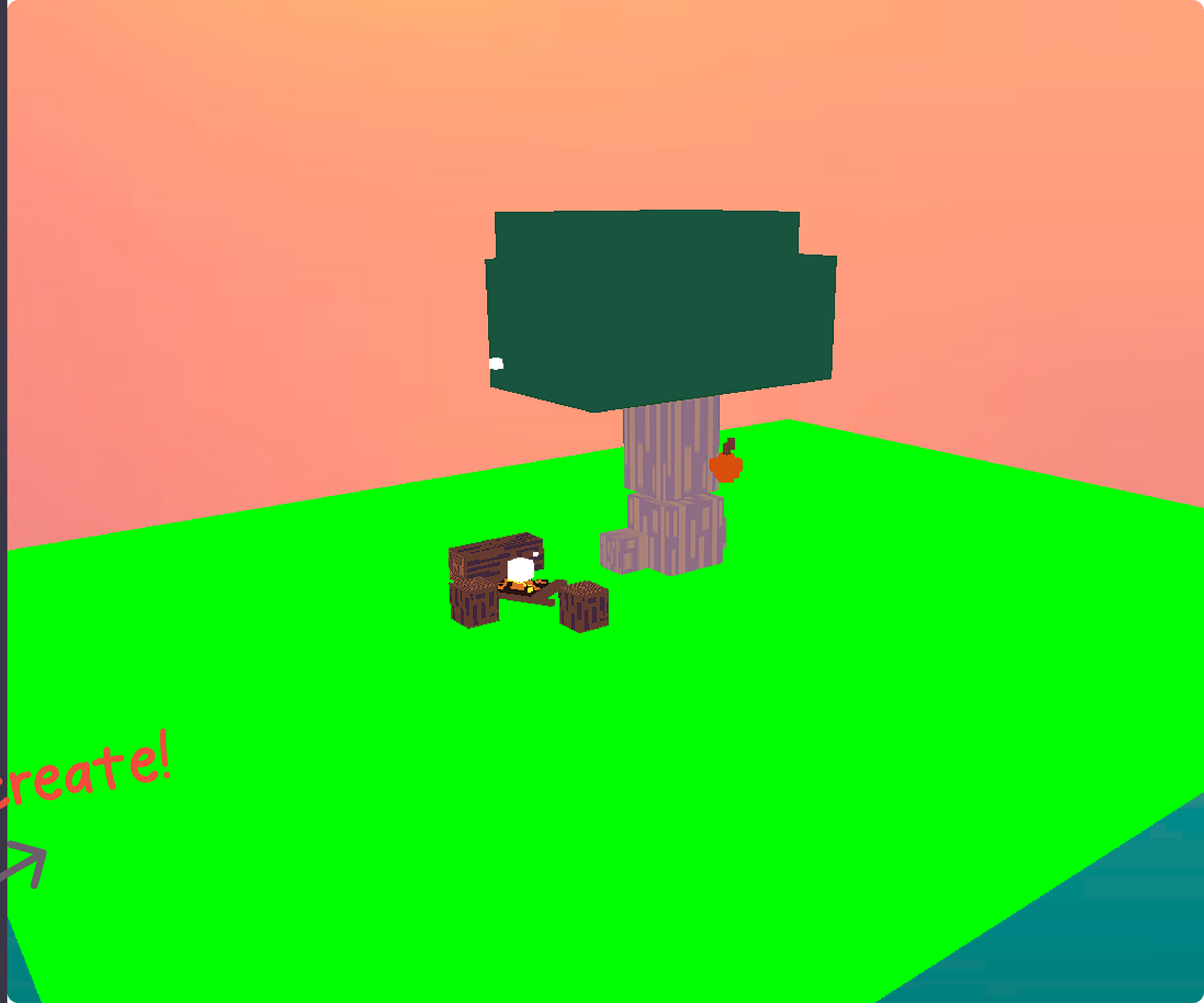


Create your own,  
LOFI  
animation, powered  
by Voxels.



What you will create!

*Ideal and fun for all hackers!*

# *Hello fellow hacker!*

This is the start of a wonderful adventure. You will create your very own **LOFI** animation!

This will be done with the help of:

+ Three.js  
+ Voxels!

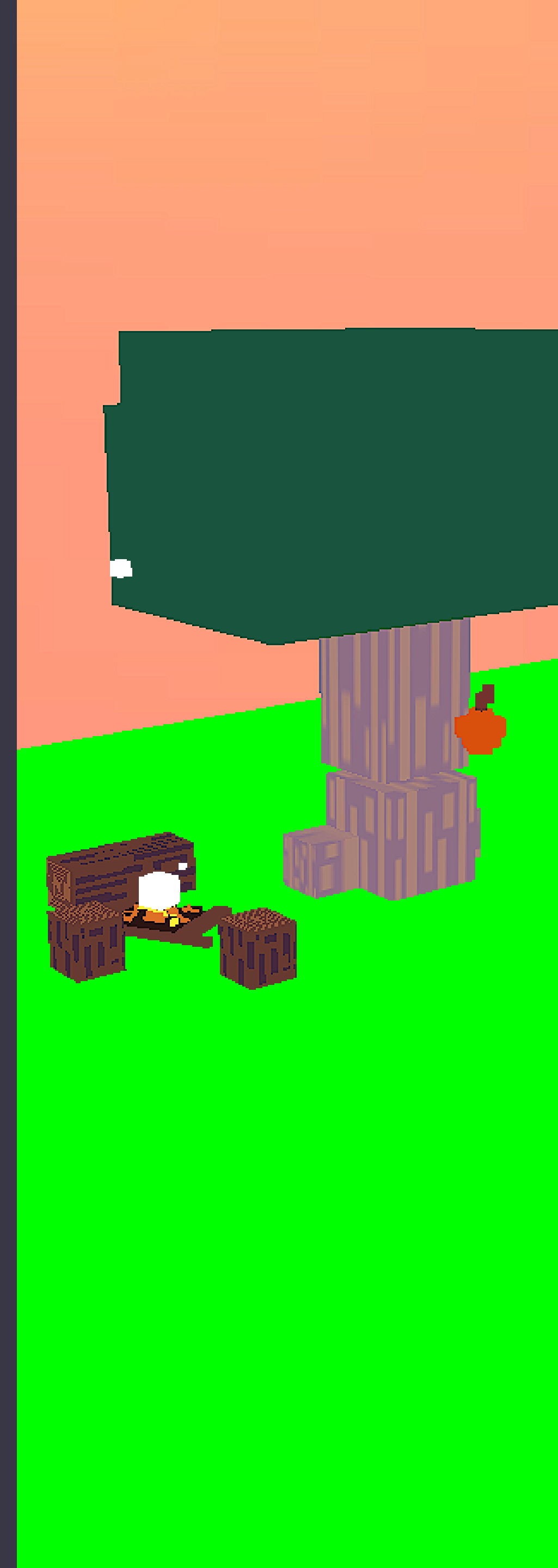


# The Outline

But before we get started let's see what we will accomplish!

Brace yourself, the list might seem a bit length but you will complete it in not time! You got this!

- Forking the template, replit fundamentals
- Quick explanation of the template
- THREE.JS Basics
- Adding a camera, sample cubes and cube animation!
- Adding textures
- Three.js must haves
- Recording functionality





## OBJECTIVE #1

Forking the template, replit  
fundamentals

# Forking

The first part of your journey is this!

We first must need to fork the template, because it has many useful assets and all of the libraries pre-installed.

After opening the link provided, click the, **FORK**, button.



1



# Forking

On the prompt enter your animation name and click, Fork Repl

## Fork Repl

Name 19 / 60

YOUR ANIMATION NAME

Description 0 / 1000

What does this Repl do?

Privacy

Public  Private

Anyone can view and fork this Repl.

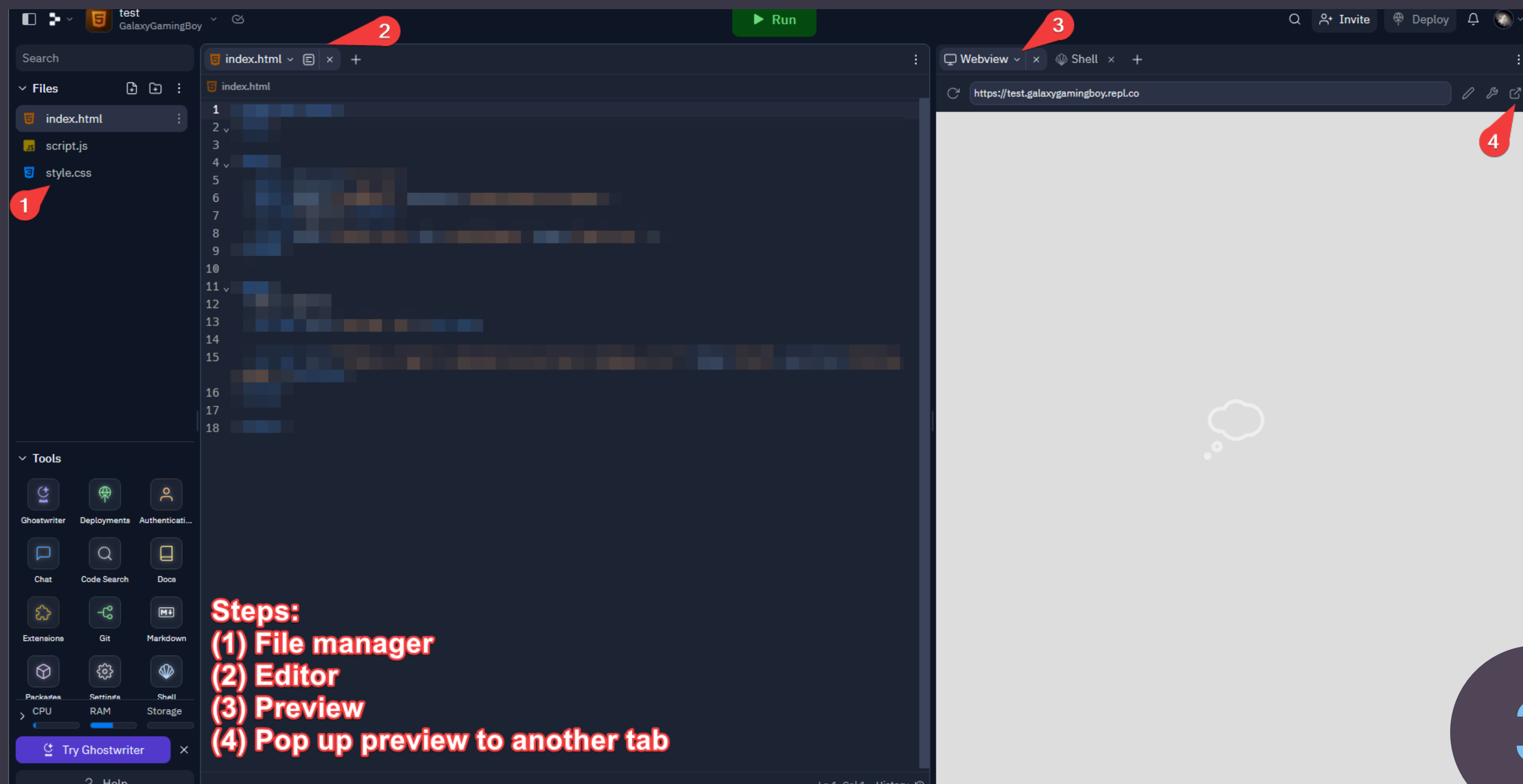
2



# Explaining Replit

After that your project that the animation will live in should be loaded into the same tab. It should look like the image below.

1. Indicates the file manager, there you can see all of your files.
2. Files opened with an editor below them.
3. A preview window that will preview your animation, it should automatically reload if it doesn't press RUN
4. Open the preview on another tab





## OBJECTIVE #2

Quick explanation of the template



# Template Explanation

Phew I hope forking the project wasn't that hard because we got more to do! :D

Let's start with a quick explanation of what the template provides us with:

- + The necessary libraries already downloaded
- + Bare necessary HTML code
- + Bare necessary JS code
- + Simple Assets

Feel free to skim this session if you have (or even don't have) time constraints!



# HTML Code

```
1 <!DOCTYPE html>
2 <html>
3
4 <head> Defines the head, including title
5   <meta charset="utf-8">
6   <meta name="viewport" content="width=device-width">
7   <title>Animation</title>
8 </head>
9
10 <body>
11 <div> The buttons that we will use
12   <button id="run">Run!</button>
13   <button id="rec-start">Start Recording</button>
14   <button id="rec-stop">Stop Recording</button>
15   <span id="rec-status"></span>
16 </div>
17
18 <script src="https://replit.com/public/js/replit-badge-v2.js" theme="dark" position="bottom-right"></script>
19
20 <!-- Import THREE.JS -->
21 <script async src="https://unpkg.com/es-module-shims@1.6.3/dist/es-module-shims.js"></script>
22
23 <script type="importmap"> Importing the libraries
24   {
25     "imports": {
26       "three": "https://unpkg.com/three@0.153.0/build/three.module.js",
27       "three/addons/": "https://unpkg.com/three@0.153.0/examples/jsm/"
28     }
29   }
30 </script>
31
32 <!-- Import CCapture -->
33 <script src="lib/CCapture.all.min.js"></script>
34
35 <!-- Import Script -->
36 <script src="script.js" type="module"> The script our animation will live in
37 </body>
38
39 </html>
```



# JS Code

2

```
1 const runButton = document.getElementById('run') Getting the elements from the HTML
2 const startRec = document.getElementById('rec-start')
3 const stopRec = document.getElementById('rec-stop')
4 const statusRec = document.getElementById('rec-status')
5 stopRec.disabled = true
6
7 // () => {} This is a arrow function
8
9 v runButton.addEventListener('click', () => { On run
10   runButton.disabled = true Disable the run button
11   // Executed on run
12 })
13
14 v startRec.addEventListener('click', () => { On start recording
15   startRec.disabled = true Disable the start recording button
16   stopRec.disabled = false Enable the stop recording button
17   // Executed on start rec
18 })
19
20 v stopRec.addEventListener('click', () => { On stop recording
21   startRec.disabled = false Enable the start recording button
22   stopRec.disabled = true Disable the stop recording button
23   // Executed on stop rec
24 })
```





OBJECTIVE #3

THREE.JS Basics

# Importing

This section will cover the basics of three.js so you can start with the rendering.

Yes, its time! This is where we import THREE.JS for rendering. :D If you had previous experience with Javascript you may already know how to import things but for those who don't, here is an explanatory Image.



1



# Importing

For those who **have** figured it out by the image congrats to you!

Imports are just requests to another javascript file saying that we want to load something from them.

Now let's add them! Your javascript lives on the script.js file. So all we got to do is add 1 import statements, well actually 3 because we also got to import the 3D model loader and the orbit controller.

```
import * as THREE from "three"; // Imports everything from THREE.JS
import { OrbitControls } from "three/addons/controls/OrbitControls.js"; // Imports
the orbit controller
import { OBJLoader } from "three/addons/loaders/OBJLoader.js"; // Imports the 3D
model loader
```

JavaScript

2



# Adding a renderer

Well it is time! We will add one of the most crucial piece of 3D rendering. **The RENDERER!**

It will handle, well, the rendering :D

Three.js makes this process a lot less tedious that it would normally be so let's jump into it.

In the first line we initialize the THREE WebGL renderer, it will handle the rendering job. In the second line we set the rendering size and we pass the window width and height. In the third line we append the renderer, so we can actually see.

Copy paste these lines:

```
const renderer = new THREE.WebGLRenderer();
renderer.setSize(window.innerWidth, window.innerHeight);
document.body.appendChild(renderer.domElement);
```

JavaScript

3



# The coordinate system

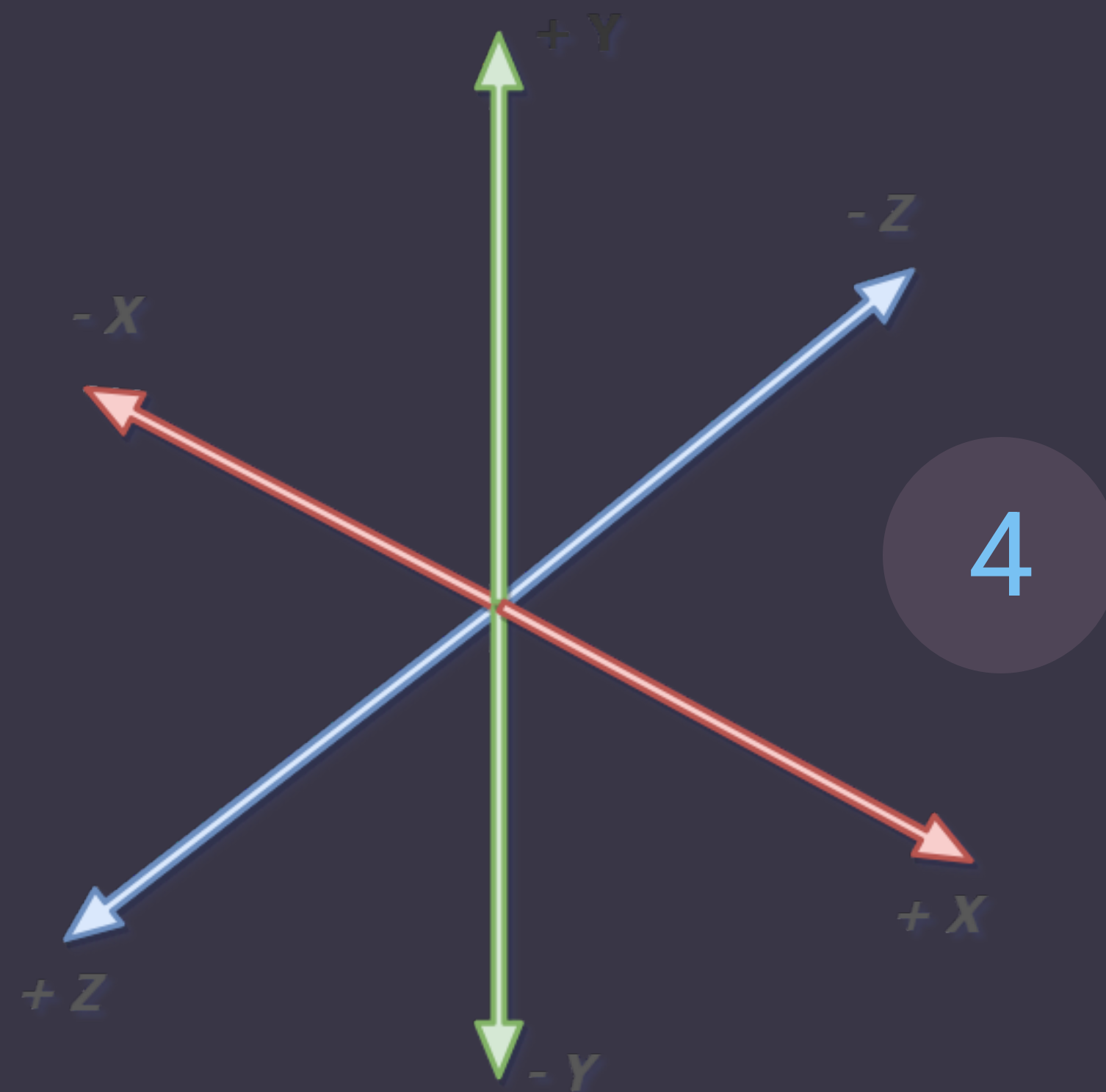
Well let's make a quick reference to the three.js coordinate system.

Like mostly all 3D applications, three.js utilizes the *Cartesian* coordinate system.

Here is a quick *illustration* of it below!

As shown above the height uses the Y variable, the depth uses the Z, and the width uses the X variable. Well that is all you need to know about it!

Let's render some cubes

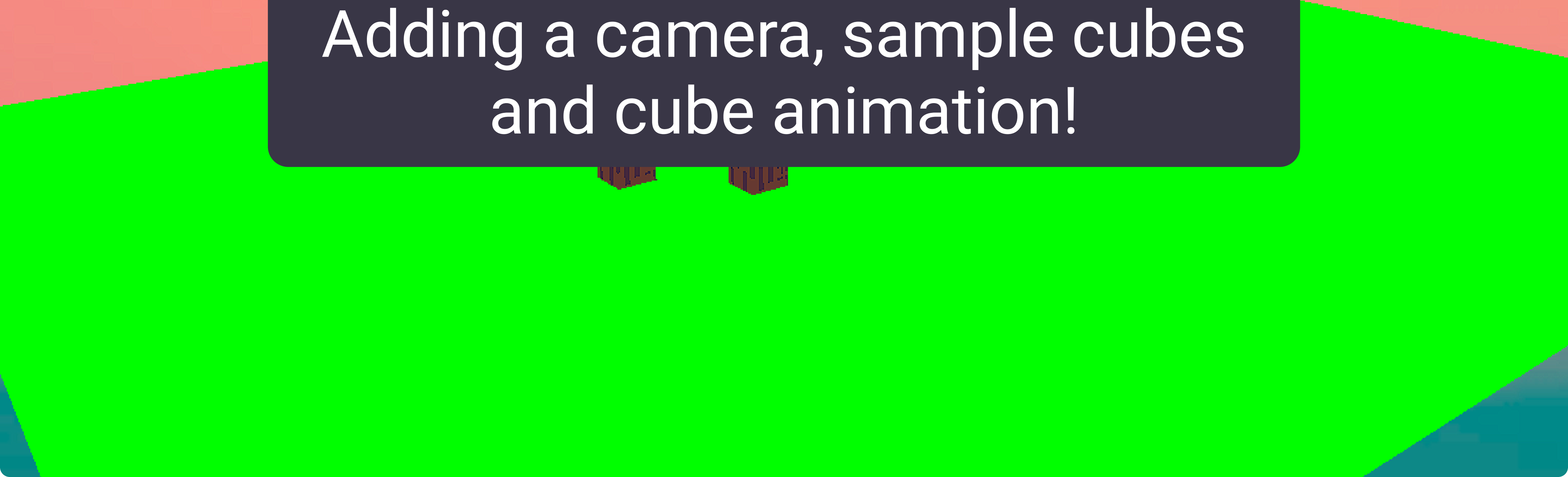






## OBJECTIVE #4

Adding a camera, sample cubes  
and cube animation!



# Scene, The world of our animation

We **need** to add a camera so we can render the scene!

To create the camera we will also **need a scene first**, so let's add that!

Copy this line above the render code, in order to initialize a new scene

```
const scene = new THREE.Scene(); // Init Scene
```

1



# Camera, Camera, CAMERA

Let's add it :D

Copy this line above the render code, in order to initialize a new scene

```
const camera = new THREE.PerspectiveCamera(  
    50,  
    window.innerWidth / window.innerHeight,  
    0.1,  
    1000  
); // Init Camera  
  
// Change camera position  
camera.position.z = 16;  
camera.position.y = 16 / 4;
```

- FOV: Camera frustum vertical field of view.
- ASPECT: Camera frustum aspect ratio.
- NEAR: Camera frustum near plane.
- FAR: Camera frustum far plane. For now use these defaults



# Cube ✨

To add a cube we first need to create a material.

A material defines the appearance of the cube.

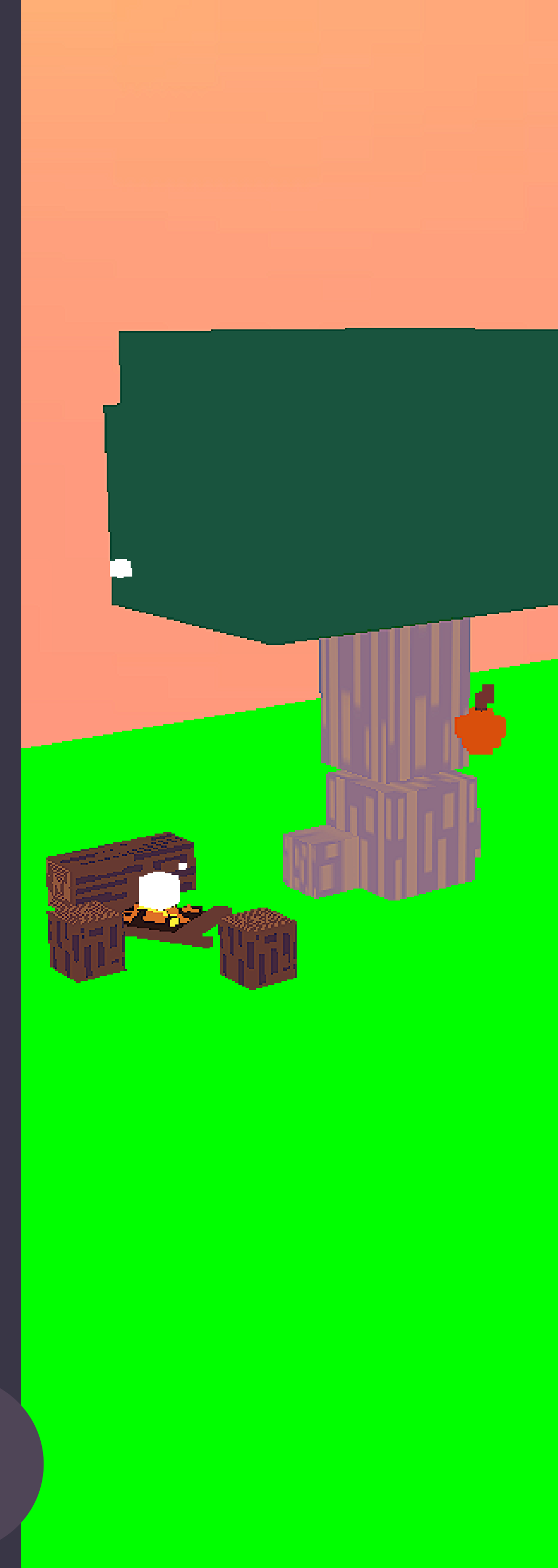
It can either be a color or a textures (called map), More about textures later.

In the first line we define the cube material, which the material that the cube will use.

Then we create a cube with the THREE.Mesh with the parameters of it's geometry aka size ( using THREE.BoxGeometry ) and the material, which we pass the cube material.

```
JavaScript
// Materials
const cubeMaterial = new THREE.MeshBasicMaterial({ color: 0xff0000 }); // We need
to add a material, each material takes the arguement inside an object called
color. It is in HEX.

// Add the objects
const cube = new THREE.Mesh(new THREE.BoxGeometry(1, 1, 1), cubeMaterial); // We
create a new cube, by default it DOES not exist in the scene
```



# Rendering the cube

To render the cube we first need to add it to the scene.

Specifically you will need to add it into the run arrow function, in order to render it only when the run button is clicked.

```
scene.add(cube);  
renderer.render(scene, camera);
```



# ✨ PRO TIP ✨

You may reuse the material for more than one cube, but if you want a different color (or texture in the future) split them into different variable.

**i.e.** All wood blocks would use a woodMaterial but glass will use a glass material.





# OBJECTIVE #5

Adding textures

# Adding the texture loader

Yes you heard that right! Forget the boring colors, let's add real textures!  
First upload your picture via dragging and dropping it in the assets/ folder.  
I will name mine cube.jpg

First we need to create a loader to load the image. Copy this code above the materials.

This just creates a loader, now we need to import the images.

```
const loader = new THREE.TextureLoader();
```

1





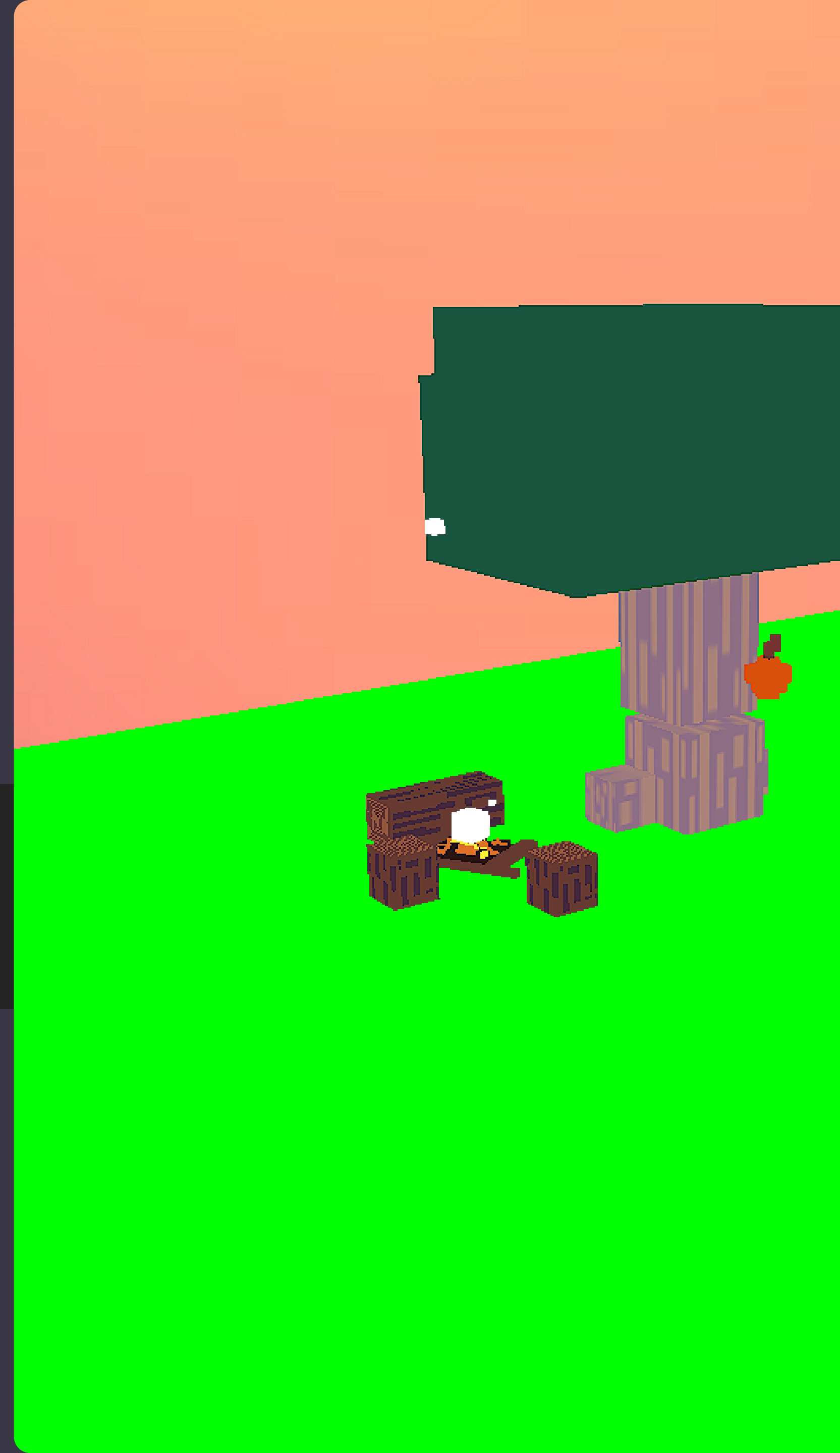
# Adding the texture loader

In the cube material replace the color: `0xWHATEVER`,  
with `map: loader.load('assets/cube.png')`

That is it! Wasn't that easy?

```
const cubeMaterial = new THREE.MeshBasicMaterial({ map:  
loader.load('assets/cube.png') });
```

2





OBJECTIVE #6

Three.js Must Haves

# Skybox

Well a black void is boring, I hope we can all agree with that!  
A skybox was already in the template in the assets/skybox.png :D  
Isn't that handy, you may replace it of course!

Now between the loader and the materials copy the code shown below.  
This will set the background to the assets/skybox.jpg ✨ !

```
scene.background = loader.load("assets/skybox.jpg");
```

1



# Background Audio

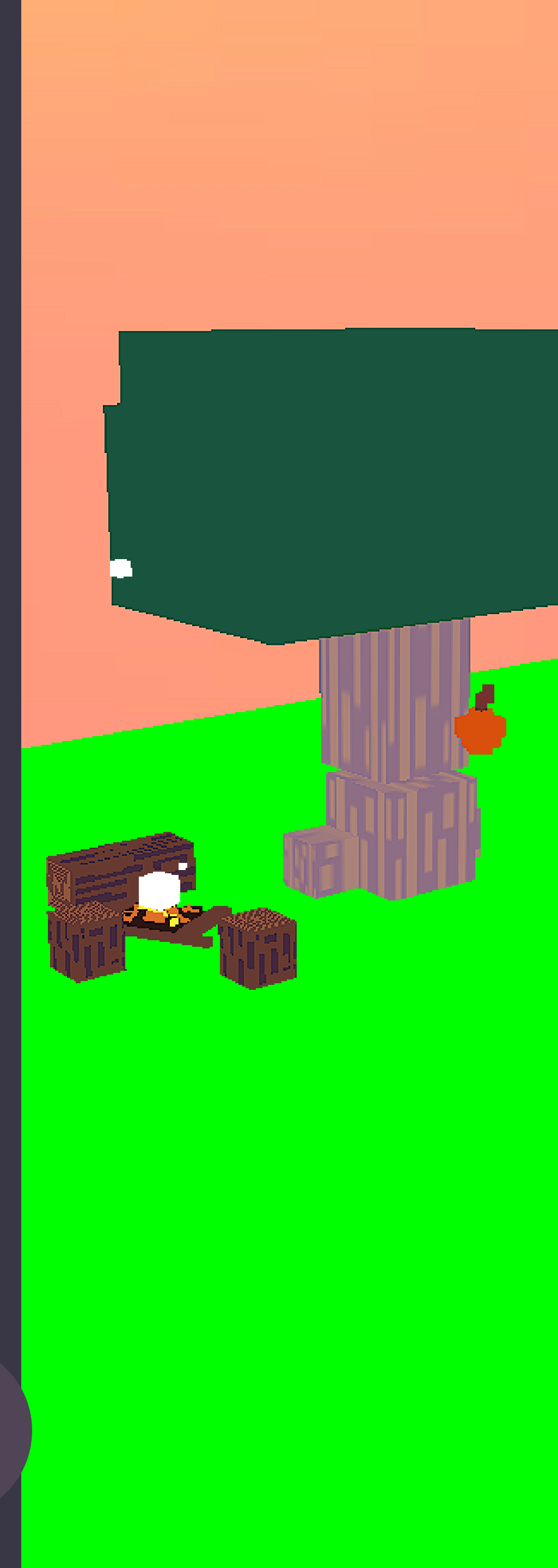
Now let's add some audio.

The audio takes place at the end of the run arrow function, just before the animate  
Copy this code the code shown below.

The order of the music must be last as to not interfere with the position / rendering.

```
JavaScript
const listener = new THREE.AudioListener(); // We create the listener
const soundSettings = new THREE.Audio(listener); // >> >> the audio settings
const audioLoader = new THREE.AudioLoader(); // >> >> the audio loader, so we can
load the bg
camera.add(listener); // We attach it to the camera

audioLoader.load("assets/bg.mp3", function (buffer) {
    soundSettings.setBuffer(buffer);
    soundSettings.setLoop(true); // Set the audio to loop
    soundSettings.setVolume(0.5); // Audio volume
    soundSettings.play(); // Play Audio
});
```



# Making the Camera Rotational

think we can agree that static cameras are boring so let's make ours to rotate!  
So what are you waiting! After the render code, copy the lines below. ( Picture 3 )

Now the last thing we need to do is call it at the end of our animate function! Just copy this line, before the render! ( Picture 4 )

```
JavaScript
// Controls
const controls = new OrbitControls(camera, renderer.domElement); // Create control
controls.target.set(0, 0.5, 0); // Set target
controls.update(); // Update it
controls.enablePan = false; // Disable pan
controls.enableDamping = true; // Enable damping
```

3

```
controls.update();
```

4





OBJECTIVE #7

Recording

# Capturer

Well the first thing we need to do is add the capturer so we can use it. Copy the code after picture 1 after the imports. Feel free to change the video format, if needed, but we prefer WEBM.

Now we need to add one more line just **before** the `renderer.render` function call on the **animate** method. ( Picture 2 )

```
capturer.capture(renderer.domElement);
```

2

```
var capturer = new CCapture({  
  format: "webm",  
  workersPath: "/lib/",  
  verbose: true,  
  framerate: 60,  
}); // Format may be: GIF, WEBM, PNG, JPG
```

1



# Start / Stop Recording

To start the recording it is very simple!

On the start rec arrow function add the code shown in picture 3.

Only problem is that we can't save the result D:

But fear not!

On the stop recording arrow function add picture 4.

```
capturer.start(); // Start the capturer  
statusRec.innerText = "Recording in progress"; // Update the status
```

3

```
capturer.stop(); // Stop the capturer  
capturer.save(); // Save the recording  
statusRec.innerText = "Saving recording... This *may* take a while";
```

4





# ✨ Final Tips and Farewell ✨

It is finally time! I hope you learned enough to start on your own.  
Make sure to keep these tips in mind!

- 1) Save all objects into variables
- 2) Create new materials for new objects, reuse materials for similar ones
- 3) Add objects to the scene on the run method
- 4) Change object position.rotation on run method *after* adding them to the scene.
- 5) Use the tick variable to keep track of time in your animations
- 6) Feel free to seek the demo code for more examples

For **advanced adventures**, seek the advanced section  
in the *text jam*.

