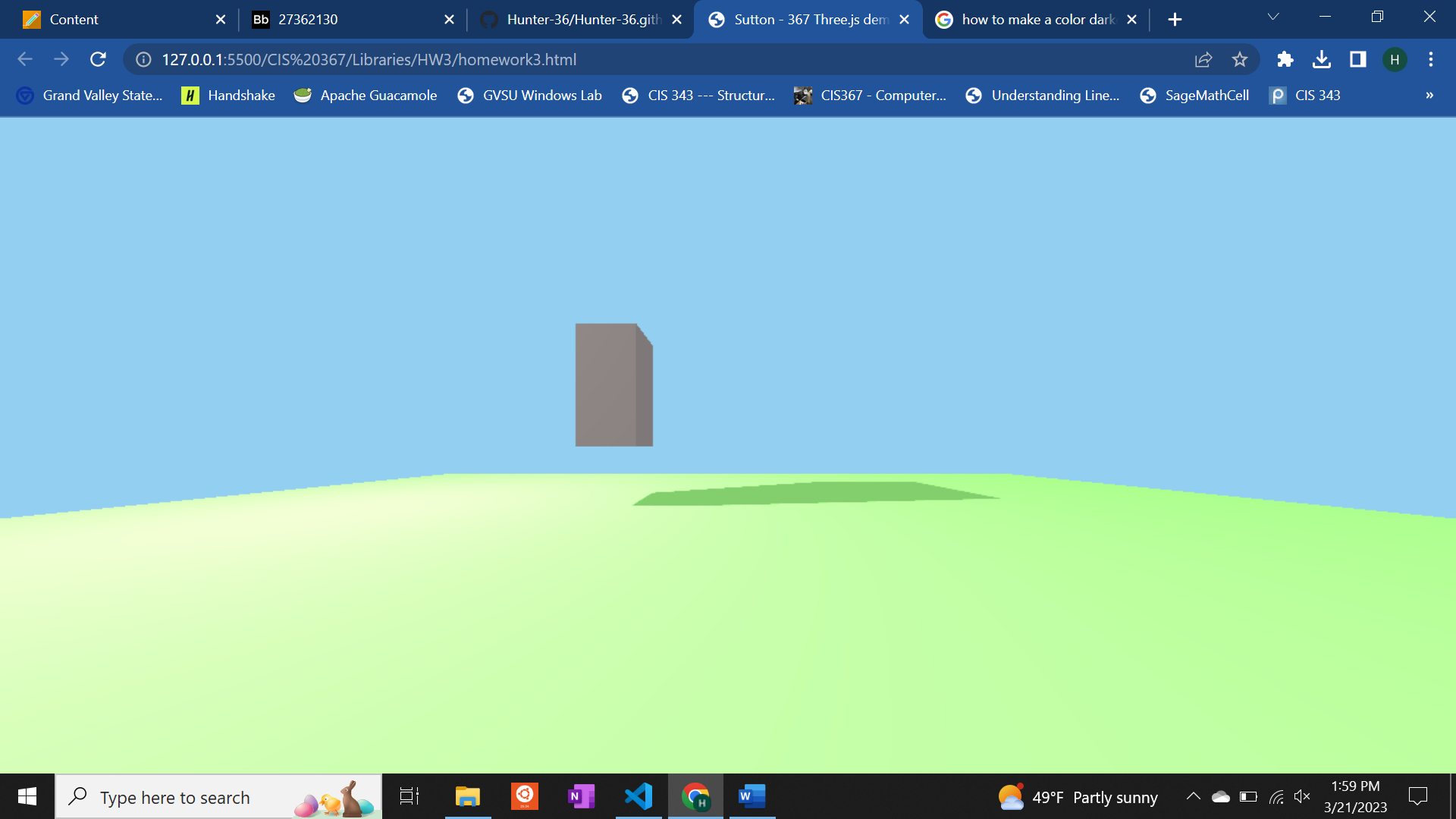
1. What does updating fov and farPlane do?

Well, fov is field of view and it widened the view of the plane and cube, basically making it be able to see more. And I believe far being changed to 10 backs the camera up from the object.

1. Provide lighting screenshots and a description of what you changed.

A screenshot of a computer

Description automatically generated with medium confidence



So, what I did was I changed the color of both the ambient light and the point light, making the ambient light softer and the point light more of a sun color. Then I changed the point light position to cast the shadows more back at the camera.

1. In the animate function, change the 100 within rotateX to be 10. What happens and **why?**

1. Provide a link to your demo site for 2D graphics.
2. Describe exactly what you have done (for Q5) **and** what the controls are.
3. Provide two screenshots of your term-project-in-progress
4. Describe your next steps for your term project
5. If you did the extra credit(s), describe what you did here.