

A Multiplayer Game

— but in the Browser

Maya Nedeljković Batić @ Armada JS Novi Sad



Maya Nedeljković Batić



@maya_ndljik



@mayacoda





Why a multiplayer game?

Why in the browser?

Next.js Conf



ARMADA BATTLESHIP



github.com/mayacoda/armada-battleship

GDD

👉 focus on mobile

👦 11yo nephew
doesn't have to like it

🏆 must haves

🚢 how to set up
scenes

Game title

Armada Battleship

Intended game systems

Web browser, specifically on mobile so targeting *iOS Safari* and *Android Chrome*

Target age of players

Adults who attend JavaScript conferences

A summary of the game's story, focusing on gameplay

The player plays as a row boat captain in the open seas. They're surrounded by enemy boats and their goal is to defeat as many other players as possible. Players engage in 1 on 1 games of battleship with other players who are currently active. The games are shorter than usual (should be only a few minutes), allowing for a fast-paced experience.

Distinct modes of gameplay

There are two modes: exploration and battle mode. In exploration, the player controls a row boat through a 3D world in which they encounter other players. They can engage with other players in battle at which point each player is presented with a Battleship game interface.

Unique selling points

- Snappy game experience — games last only a few minutes
- Leaderboard — introduces competition to the conference
- Connects conference attendees in a shared experience



Maya 08/30/2022

Hey guys, I'm preparing a talk for a javascript conference about building a multiplayer game for the browser. Since it's a JS conf and the talk is about gamedev, I want to present some game dev/design best practices like writing a game design document, prototyping, etc.

My question to you is (1) what do you actually do in practice to organize yourselves when making games and (2) is there any gamedev advice you'd absolutely want to tell an audience of non-game devs?



rikoo 08/30/2022

2)

- Don't start with a MMORPG

Perfect recipe for an authentic game development experience

✓ tight deadline

✓ overpromising



Advice for (solo) Game Dev

- Iterate
- Fail fast with prototyping and blocking
- Use documents to align across creative fields
- Have a task list to manage scope creep
- Lean on assets and finished solutions



The Plan

Phases of development

-  Prototype
-  Blockout
- ✨ Final Touches



Prototype

- List of all players who are present in the game
- Players can challenge anybody
- Battleship game 1 on 1
 - Players take turns
 - Game lasts up to 5 minutes
 - Games are HTML based

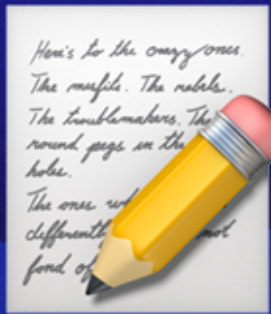


Blockout

- Players in a virtual world instead of list
- Physical proximity required to challenge
- Games are Three.js based

✨ Final Touches

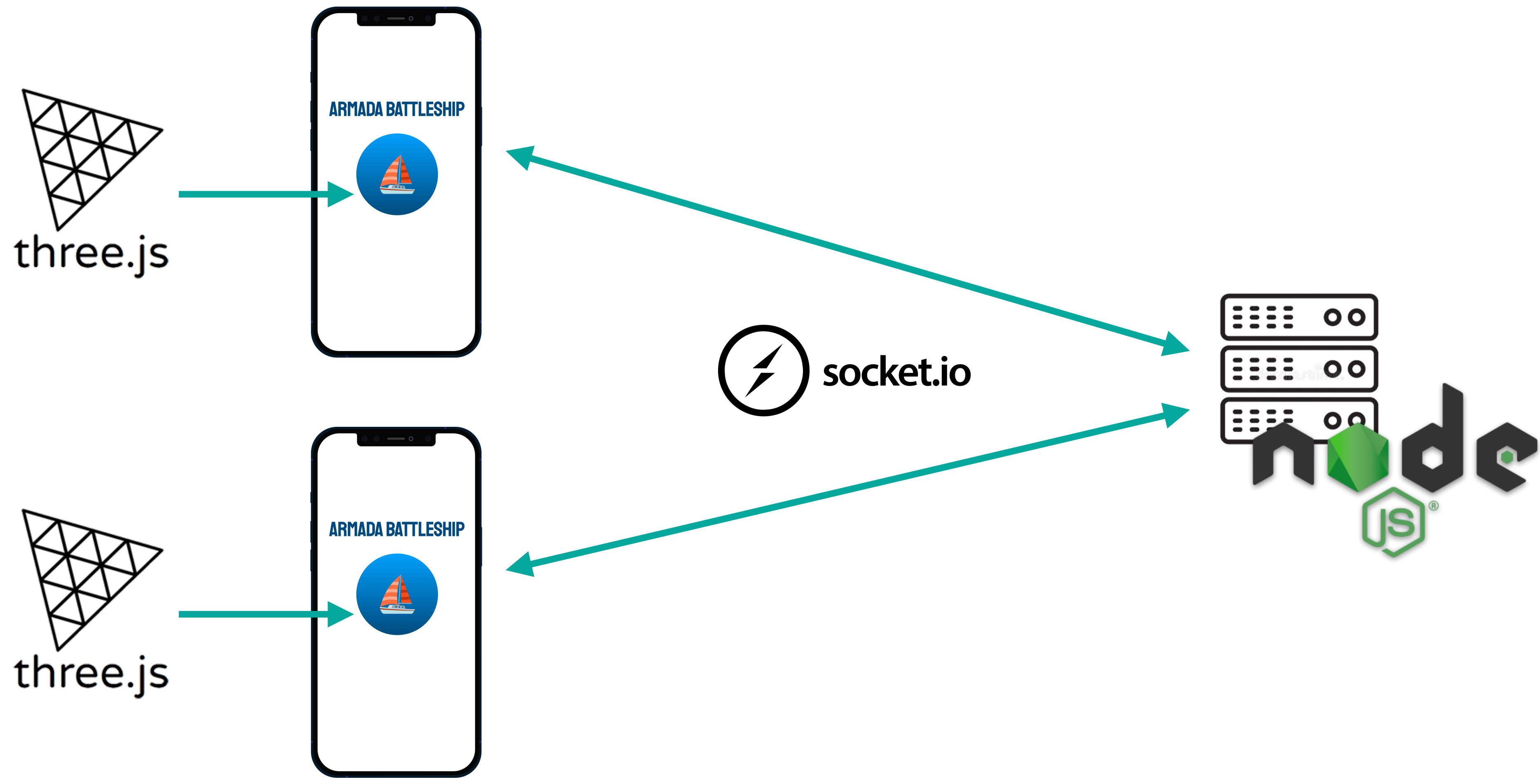
- Final art and assets
- Styling for UI
- Tweaking the gameplay



Prototype



Architecture



 Provide socket
connection

 Game logic



Server

 Manage players

 Synchronize state
between players

 Show game world

 Game mechanics

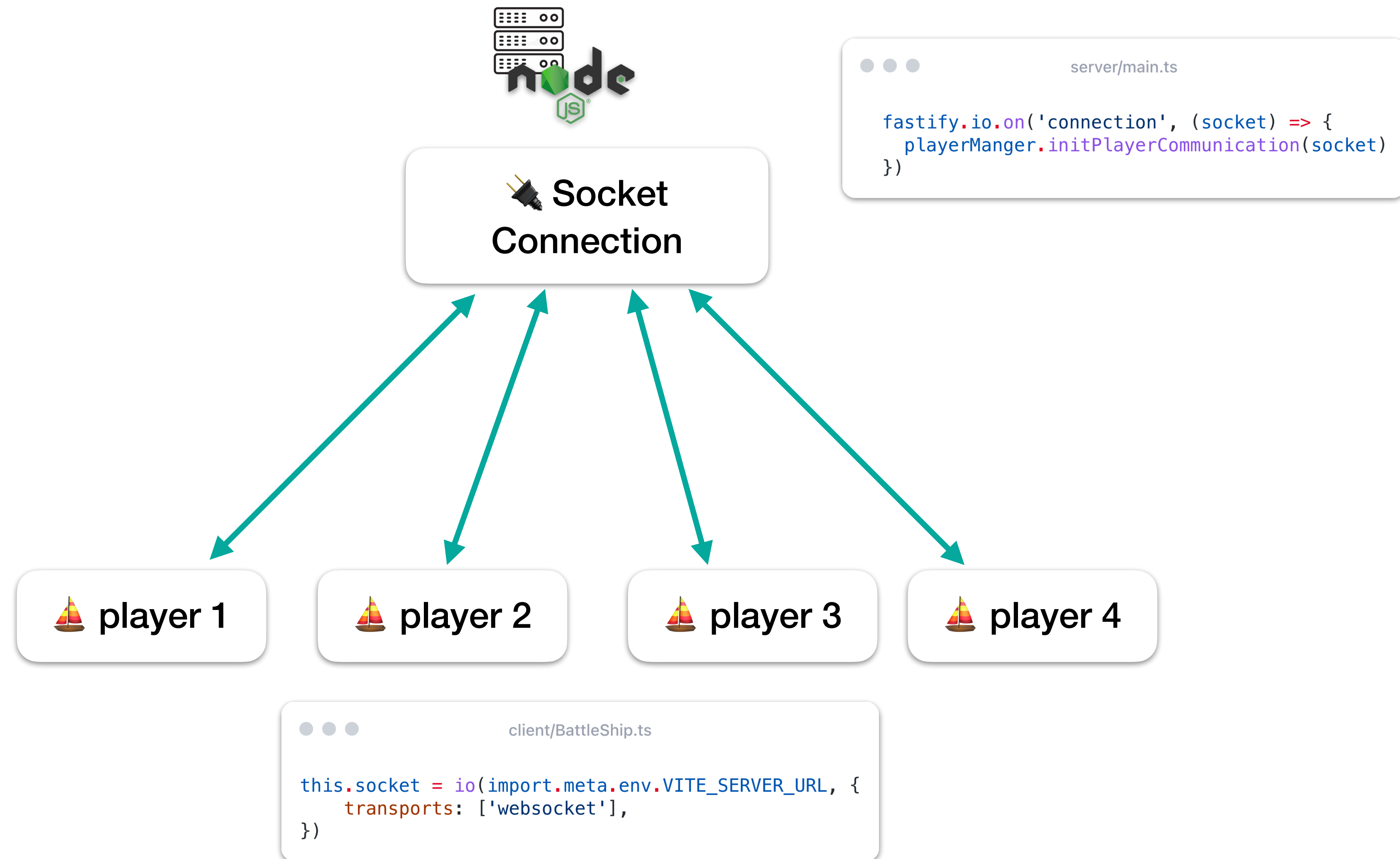


Client

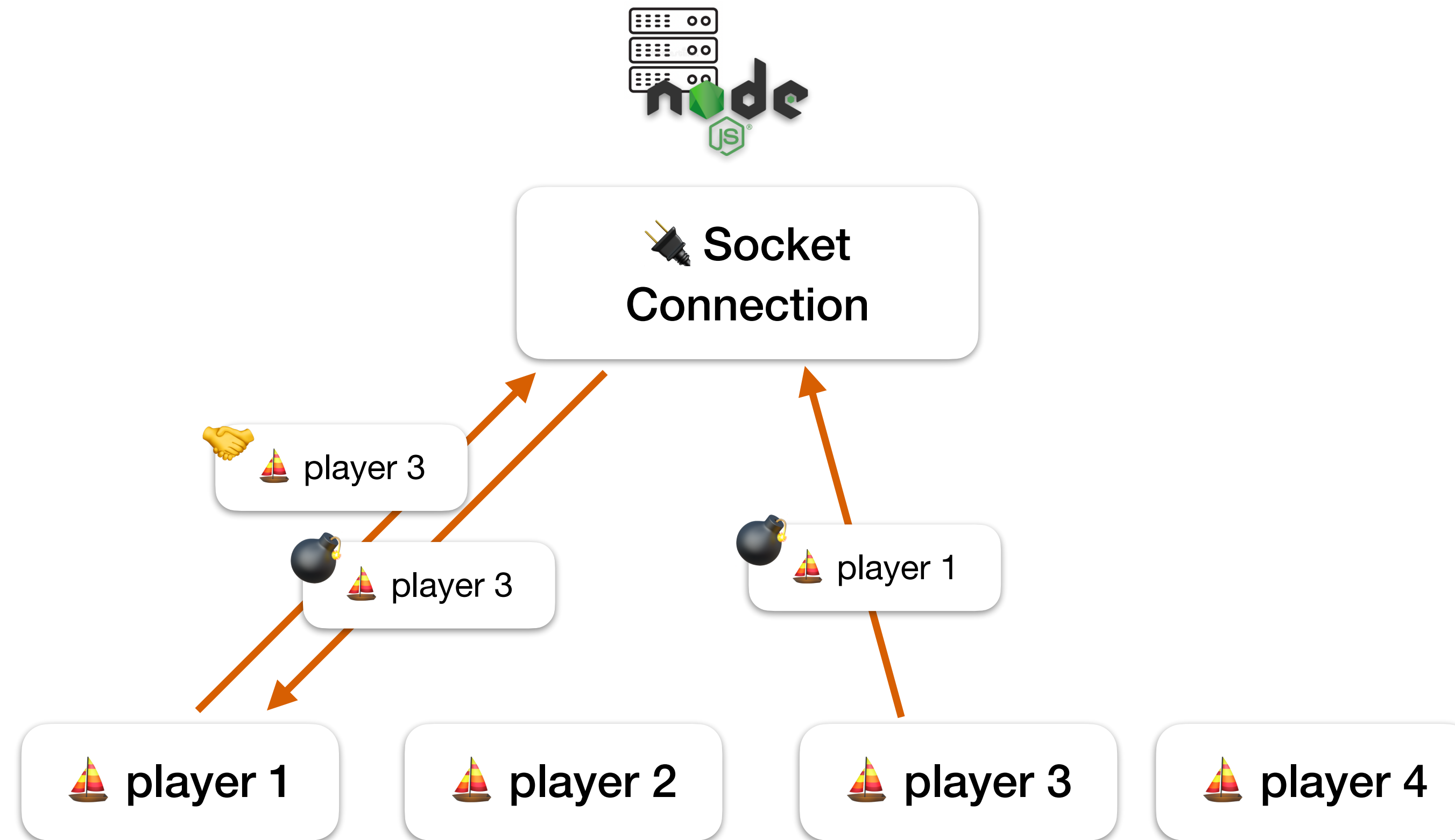
 Listen for player input
&
send to server

 Update state based
on server events

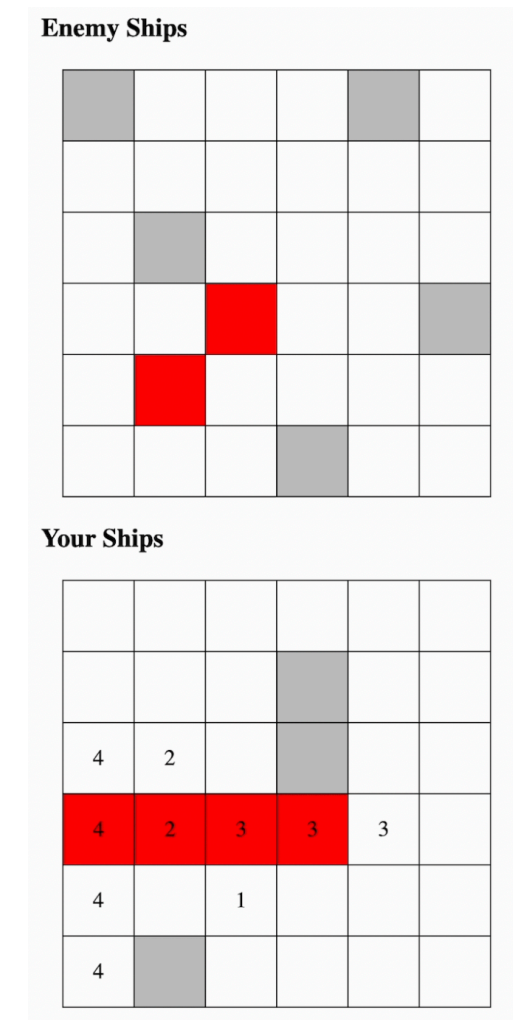
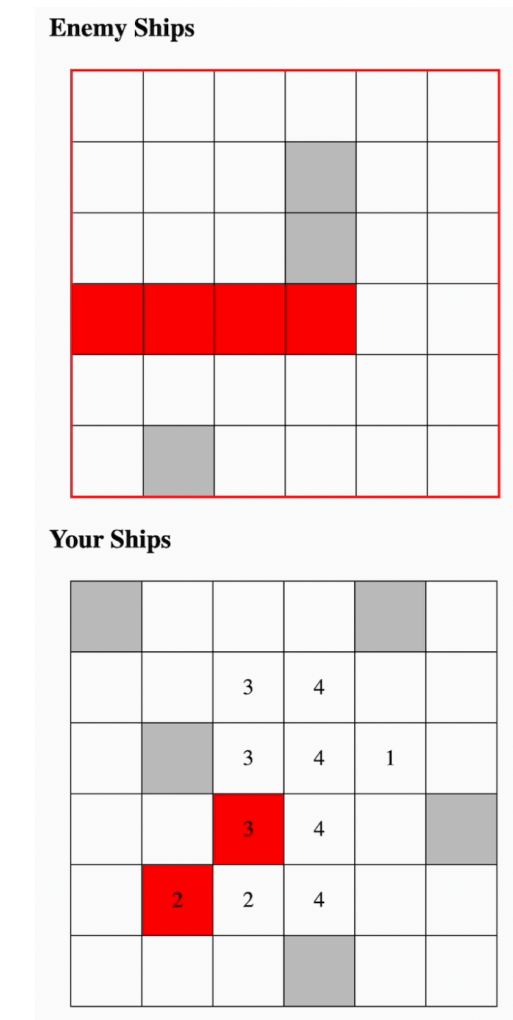
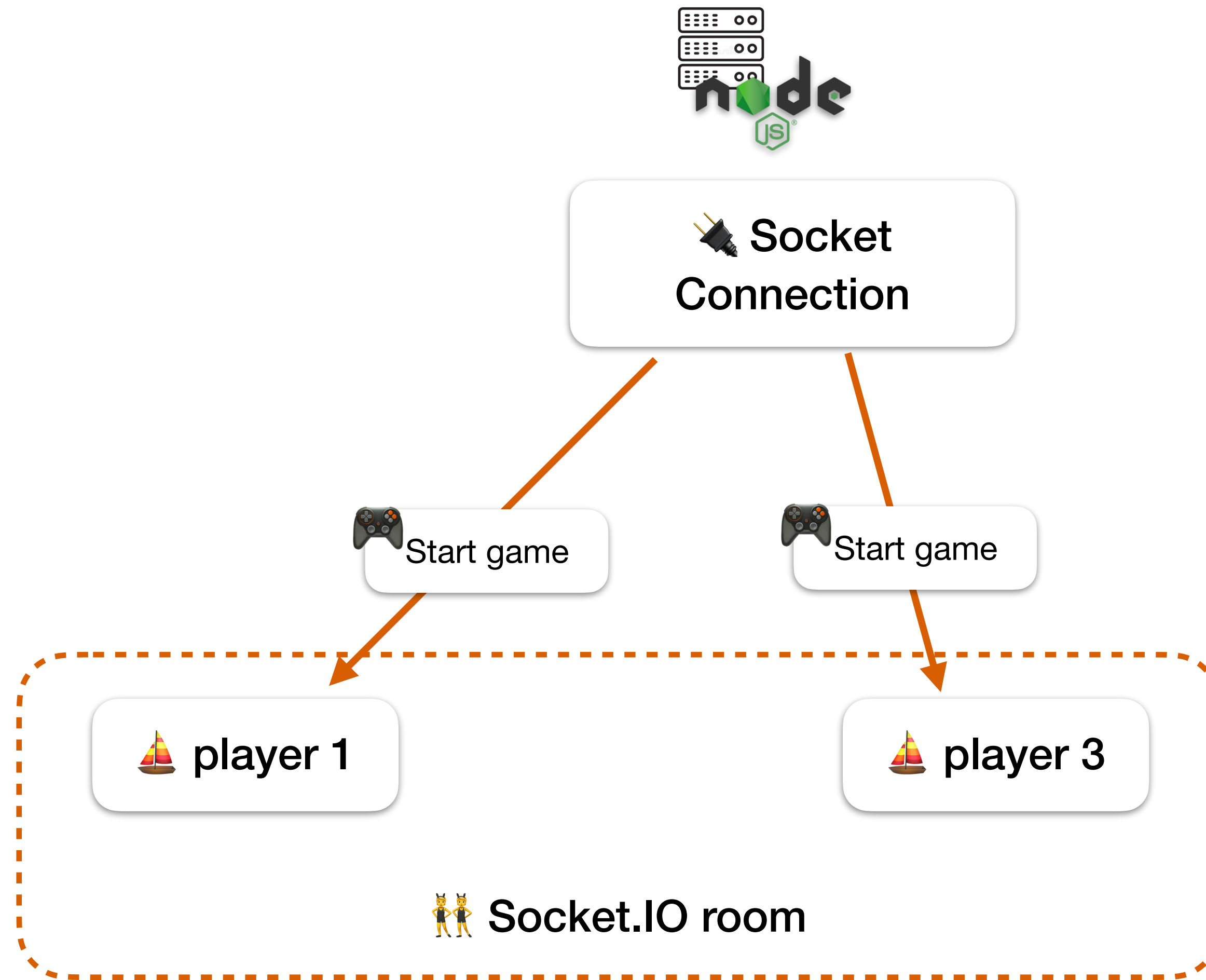
Establishing Client - Server Communication



Challenging another player



Starting a Battleship Match



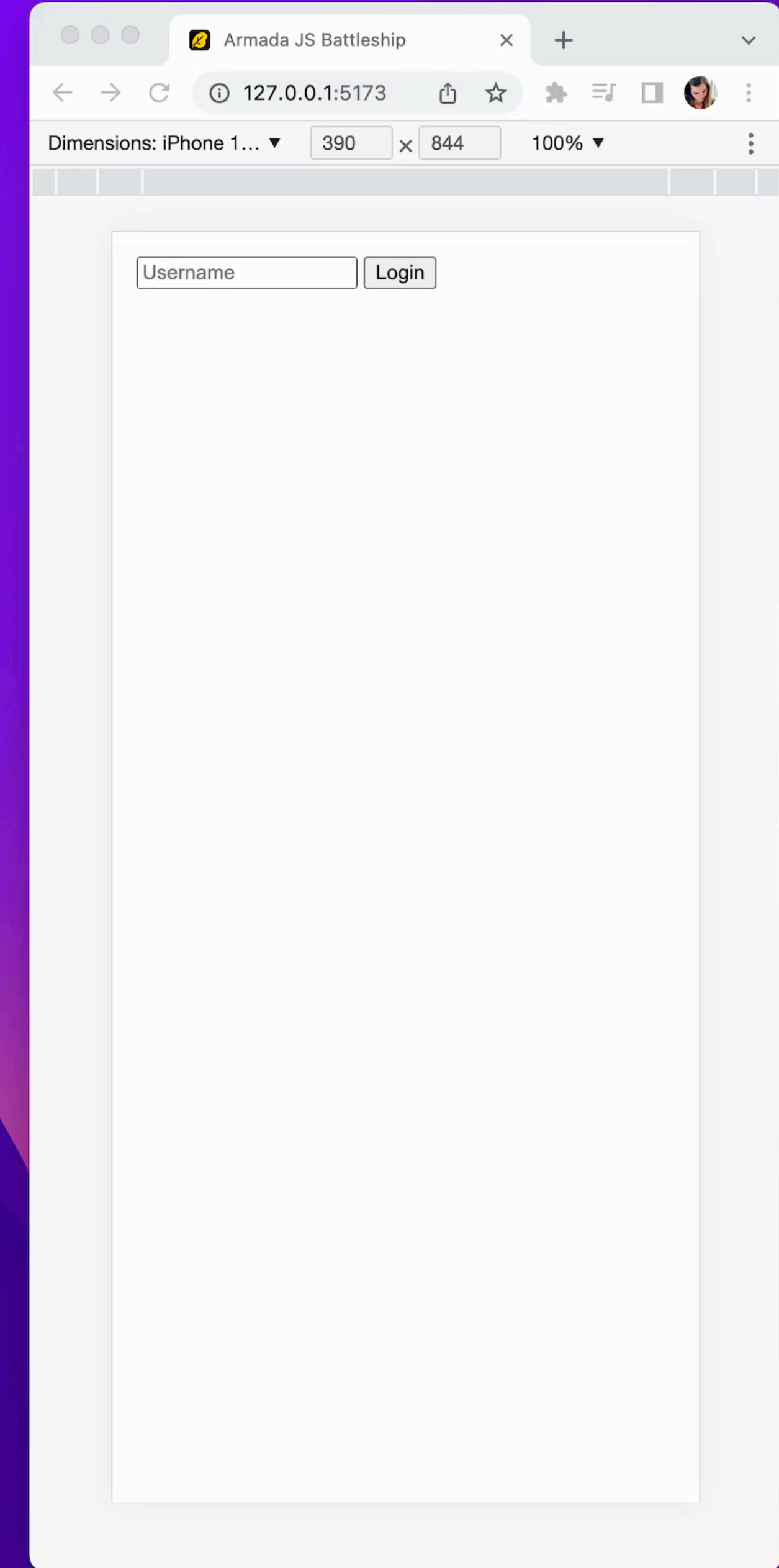
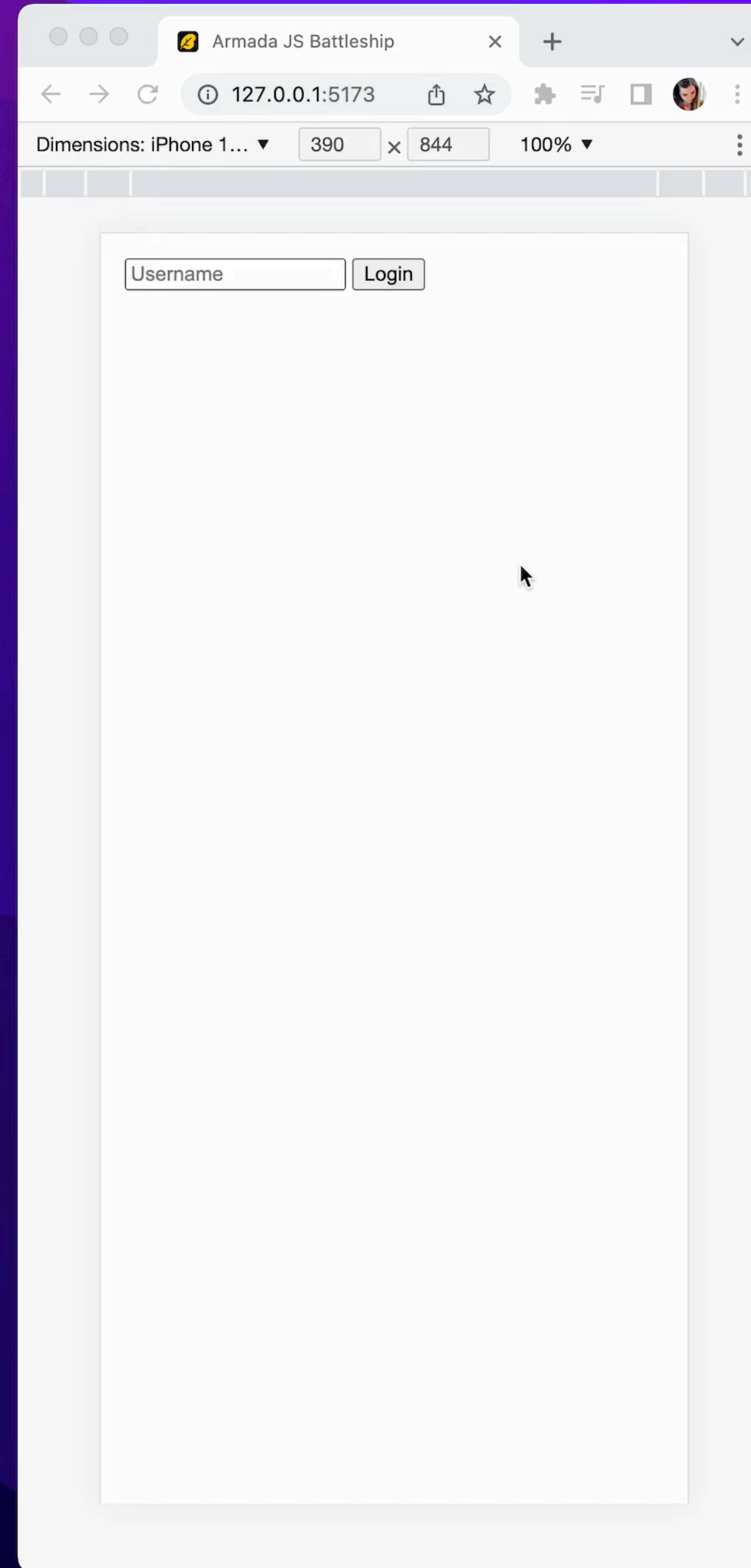
⚠ Gotchas when working with event base communication

- Order of subscribing and emitting events
- Throwing and catching errors
- Client socket listeners will eat up errors with a 'disconnect' event
- Keep an eye out for memory leaks
- Event order bringing your client or server into an invalid state

Back to basics

The HTML-based game

- ☒ List of all players who are present in the game
- ☒ Players can challenge anybody
- ☒ Battleship game 1 on 1
 - ☒ Players take turns
 - ☐ ? Game lasts up to 5 minutes
 - ☒ Games are HTML based





Blockout

Introducing 3D

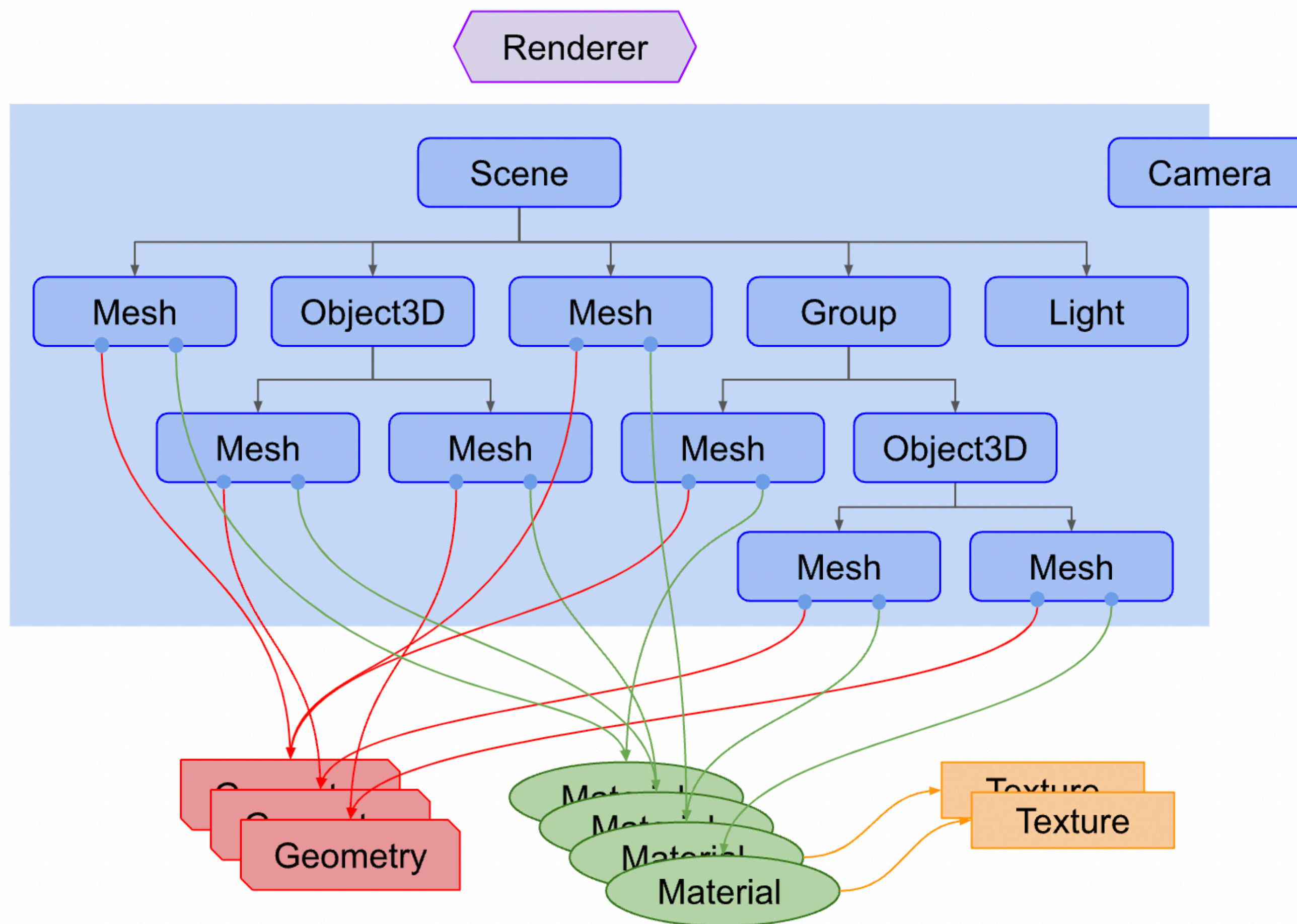
🏛️ Poor man's
game play

```
<html lang="en">
  ▶ <script>...</script>
  ▶ <head>...</head>
  ▼ <body>
    ▶ <div id="ui" style="height: 1019px;">...</div>
      <canvas id="canvas" data-engine="three.js r141" width="474" height="2038" style="width: 237px; height: 1019px;">
    </body>
  </html>
```

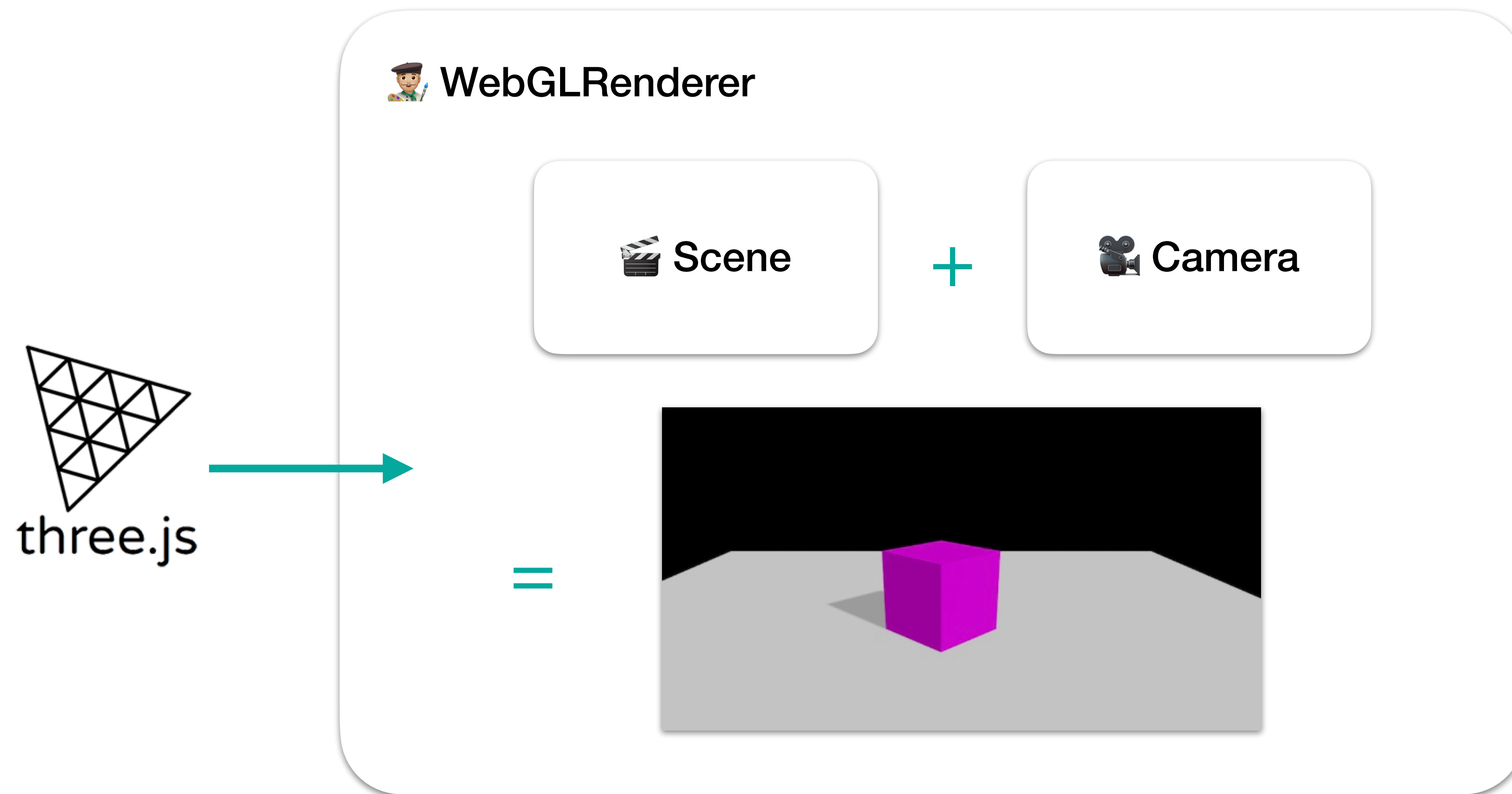
✨ Three.js canvas

Representing 3D things on a 2D picture

Scene Graph

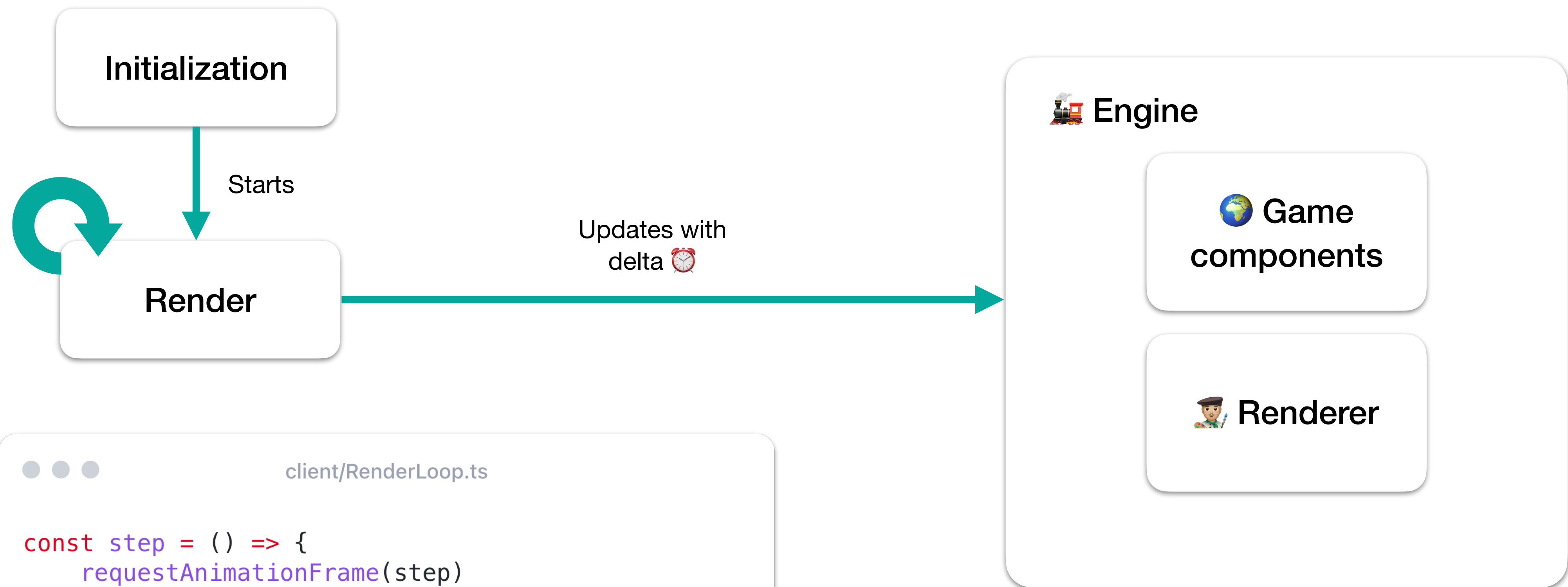


✨ Rendering the Scene





Render Loop



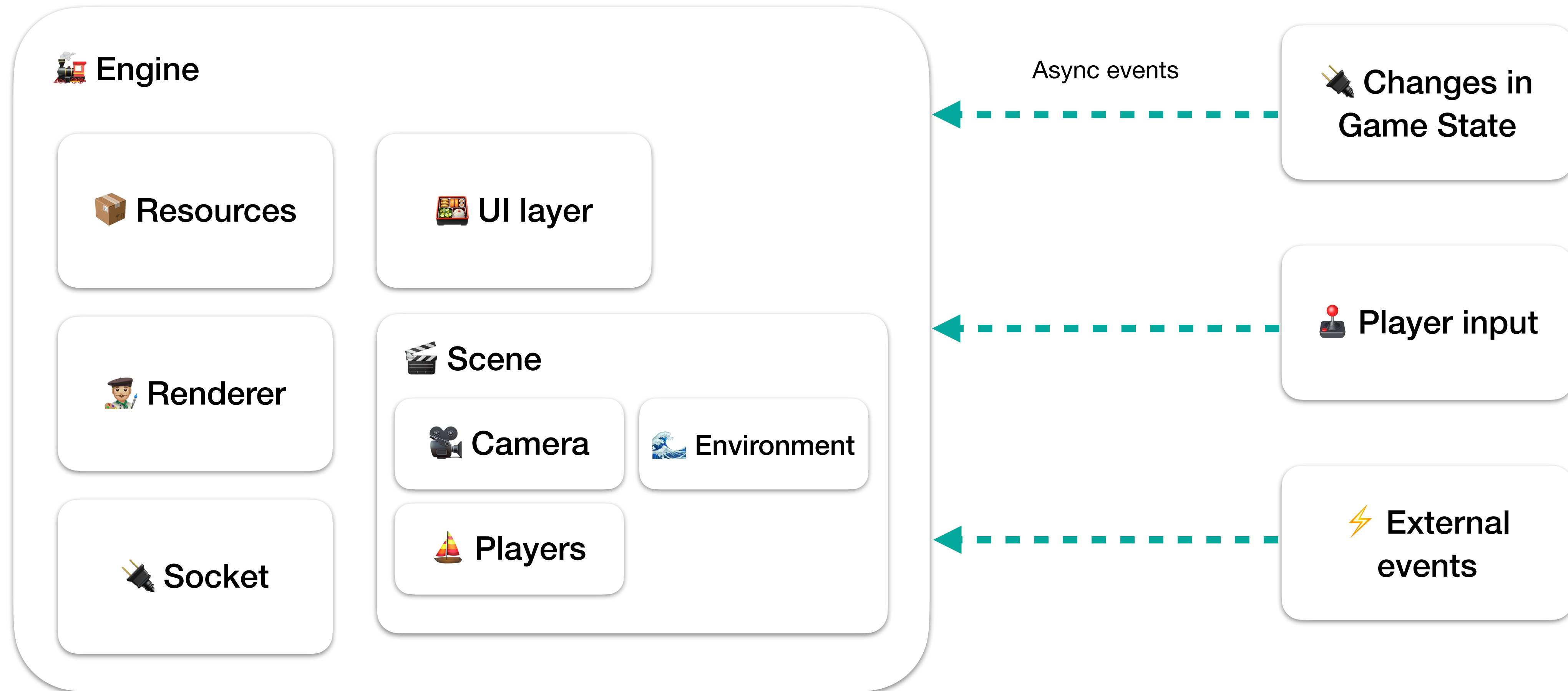
```
client/RenderLoop.ts

const step = () => {
  requestAnimationFrame(step)
  const elapsedTime = this.clock.getElapsedTime()
  this.deltaTime = elapsedTime - this.currentTime
  this.currentTime = elapsedTime

  this.engine.update(this.deltaTime)
}
step()
```




Game (Engine)



Building a 3D game

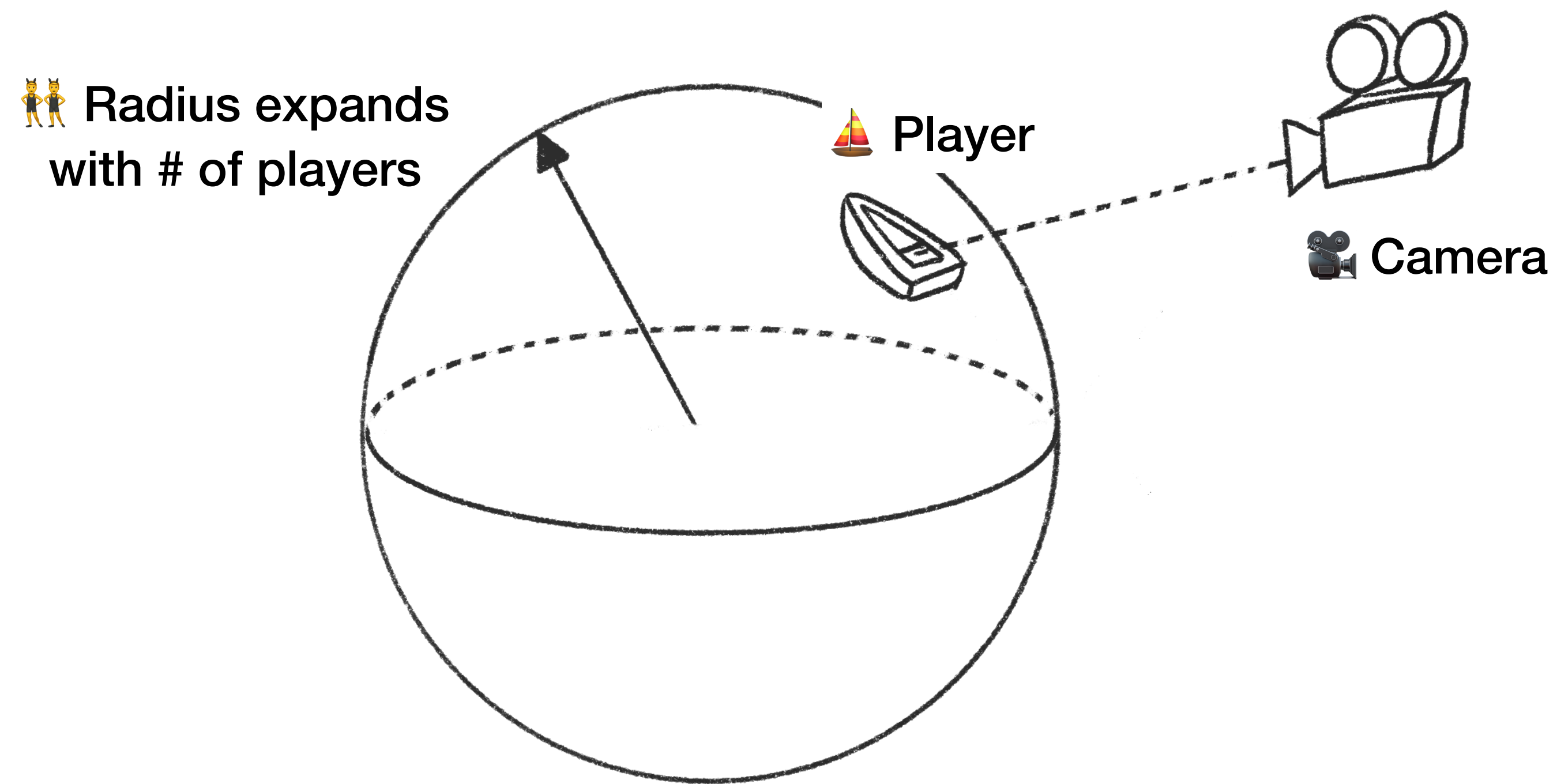
Although much of your game design is always changing, you need to establish three fundamentals early in your preproduction. I call them the “Three Cs”:

1. Character
2. Camera
3. Controls

Scott Rogers — *Level Up! The Guide to Great Video Game Design*




Camera & Controls

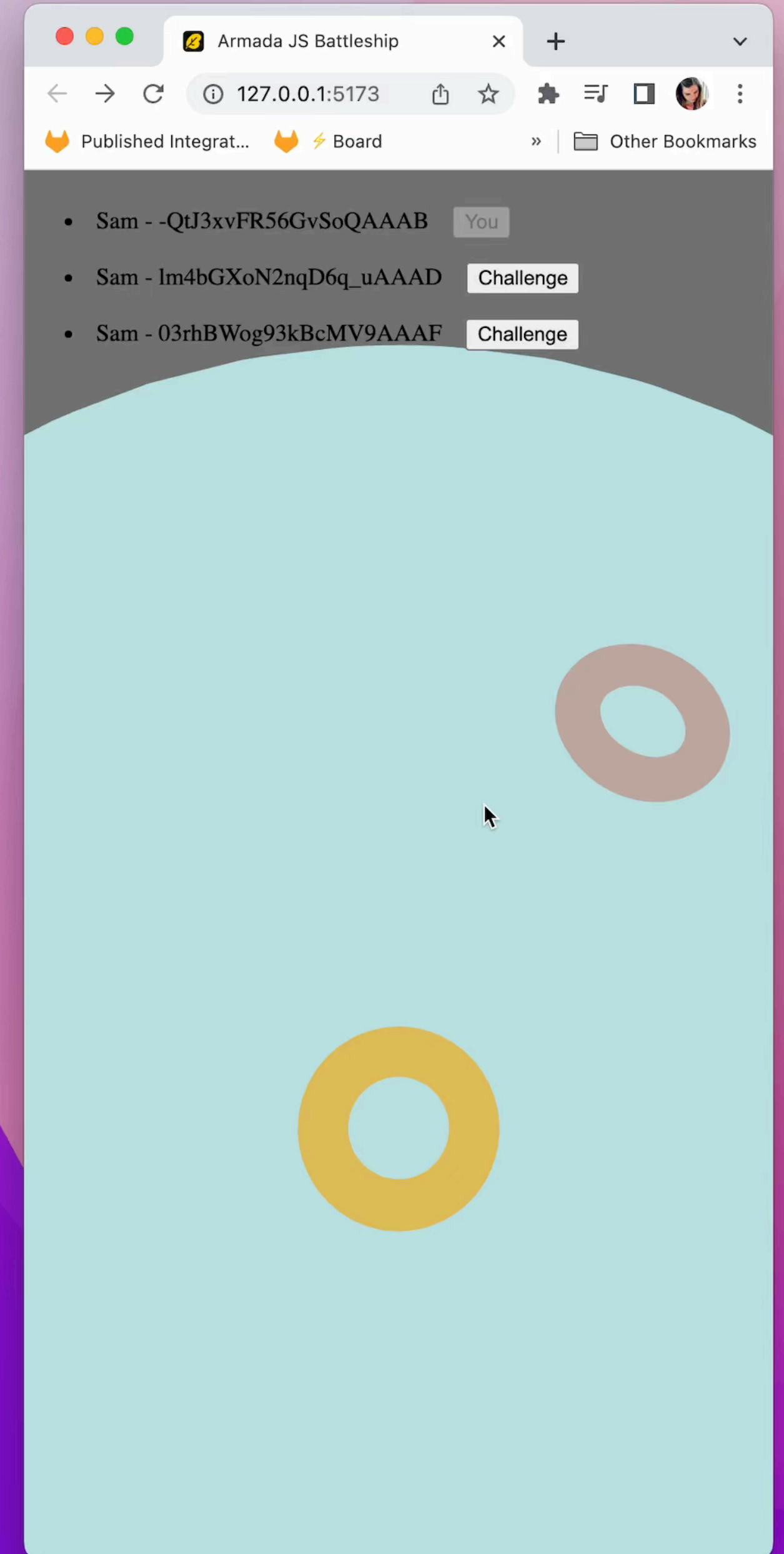
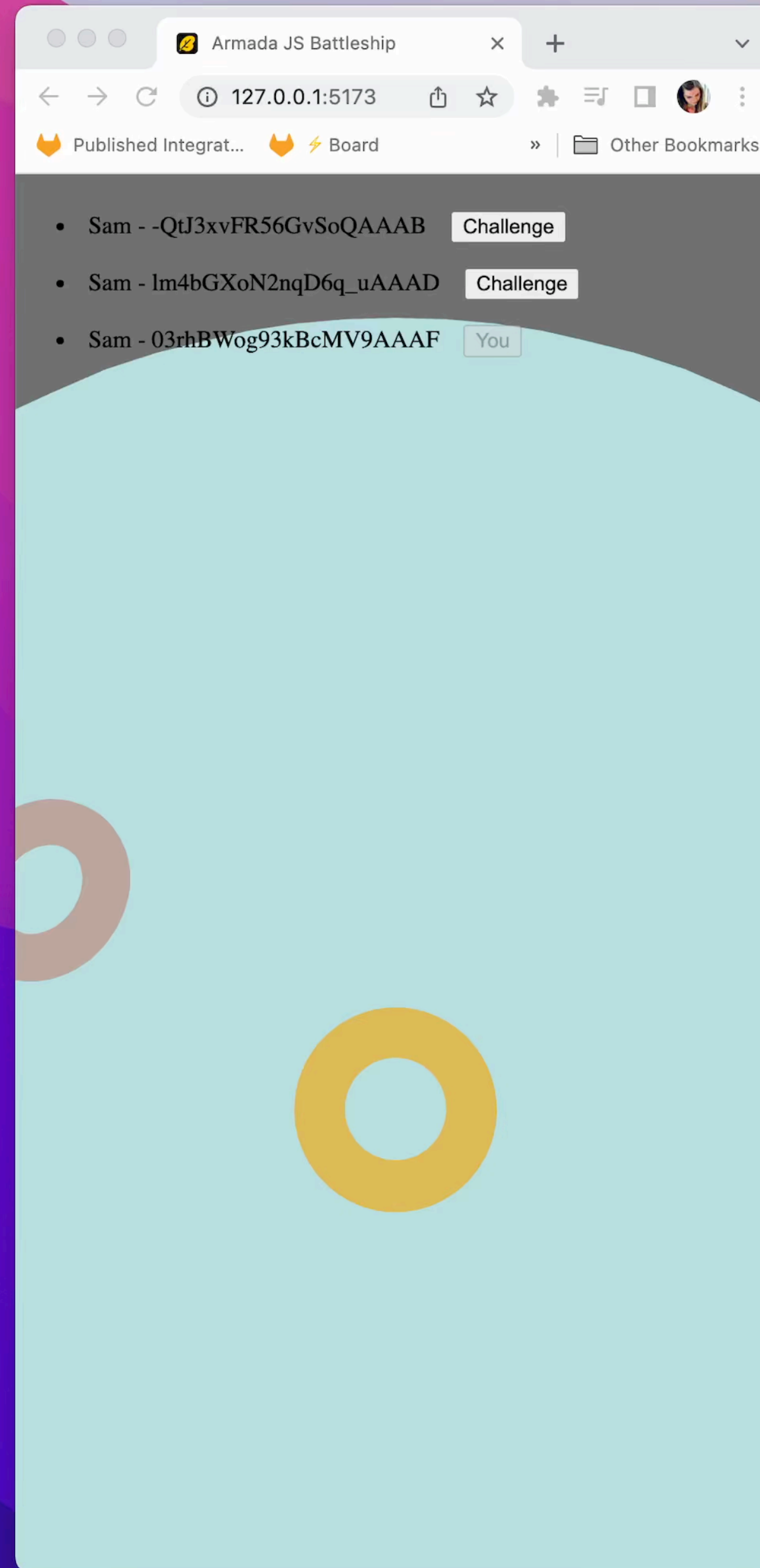
Replacing the “player list” with a globe



Camera & Controls

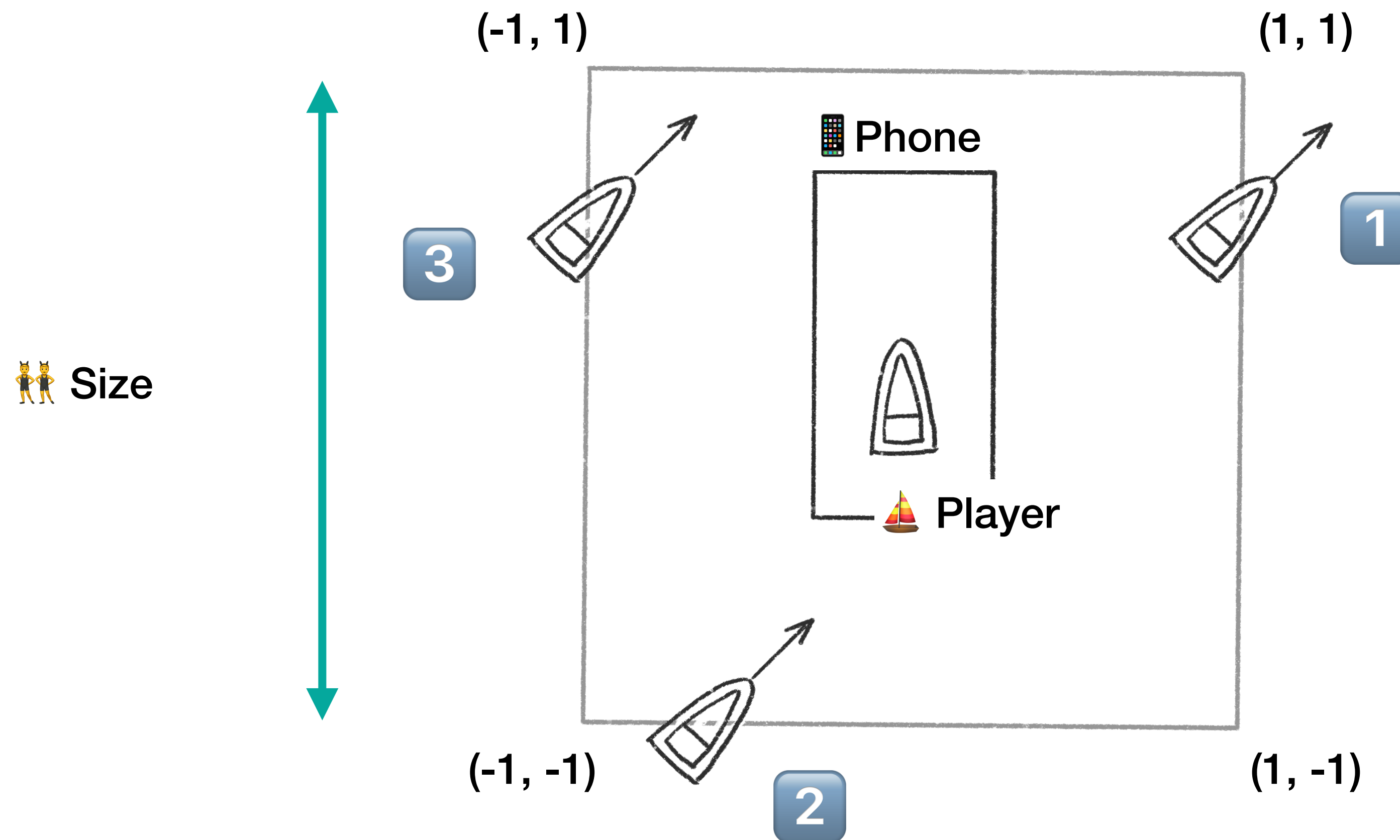
Replacing the “player list” with a globe

-  Clicking/tapping to move
-  Camera following player on globe falls into gimbal lock
-  Movement on globe is awkward






Camera & Controls

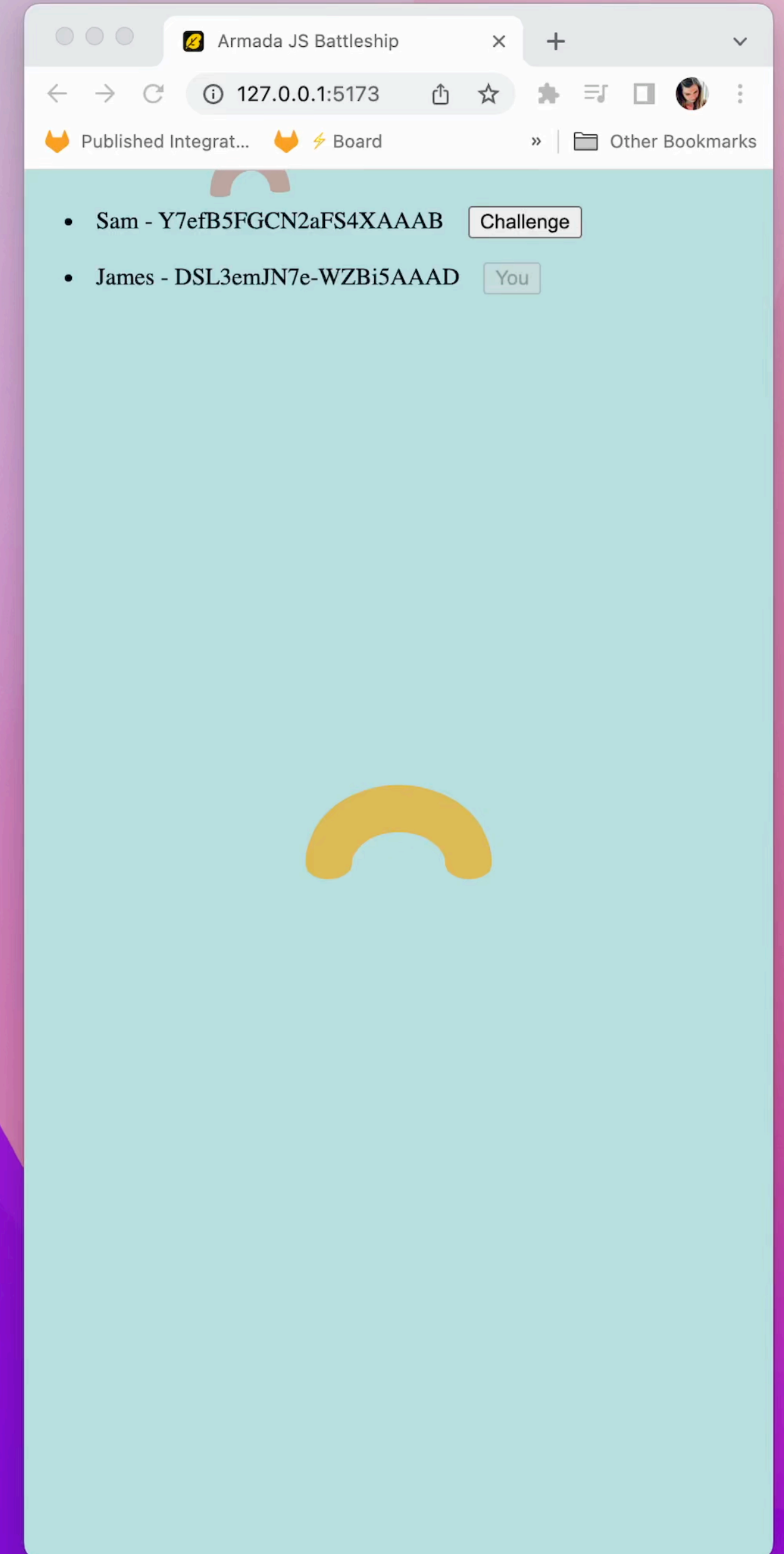
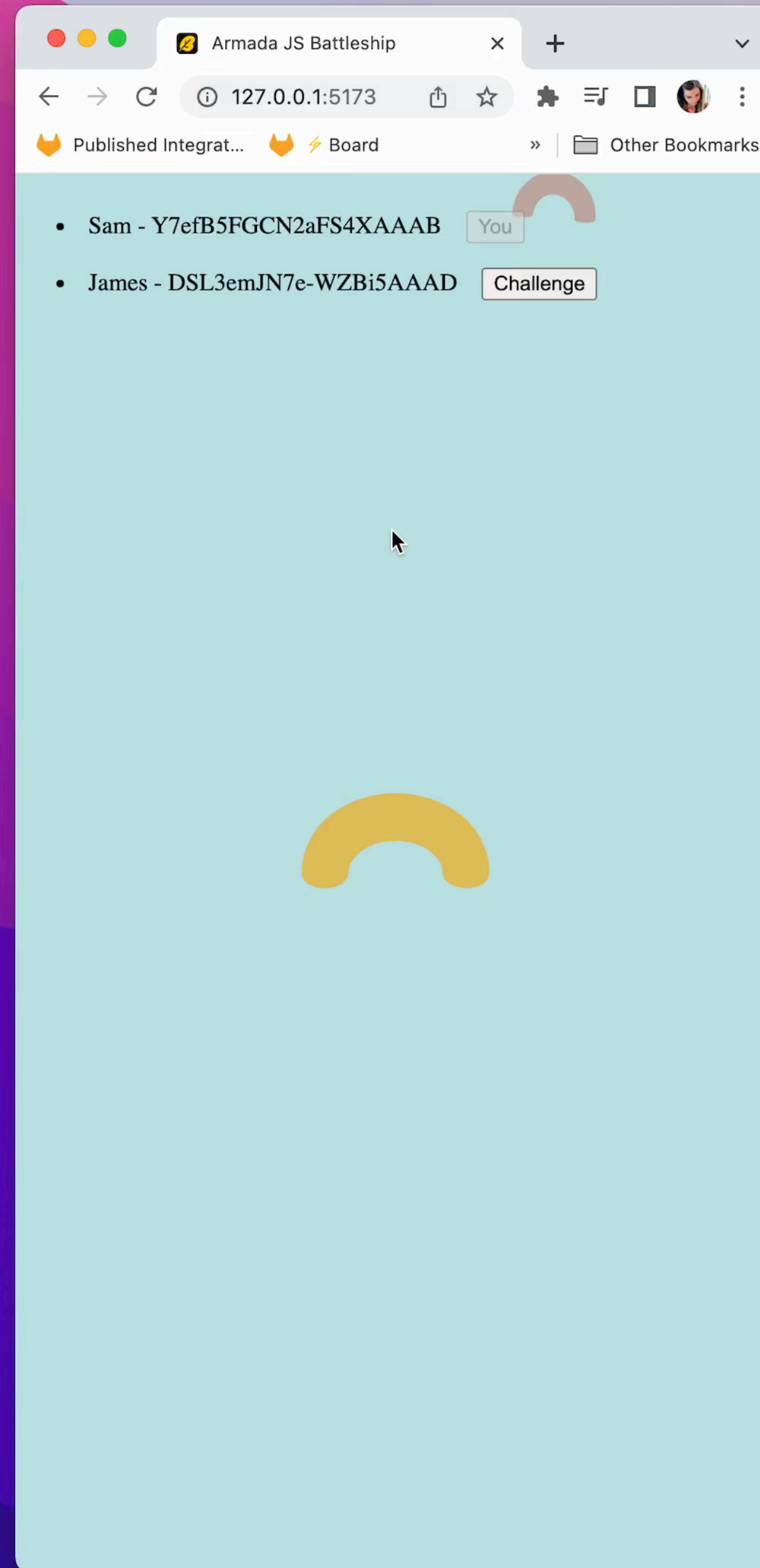
Replacing the “player list” with an infinite plane



Camera & Controls

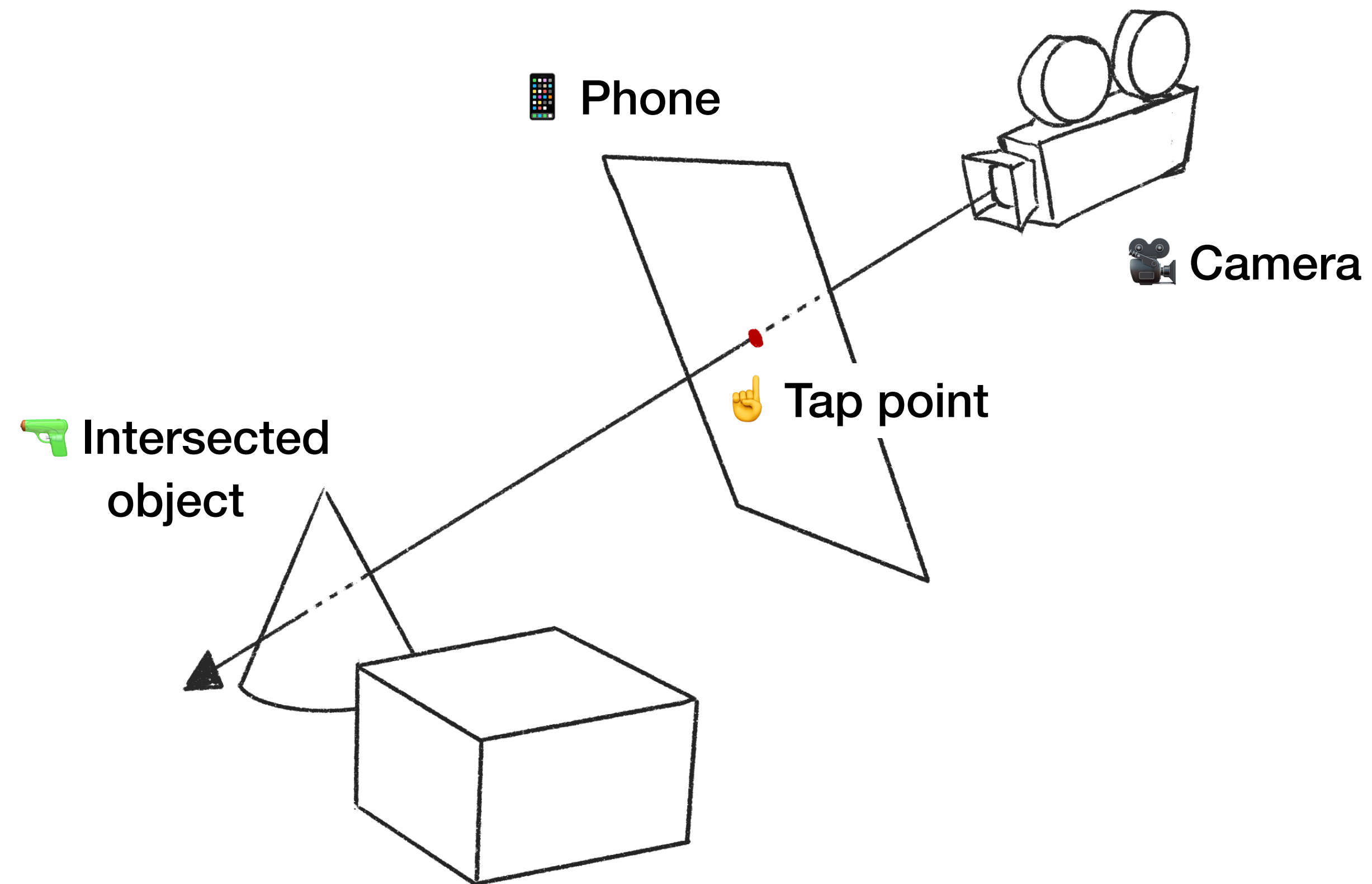
Replacing the “player list” with a plane

-  Clicking/tapping to move
-  Camera follows smoothly
-  Movement not entirely accurate on opponent's screen



Challenging a player

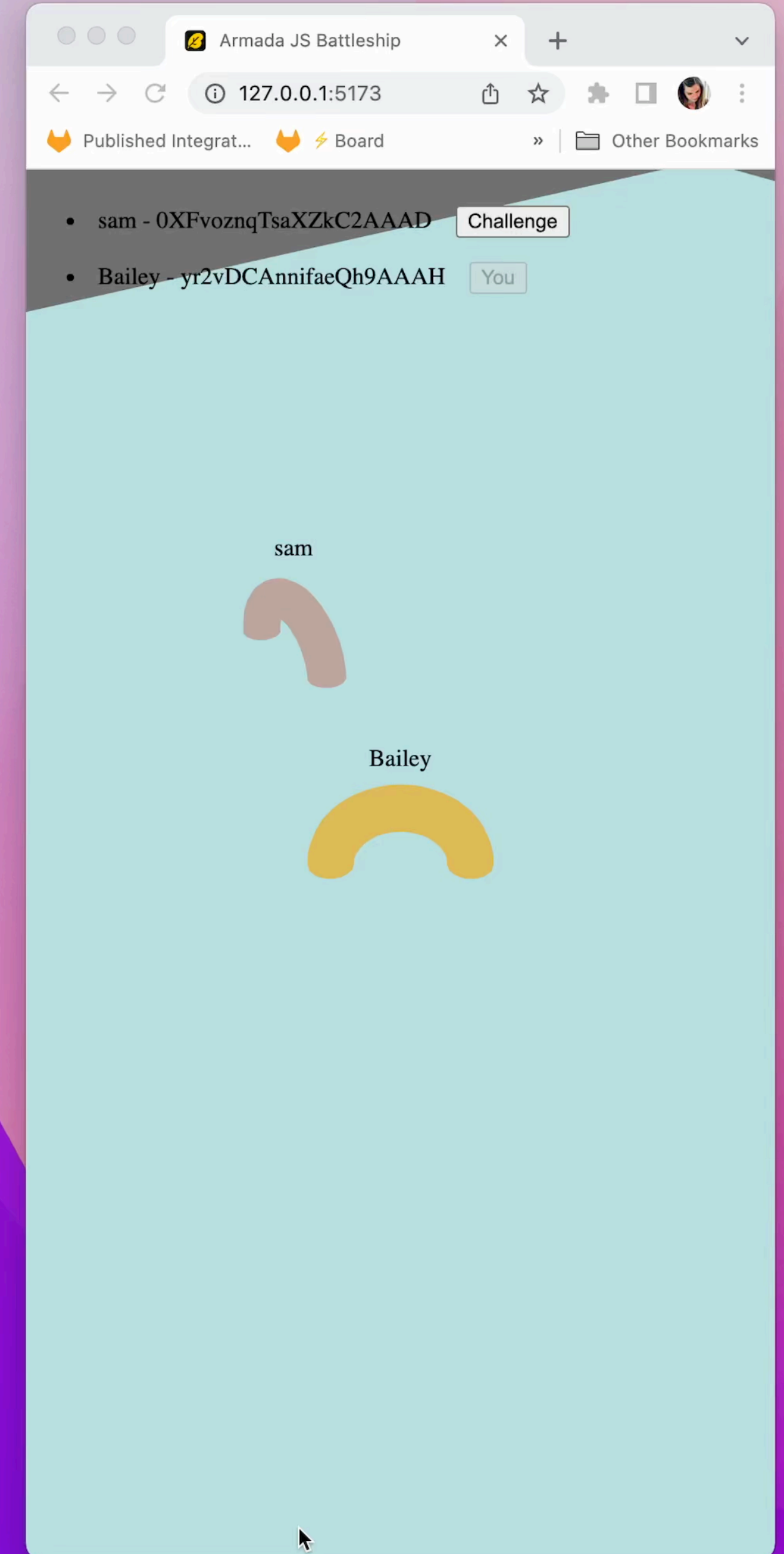
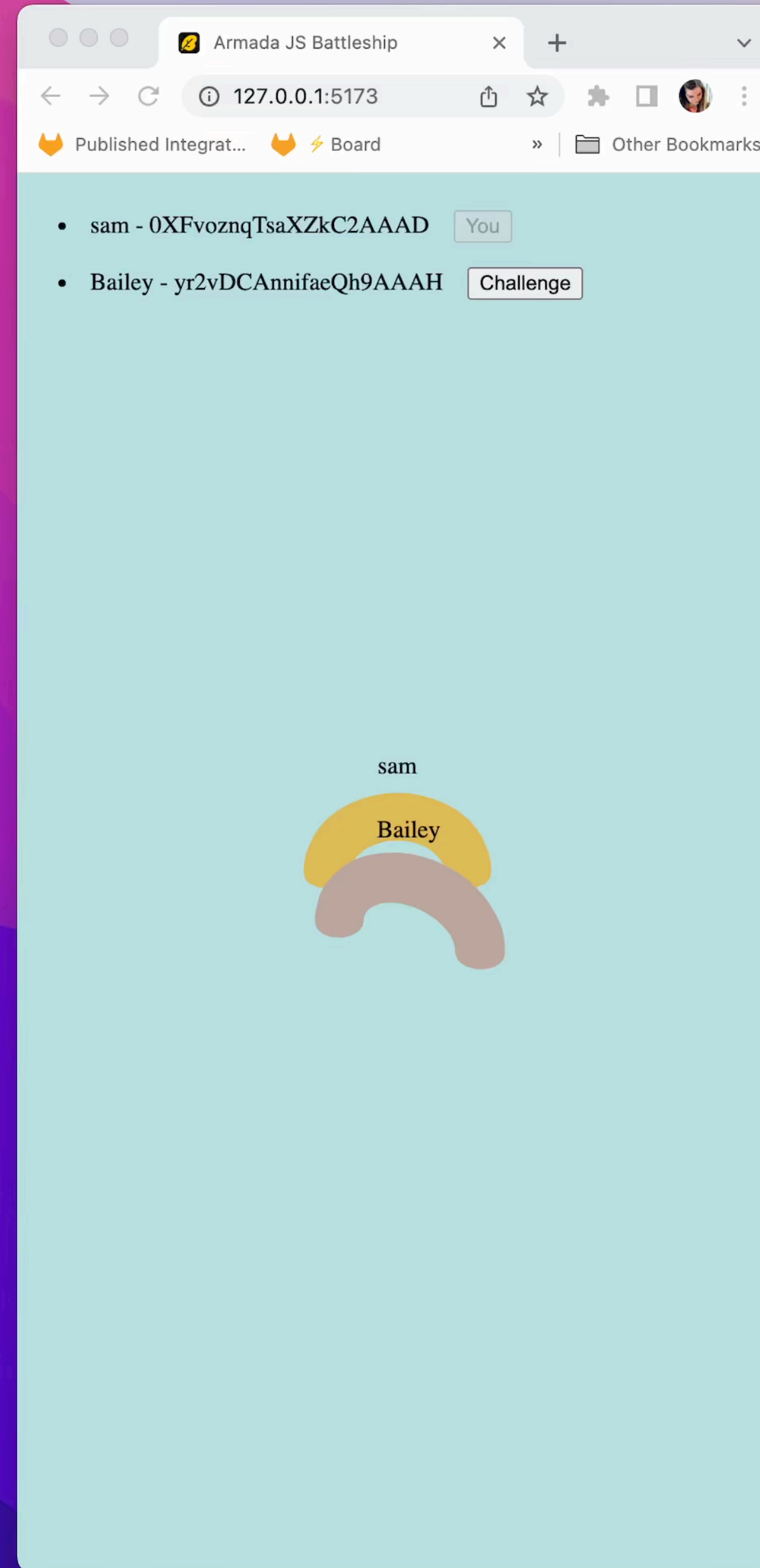
Raycasting instead of clicks



Challenging a player

Raycasting instead of clicks

- ✓ Clicking on player shows challenge button
- ✓ Button shown next to player on screen
- ✓ Challenge button still works! 🎉
- 🐛 Movement not entirely accurate on opponent's screen



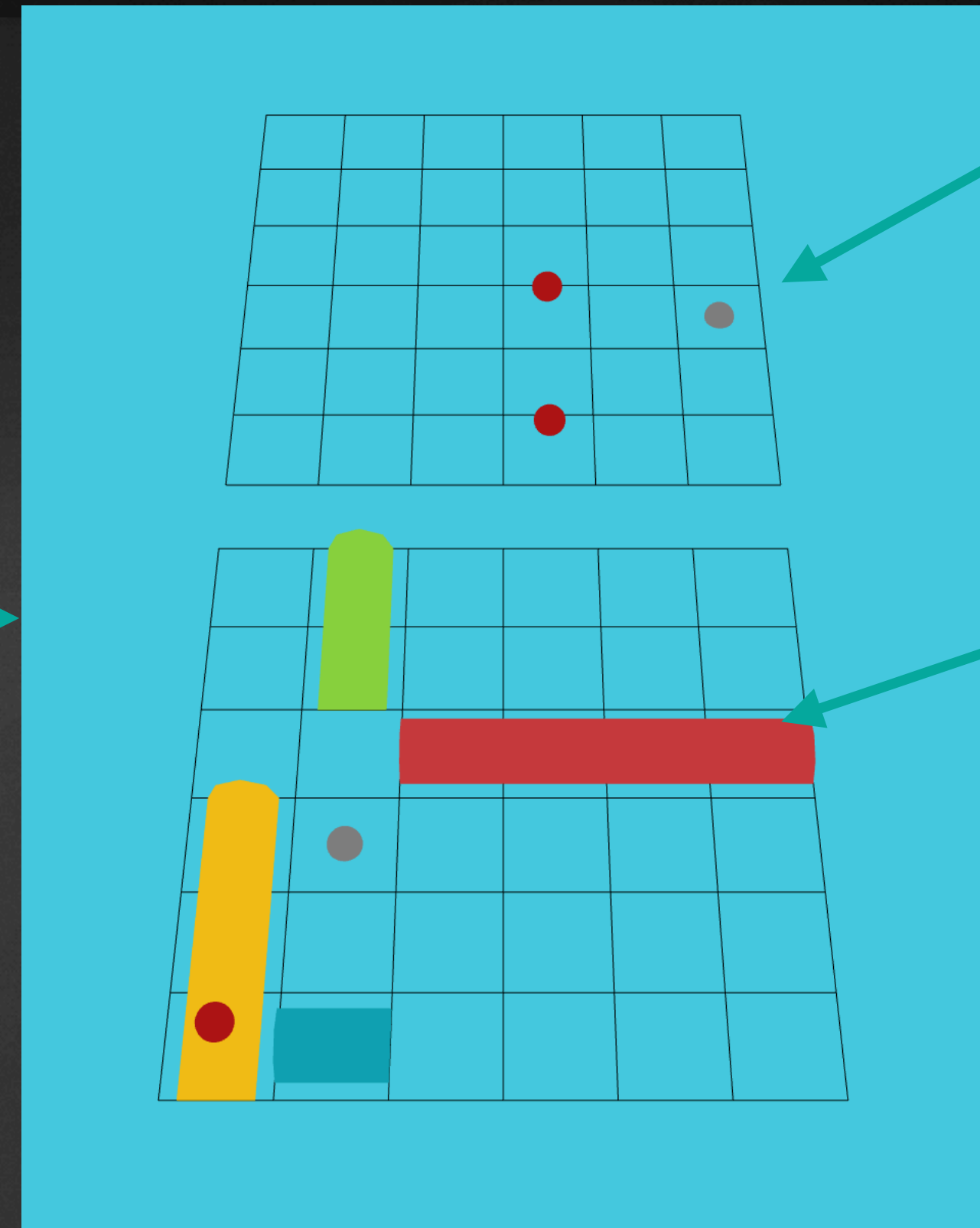
Battleship Gameplay in 3D

Enemy Ships

			Gray		
			Gray		
Red	Red	Red	Red		
	Gray				

Your Ships

		3	4		
		3	4	1	
		3	4		
	2	2	4		



client/GameBoard.ts

```
const enemyGrid = new THREE.GridHelper(
    this.gridWidth,
    GRID_SIZE,
    '#000',
    '#000'
)
```

client/GameBoard.ts

```
const geometry = new THREE.CylinderGeometry(
    radius,
    radius,
    ship.type * this.squareWidth
)
const material = new THREE.MeshBasicMaterial({ color: 0x00ff00 })
const mesh = new THREE.Mesh(geometry, material)
```


Lots of Math

```
client/GameBoard.ts

mesh.position.copy(this.playerGrid.position)
mesh.position.x -= this.gridWidth / 2
mesh.position.y -= this.gridWidth / 2

mesh.position.x += ship.start.x
mesh.position.y += ship.start.y

if (ship.direction === 'horizontal') {
  // move the ship's center
  mesh.position.x += (ship.start.x - mesh.position.x)
  mesh.rotation.z = Math.PI
  // move mesh up by half a square
  mesh.position.y += this.squareHeight / 2
} else if (ship.direction === 'vertical') {
  // move the ship's center
  mesh.position.y += (ship.start.y - mesh.position.y)
  // move mesh right by half a square
  mesh.position.x += this.squareWidth / 2
}
```



```
client/GameBoard.ts

private markResult(grid: GridHelper, x: number, y: number, hit: boolean) {
  if (hit) {
    const geometry = new THREE.SphereGeometry(0.1)
    const material = new THREE.MeshBasicMaterial({ color: '#b01717' })
    const point = new THREE.Vector3(x, y, grid.z)
    const mesh = new THREE.Mesh(geometry, material)
    mesh.position.copy(point)
    grid.add(mesh)
  }
}
```

```
const geometry = new THREE.SphereGeometry(0.1)
const material = new THREE.MeshBasicMaterial({ color: '#807d7d' })
const mesh = new THREE.Mesh(geometry, material)
mesh.position.copy(point)
grid.add(mesh)
```

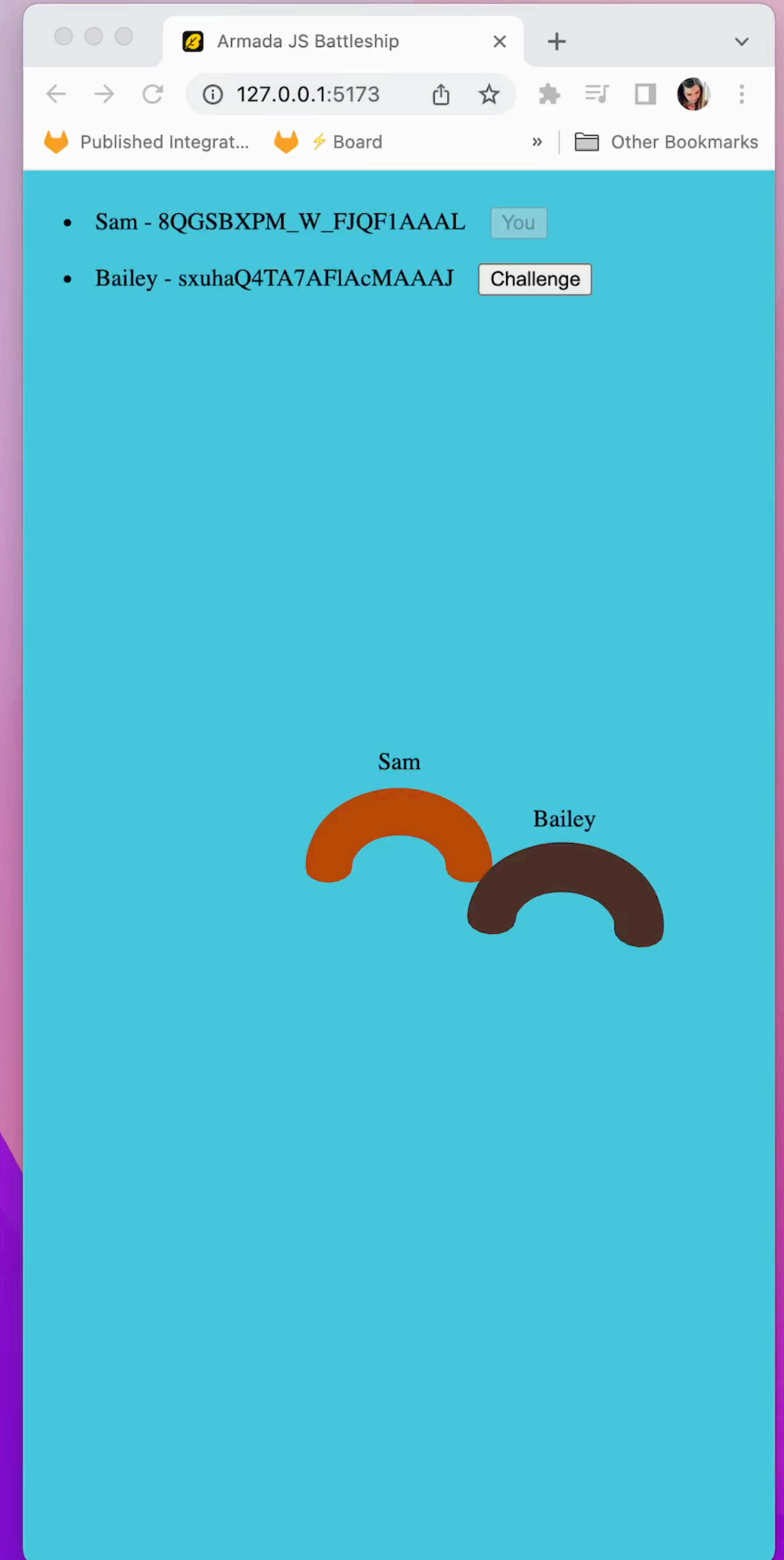
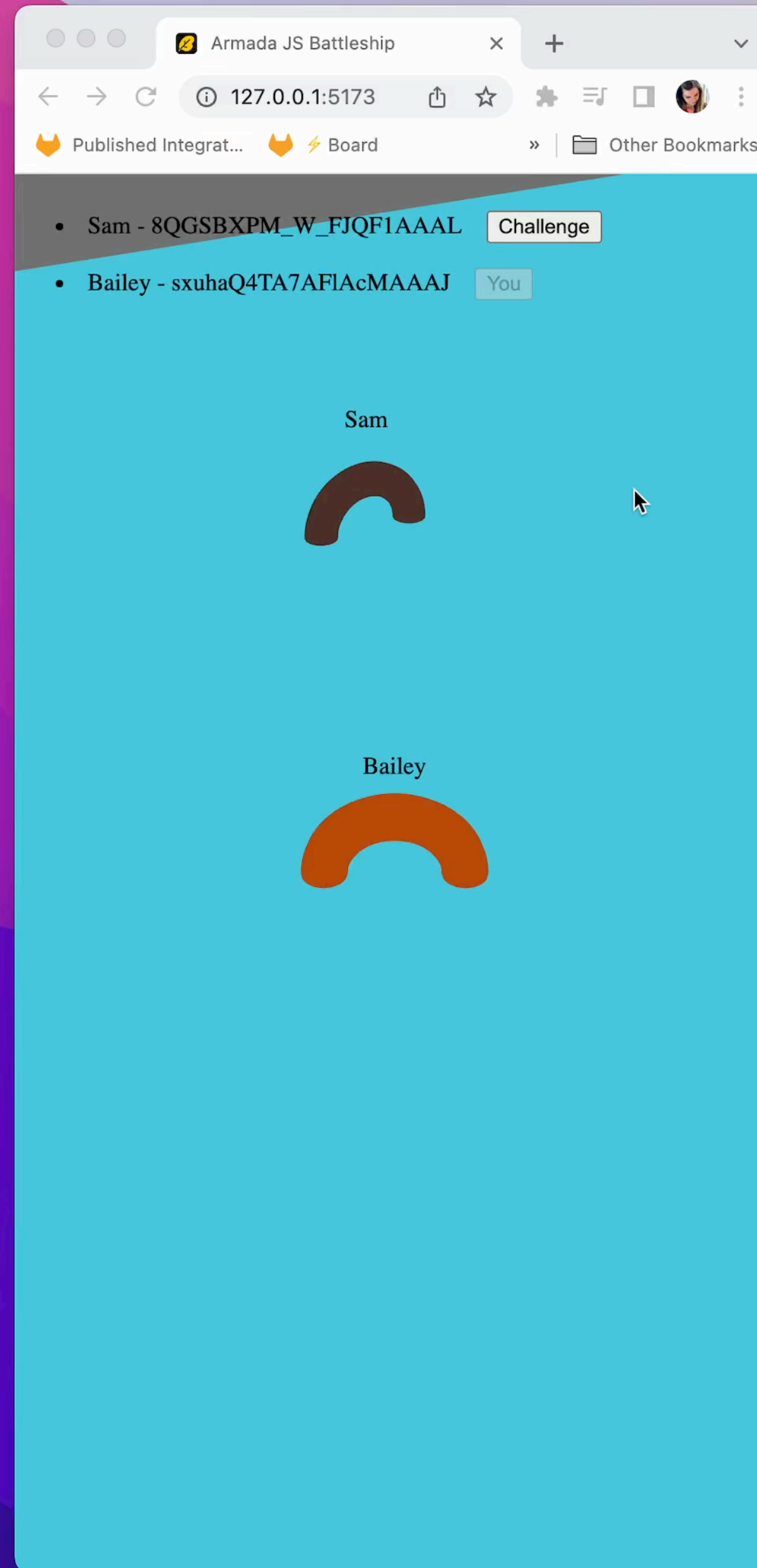
```
const point = new THREE.Vector3(
  x + this.squareWidth / 2 + this.squareWidth * 0.5,
  y + this.squareHeight / 2 + this.squareHeight * 0.5,
  grid.z
)
```

```
point.add(grid.position)
return point
}
```


Battleship Gameplay in 3D

...and all that math

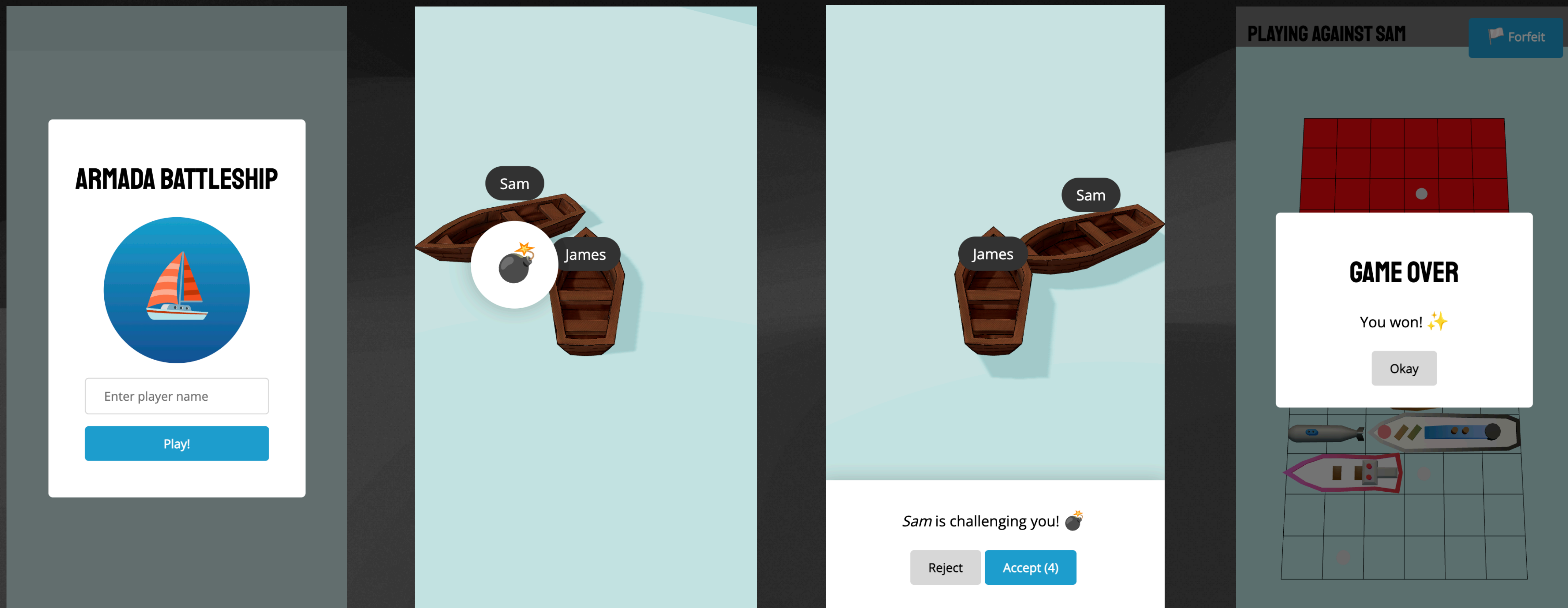
- ✓ Ships positioned correctly
- ✓ Tap on enemy grid
- ✓ Result correctly displayed on grids
- 🤪 Looks terrible





Final Touches

Styling the UI



Models

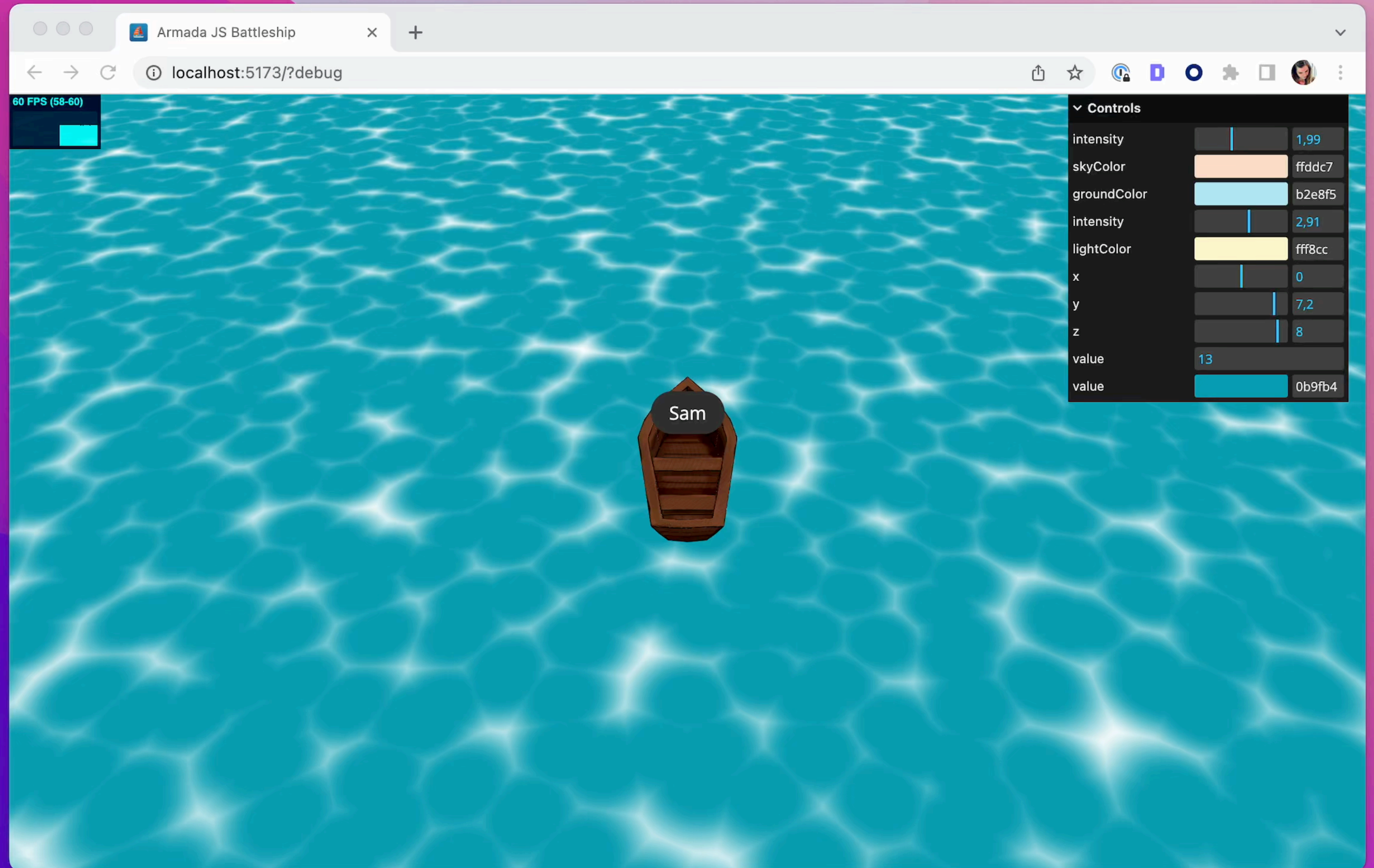


Shaders and post-processing



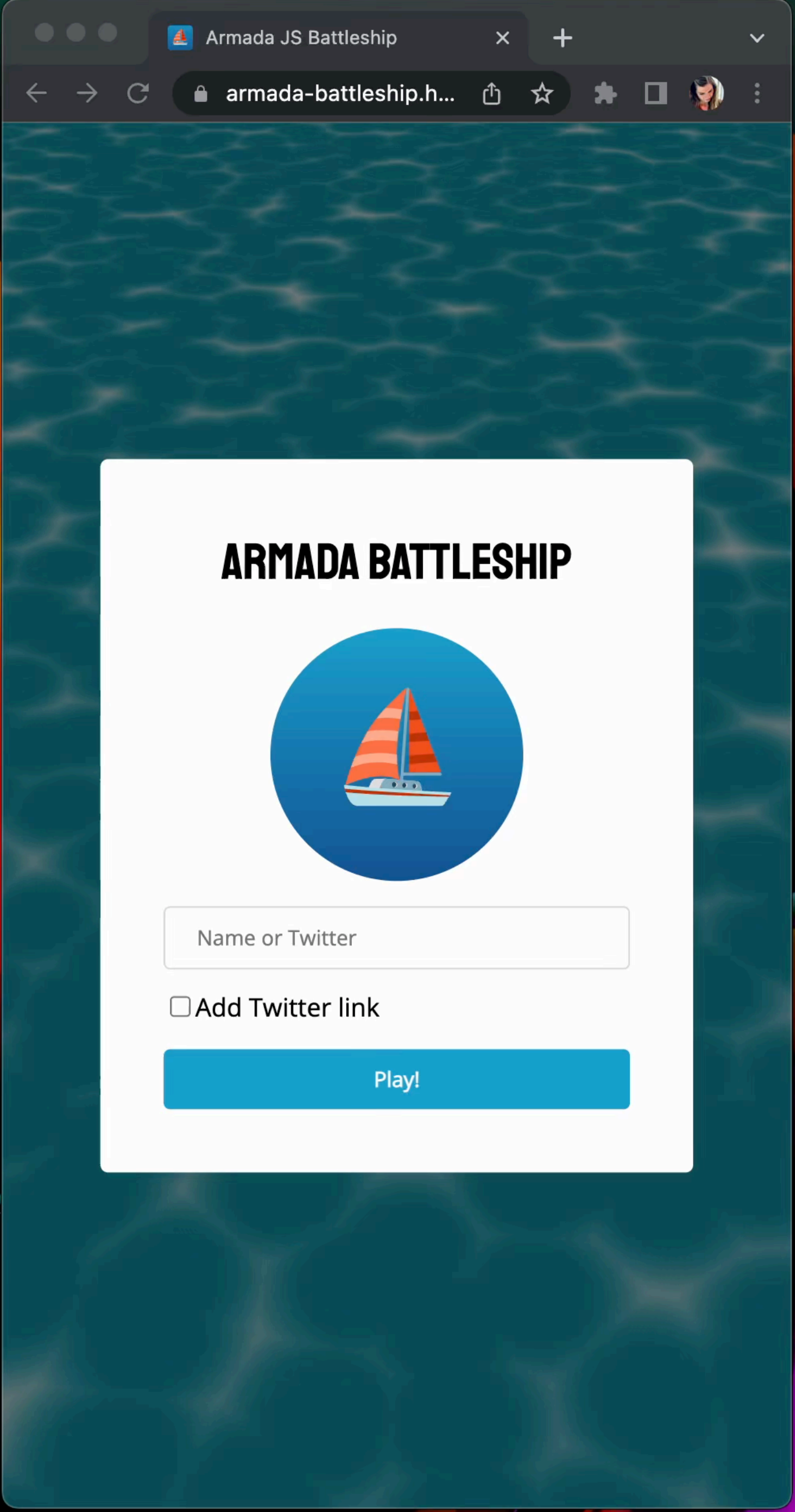
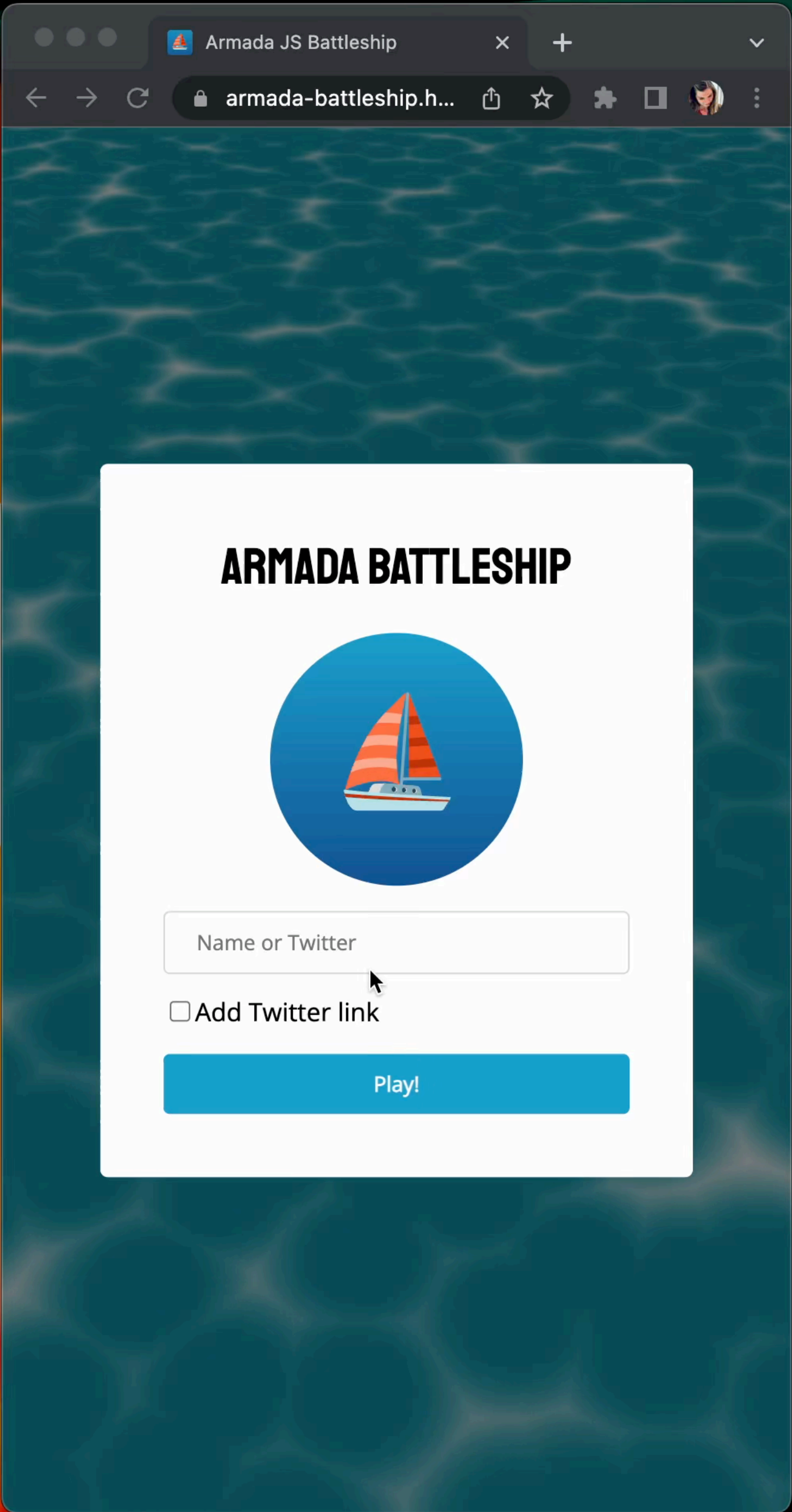
Tweaking the Three.js scene

- Colors
- Positions
- Lights
- ✨ Anything at all





Demo!





@maya_ndijk



@mayacoda

bit.ly/armada-game