Multiplayer Games with React Three Fiber and WebSockets

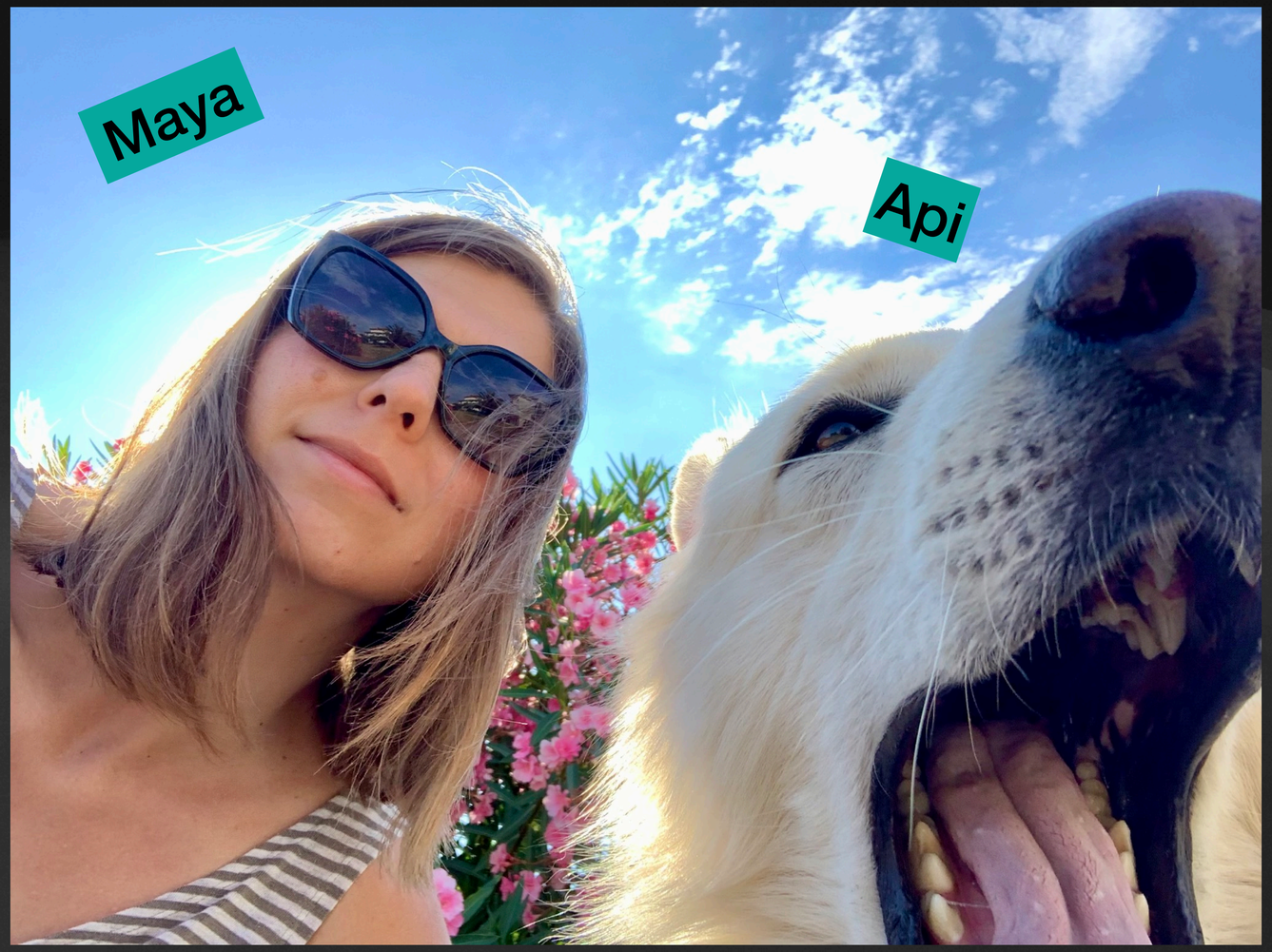
Maya Nedeljković Batić @ React Day Berlin • Dec 8, 2023

Maya Nedeljković Batić

- Art school dropout
- Software engineer at Linear
- Game development academic

@maya_ndljk

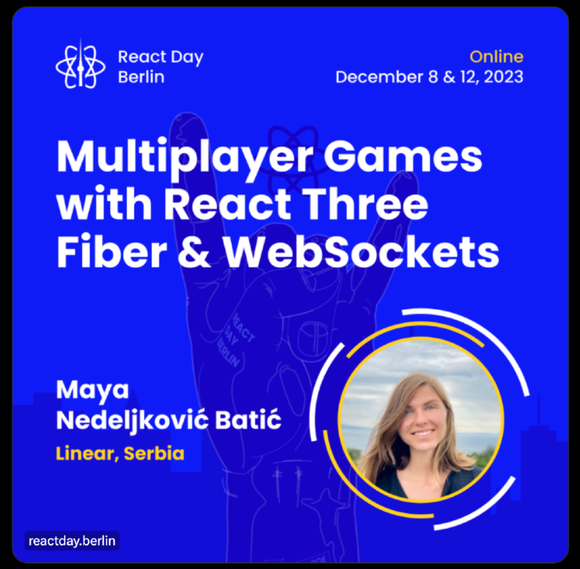
@mayacoda



Post

React Day Berlin 📅 **Dec. 8 & 12**
@reactdayberlin

🎮Multiplayer games are the coolest! You'll learn with @maya_ndljk how to structure the game code, render an interactive 3D scene in the browser, and establish two-way communication between the client and server.



10:00 AM · Sep 29, 2023 · **281.5K** Views

3 4 42 21

See similar posts

Post your reply Reply

∞ **INVNTIV** ∞ @_invntiv_ · Nov 23
why in the name of all that is holy would anyone make a game in react

1 1 27

Post

React Day Berlin 📅 Dec. 8 & 12
@reactdayberlin

🎮 Multiplayer games are the coolest! You'll learn with @maya_ndljk how to structure the game code, render an interactive 3D scene in the browser, and establish two-way communication between the client and server.

10:00 AM · Sep 29, 2023 · **281.5K** Views

3 4 42 21

See similar posts

Post your reply Reply

∞ INVNTIV ∞ @_invntiv_ · Nov 23
why in the name of all that is holy would anyone make a game in react

1 1 27

∞ INVNTIV ∞
@_invntiv_

why in the name of all that is holy would anyone make a game in react

6:44 AM · Nov 23, 2023 · **25** Views

Accessibility

Accessibility

- ✦ Accessible for you, as the developer

Accessibility

- Accessible for you, as the developer
- Accessible for your players

React Battleship

github.com/mayacoda/react-battleship

Game Plan 💪

Game Plan 💪

Game Plan 💪

Design ➡️

Game Plan 💪

Design ➡️ Prototype ➡️

Game Plan 💪

Design ➡️ Prototype ➡️ Blockout ➡️

Game Plan 💪

Design ➡️ Prototype ➡️ Blockout ➡️ Production

🎨 Design

GDD

Game title

React Battleship

Intended game systems

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

Target age of players

Adults who attend tech 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
- No login or authentication required — low barrier to entry
- Connects conference attendees in a shared experience

https://en.wikipedia.org/wiki/Game_design_document

GDD

📱 focus on mobile

Game title

React Battleship

Intended game systems

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

Target age of players

Adults who attend tech 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
- No login or authentication required — low barrier to entry
- Connects conference attendees in a shared experience

https://en.wikipedia.org/wiki/Game_design_document

GDD

🤳 focus on mobile

Game title

React Battleship

Intended game systems

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

Target age of players

Adults who attend tech 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
- No login or authentication required — low barrier to entry
- Connects conference attendees in a shared experience

👦🏽 12yo nephew doesn't have to like it

https://en.wikipedia.org/wiki/Game_design_document

GDD

📱 focus on mobile

Game title

React Battleship

Intended game systems

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

Target age of players

Adults who attend tech 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
- No login or authentication required — low barrier to entry
- Connects conference attendees in a shared experience

👦🏽 12yo nephew doesn't have to like it

⛵ how to set up scenes

https://en.wikipedia.org/wiki/Game_design_document

GDD

📱 focus on mobile

Game title

React Battleship

Intended game systems

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

Target age of players

Adults who attend tech 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
- No login or authentication required — low barrier to entry
- Connects conference attendees in a shared experience

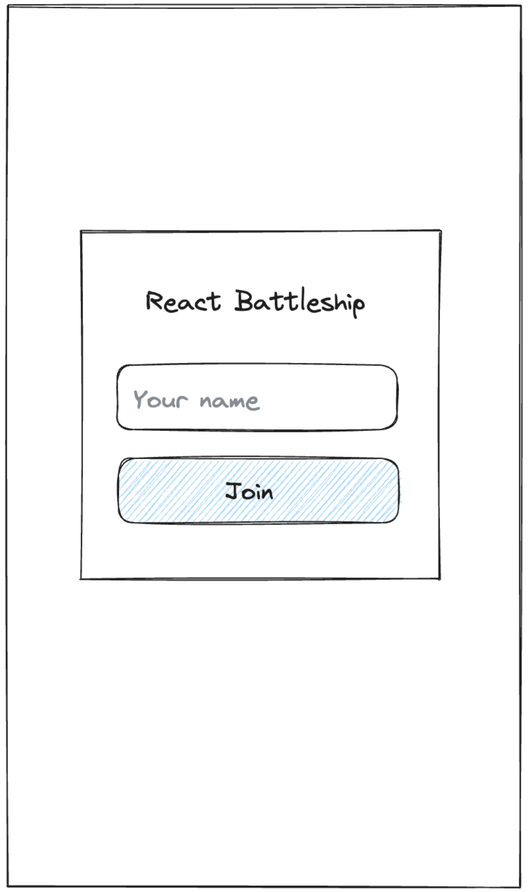
👦🏽 12yo nephew doesn't have to like it

⛵ how to set up scenes

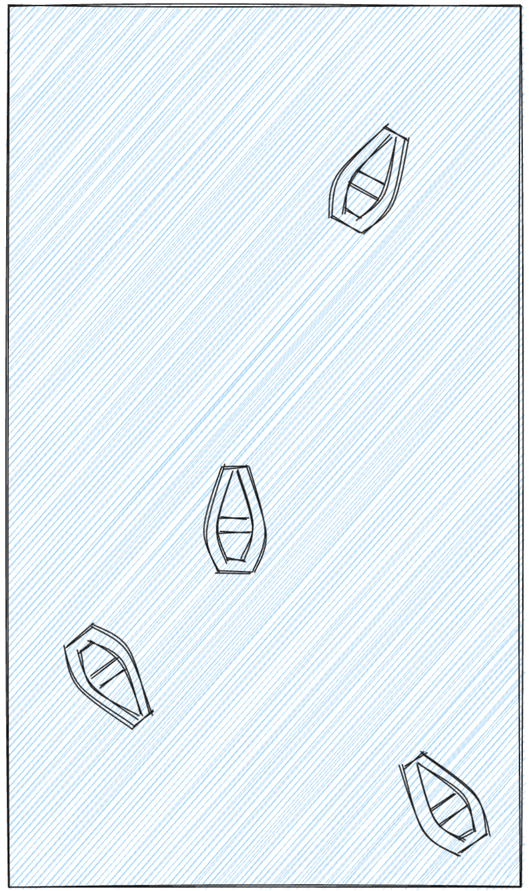
🏆 must haves

https://en.wikipedia.org/wiki/Game_design_document

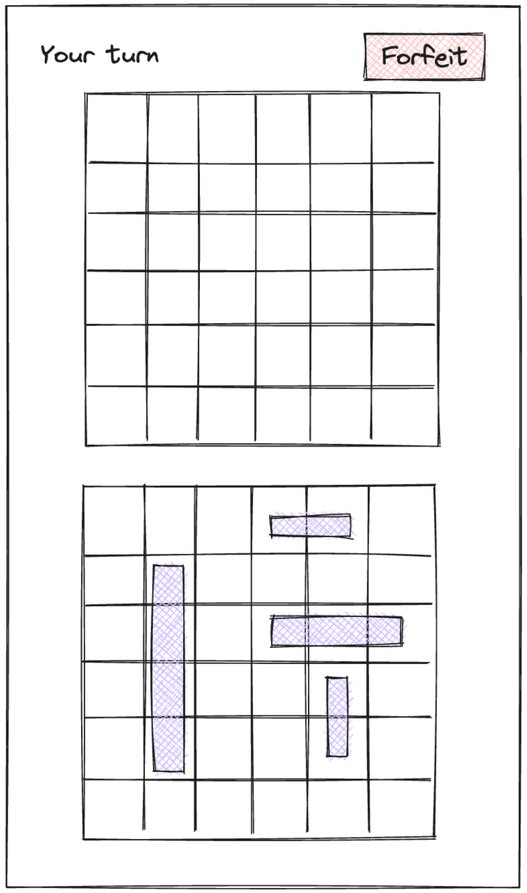
Join page



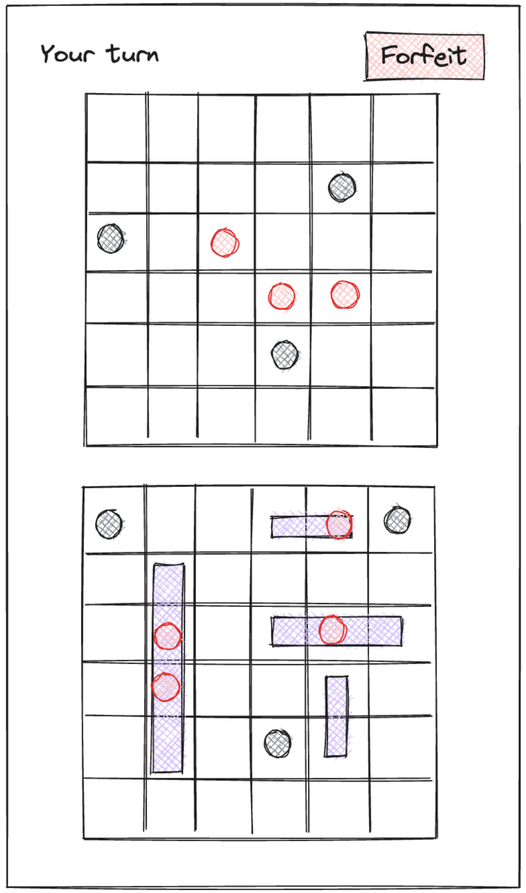
Lobby page



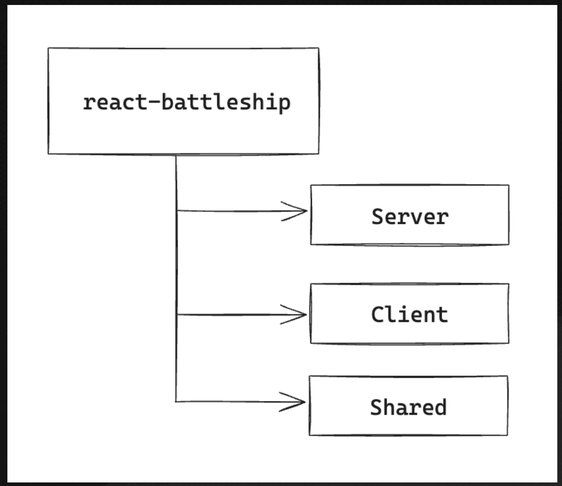
Game page

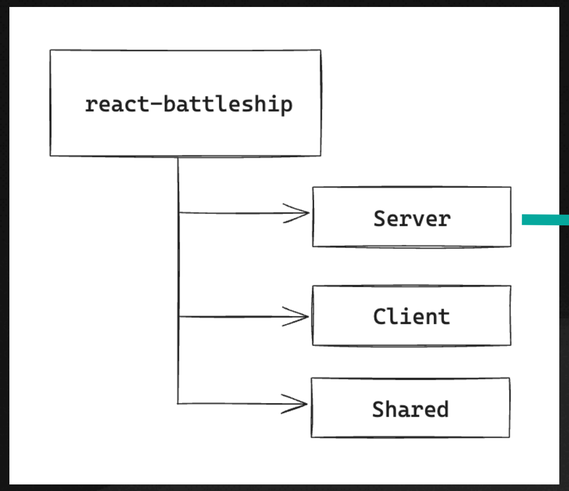


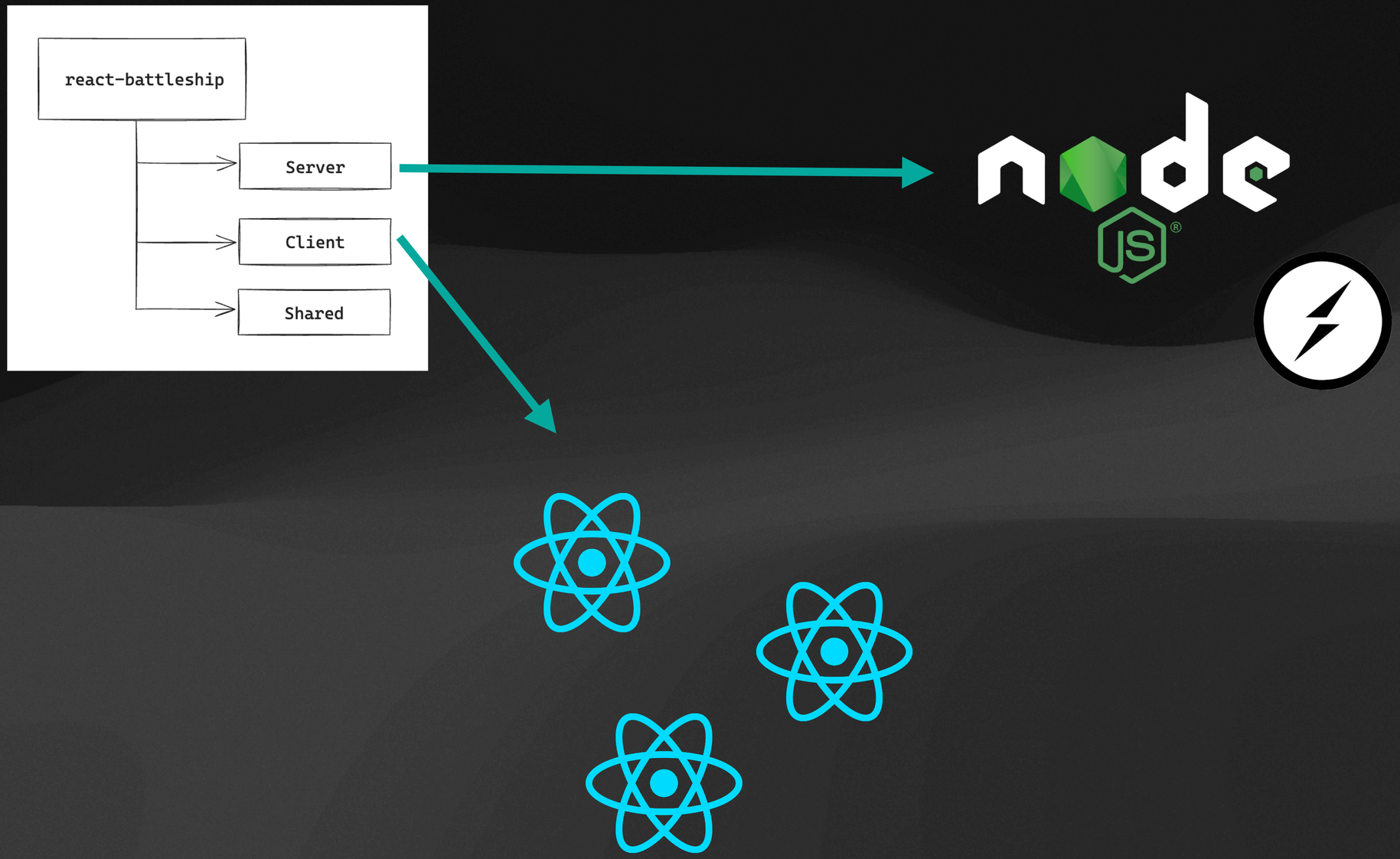
Game page

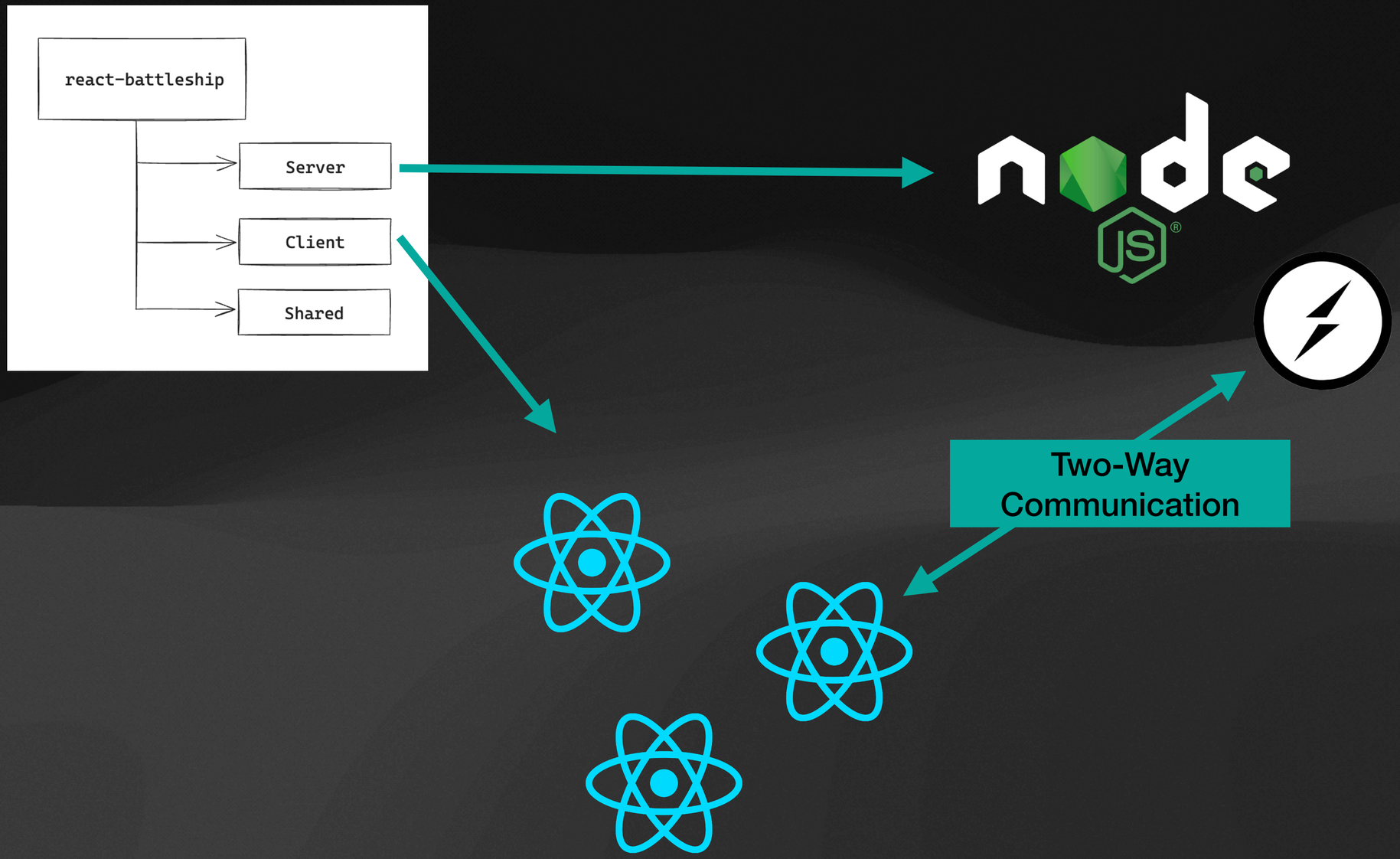


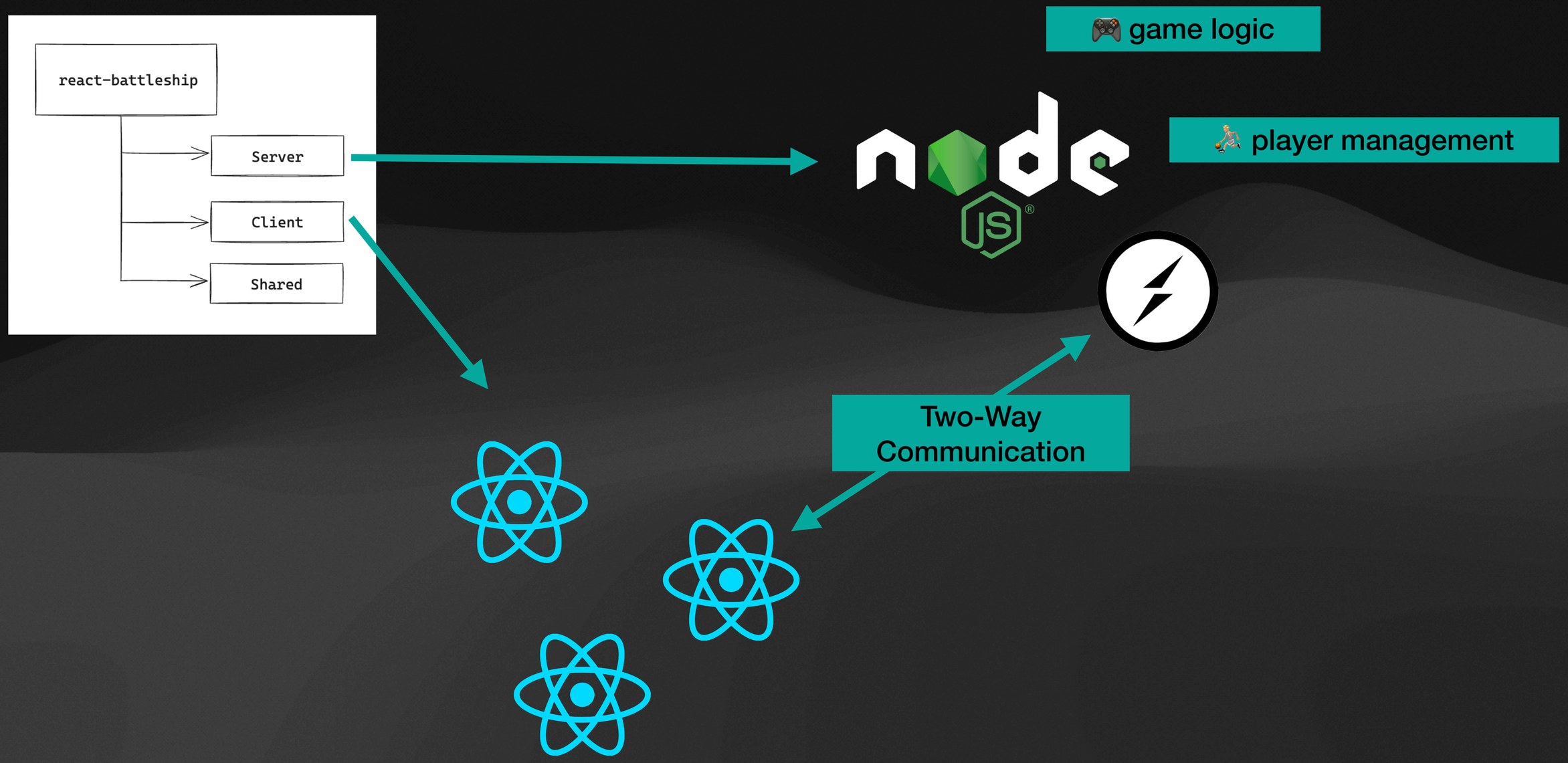
Architecture

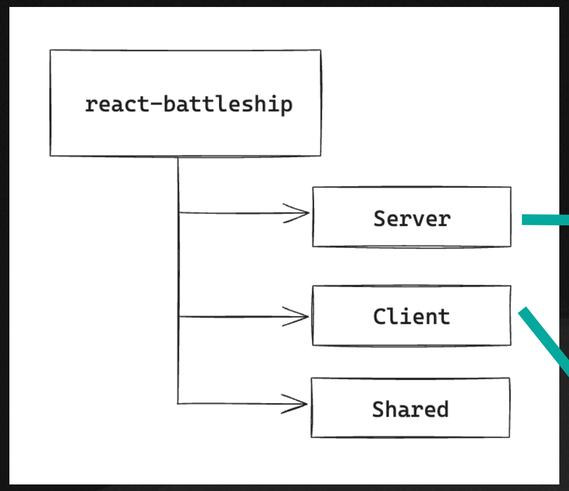

react-battleship
Server
Client
Shared


react-battleship
Server
Client
Shared


react-battleship
Server
Client
Shared


react-battleship
Server
Client
Shared
node
JS
Two-Way
Communication


react-battleship
Server
Client
Shared
🎮 game logic
🕺 player management
Two-Way
Communication



🎮 game logic

🕺 player management

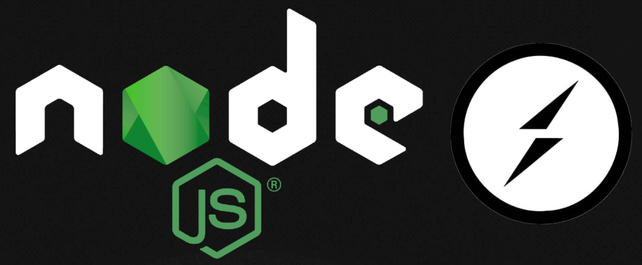
Two-Way Communication

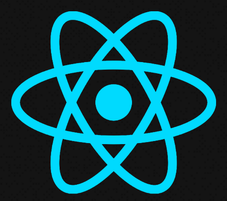
⛵ display game state

🕹️ player input

⚠️ **Disclaimer: implementation details here**
github.com/mayacoda/react-battleship

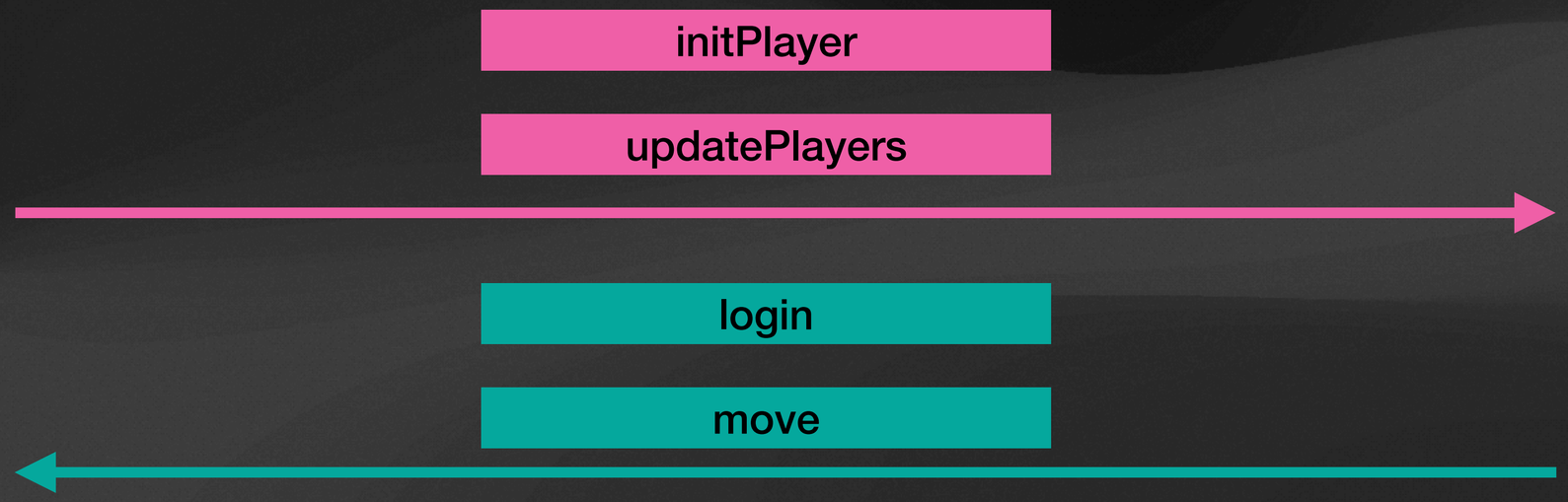
Prototype

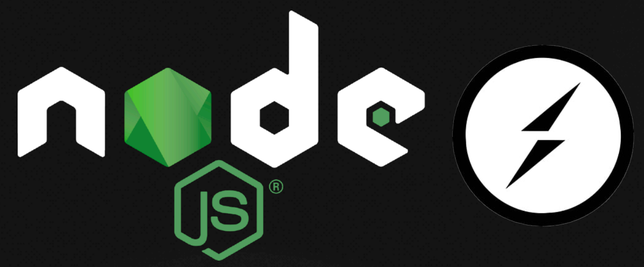

node
JS

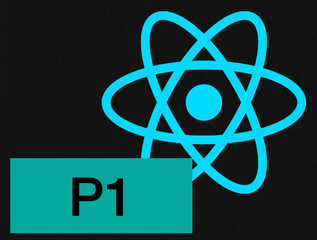


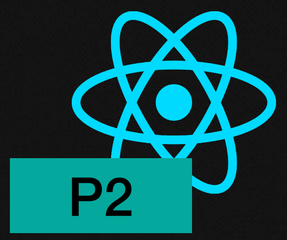
initPlayer

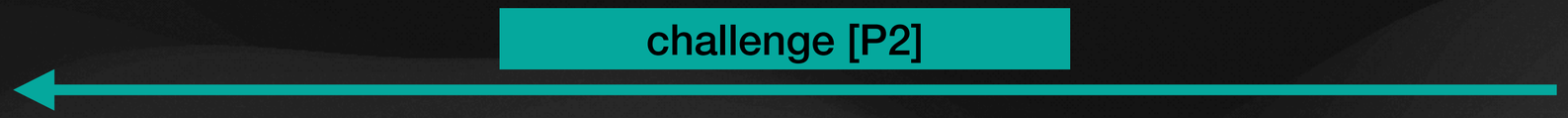
updatePlayers

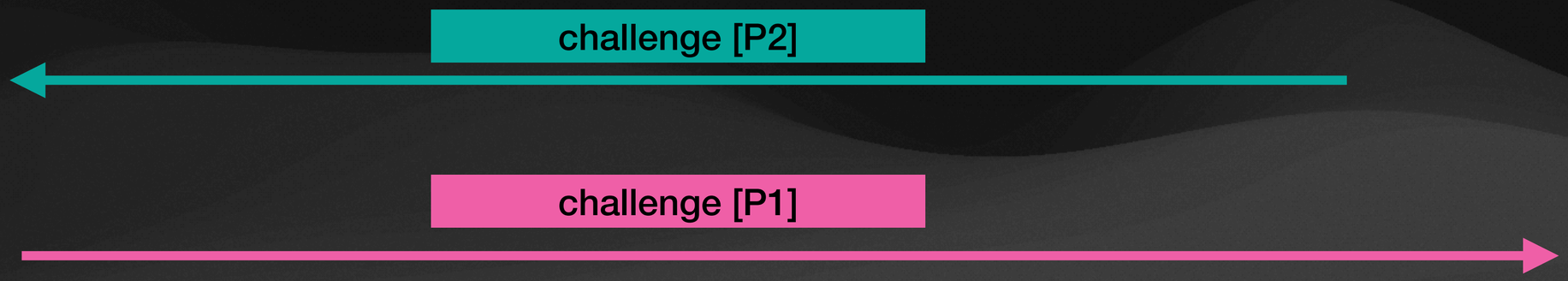

initPlayer
updatePlayers
login
move

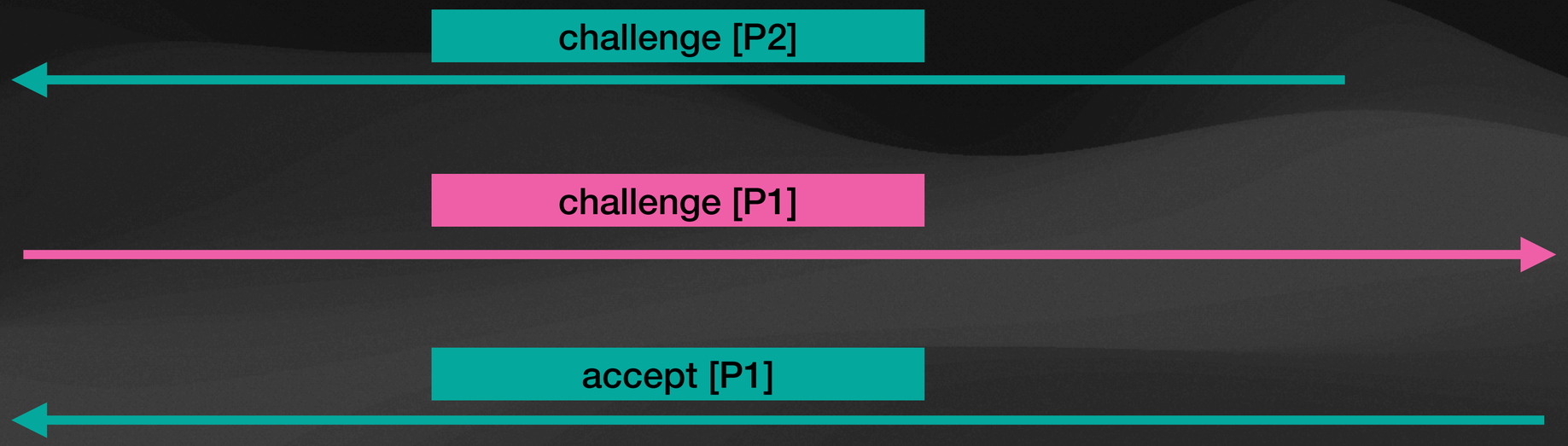

node
JS

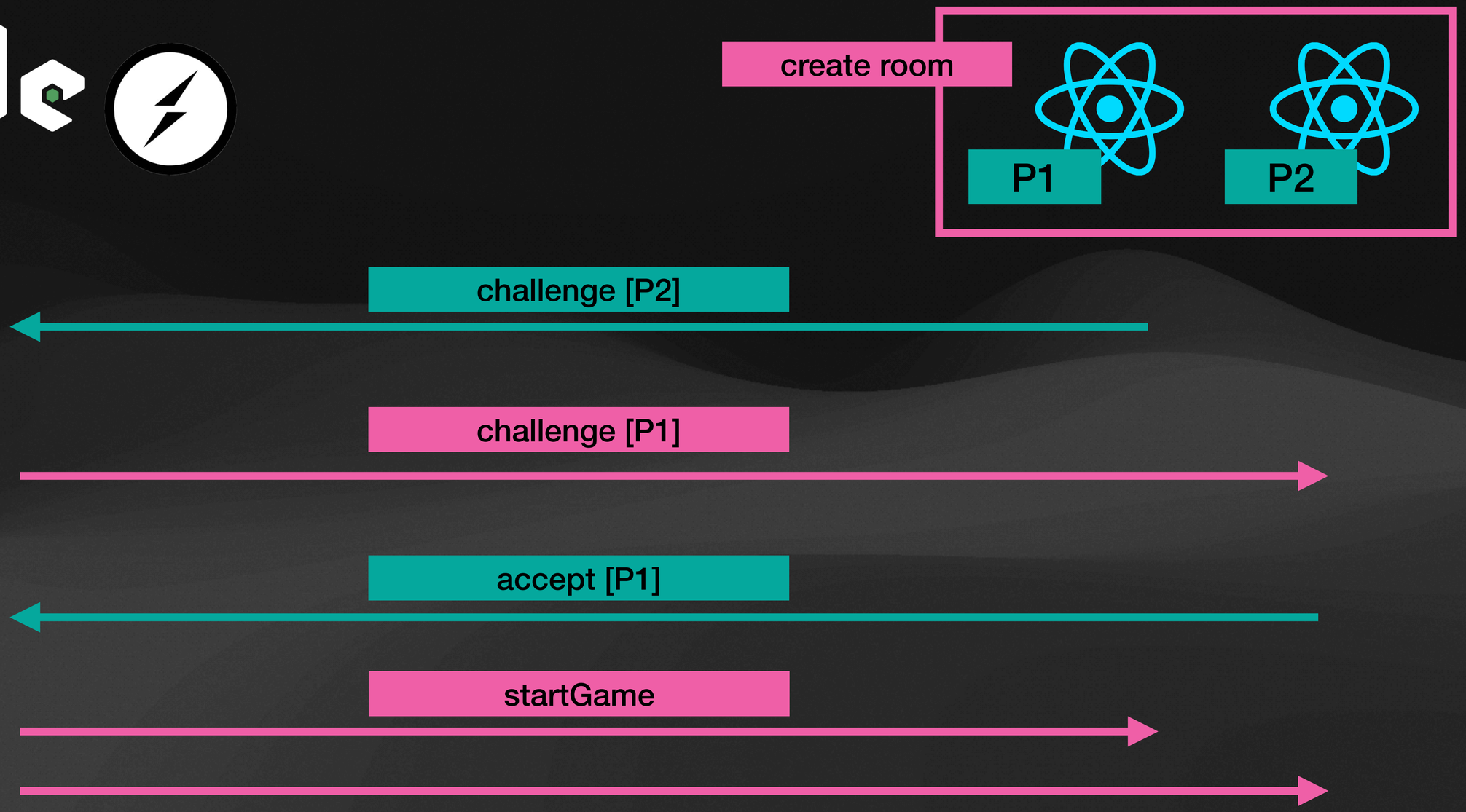

P1


P2


P1
P2
challenge [P2]

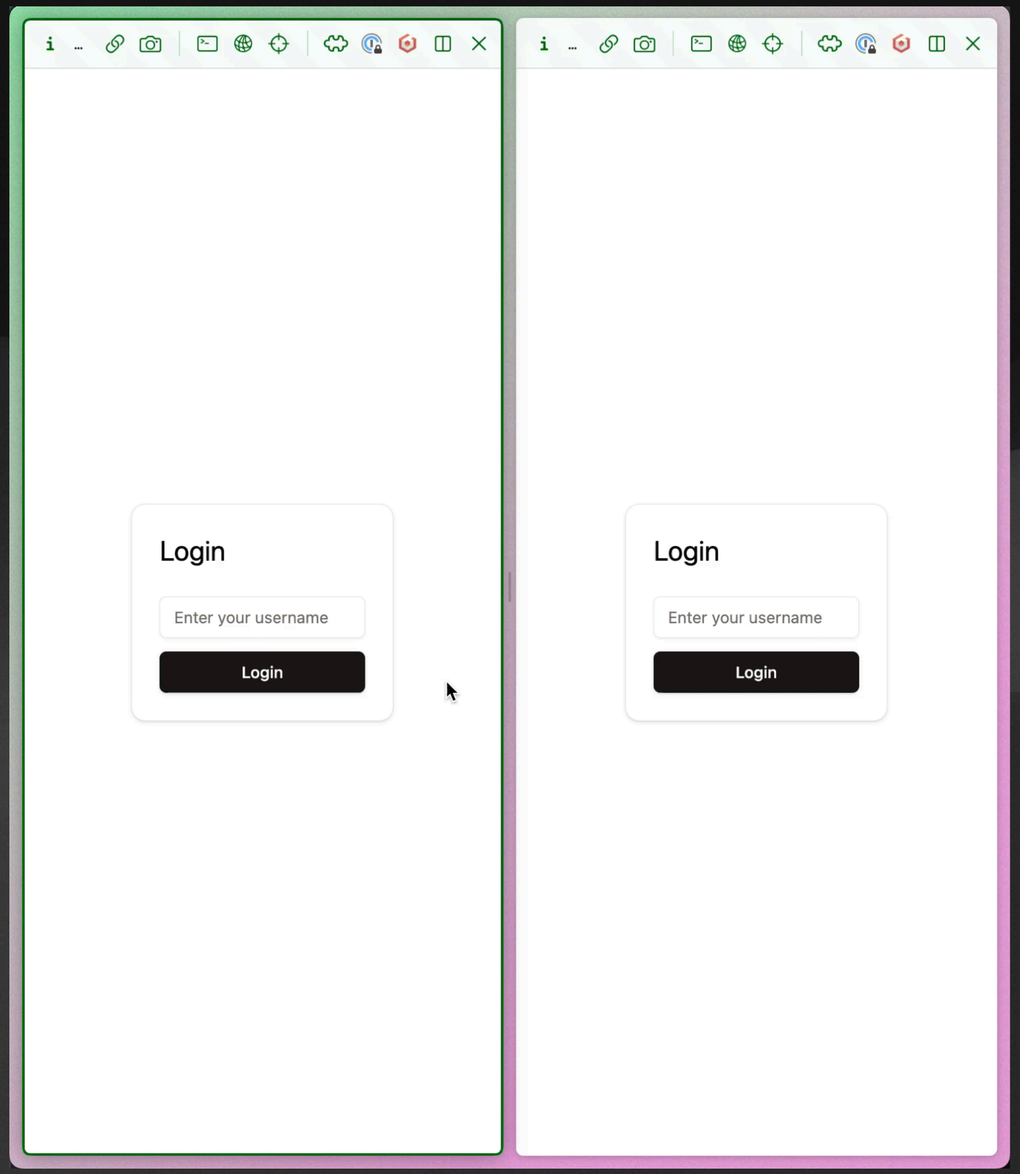

P1
P2
challenge [P2]
challenge [P1]

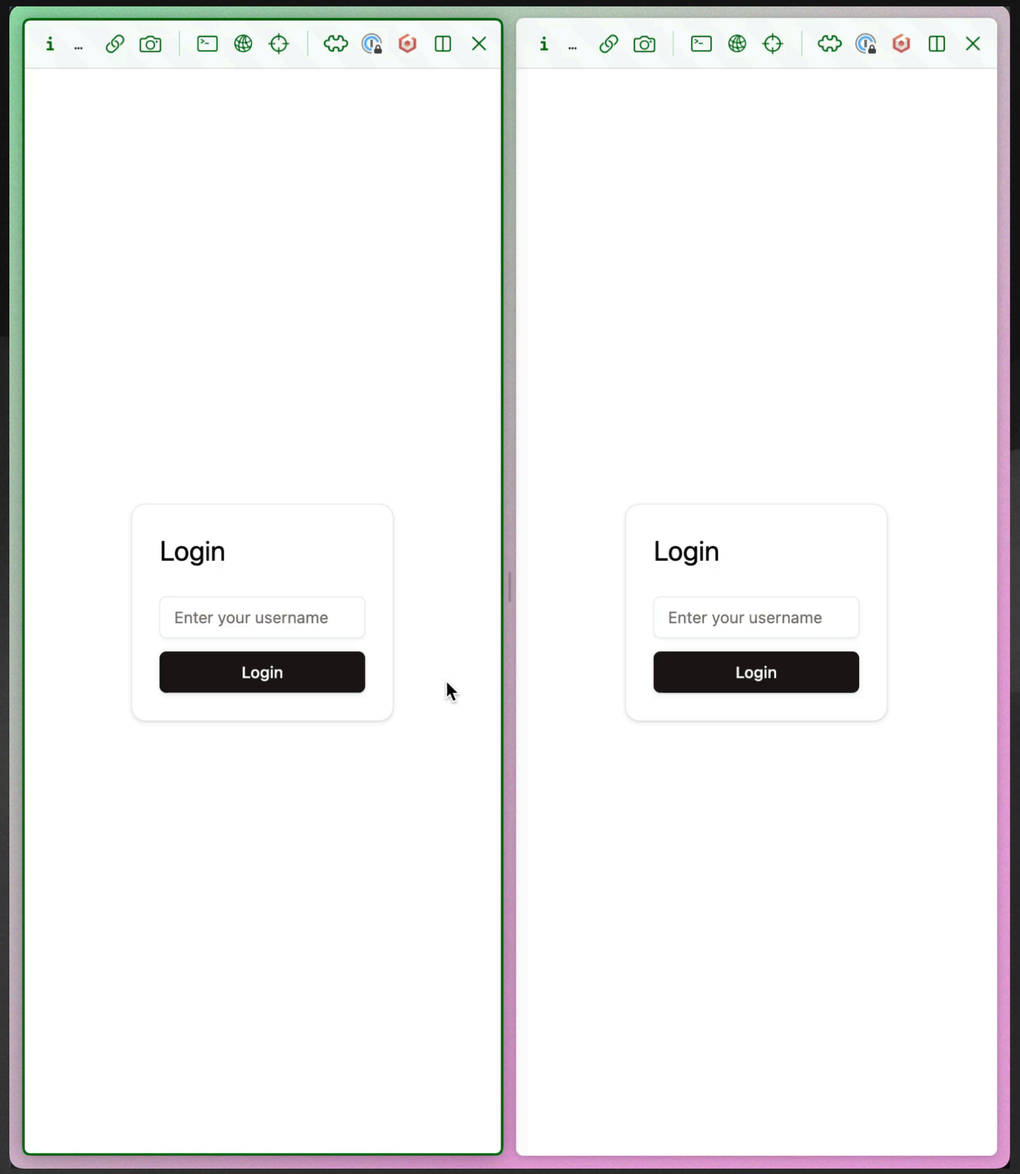

P1
P2
challenge [P2]
challenge [P1]
accept [P1]


create room
P1
P2
challenge [P2]
challenge [P1]
accept [P1]
startGame

Synchronizing server state with client state with React Context

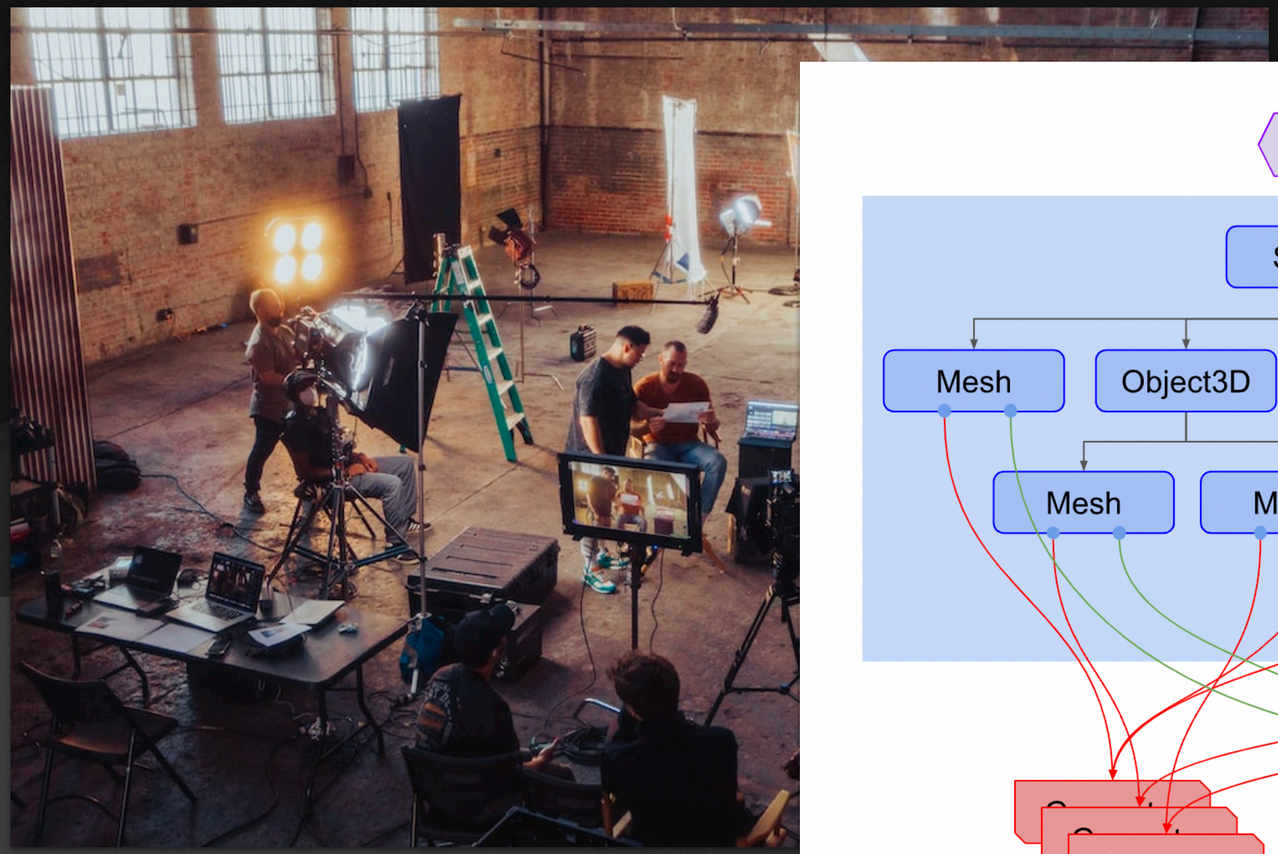
```
export const GameProvider = ({ children }: PropsWithChildren) => {
  const [players, setPlayers] = useState<Record<string, Player>>({});
  const [currentPlayer, setCurrentPlayer] = useState<Player | null>(null);
  const [challenges, setChallenges] = useState<Player[]>([]);
  const [onGameFinished, setOnGameFinished] = useState<() => void>(() => {});
  const [socket, setSocket] = useState<TypedClient | null>(null);
  const [gameState, setGameState] =
    useState<GameContextType["gameState"]>(null);
  const [previousGames, setPreviousGames] = useState<EndState[]>([]);
  const [isConnected, setIsConnected] = useState(false);
  const navigate = useNavigate();

  useEffect(() => {
    // Establish WebSocket connection when the component mounts
    const newSocket: TypedClient = io(SOCKET_URL, {
      transports: ["websocket"],
    });
    setSocket(newSocket);
    let localPlayers: Record<string, Player> = {};
    let localPlayer: Player | null = null;

    newSocket.on("disconnect", () => {
      setIsConnected(false);
      setGameState(null);
      setPlayers({});
      setCurrentPlayer(null);
      setChallenges([]);
      setPreviousGames([]);
      navigate("/");
    });
    newSocket.on("initPlayer", (p: Player) => {
      setIsConnected(true);
      localPlayer = p;
      setCurrentPlayer(localPlayer);
```

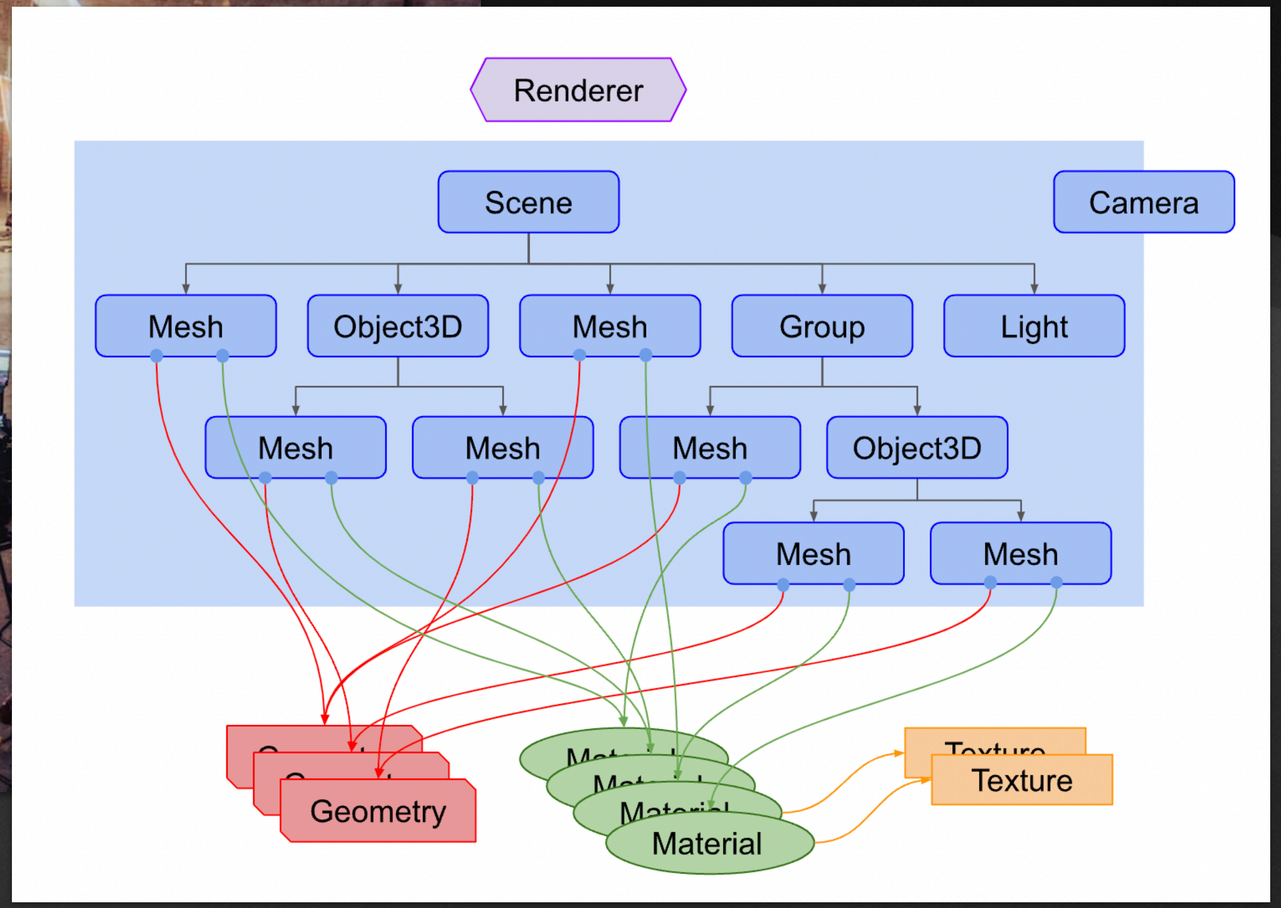

Login
Enter your username
Login
Login
Enter your username
Login


Login
Enter your username
Login
Login
Enter your username
Login

📦 Blockout

https://threejs.org/manual/#en/fundamentals





https://threejs.org/manual/#en/fundamentals

Three.js

```
const scene = new THREE.Scene()
const camera = new THREE.PerspectiveCamera(75, width / height, 0.1, 1000)

const renderer = new THREE.WebGLRenderer()
renderer.setSize(width, height)
document.querySelector('#canvas-container').appendChild(renderer.domElement)

const mesh = new THREE.Mesh()
mesh.geometry = new THREE.BoxGeometry()
mesh.material = new THREE.MeshStandardMaterial()

scene.add(mesh)

function animate() {
  requestAnimationFrame(animate)
  renderer.render(scene, camera)
}

animate()
```

Three.js

```
const scene = new THREE.Scene()
const camera = new THREE.PerspectiveCamera(75, width / height, 0.1, 1000)

const renderer = new THREE.WebGLRenderer()
renderer.setSize(width, height)
document.querySelector('#canvas-container').appendChild(renderer.domElement)

const mesh = new THREE.Mesh()
mesh.geometry = new THREE.BoxGeometry()
mesh.material = new THREE.MeshStandardMaterial()

scene.add(mesh)

function animate() {
  requestAnimationFrame(animate)
  renderer.render(scene, camera)
}

animate()
```

R3F.jsx

```
function App() {
  return (
    <div id="canvas-container">
      <Canvas>
        <mesh>
          <boxGeometry />
          <meshStandardMaterial />
        </mesh>
      </Canvas>
    </div>
  )
}
```

ReactLobby.jsx

```jsx
<div className="flex flex-col h-screen">
  <Table>
    <TableHeader>
      <TableRow>
        <TableHead>Name</TableHead>
        <TableHead>Status</TableHead>
      </TableRow>
    </TableHeader>
    <TableBody>
      {players.map((player) => (
        <TableRow key={player.id}>
          <TableCell>{player.name}</TableCell>
          <TableCell>
            {player.isPlaying ? (
              <Badge>Playing</Badge>
            ) : player.id !== game.currentPlayer?.id ? (
              <Button size="sm" onClick={() => onChallenge(player.id)}>
                Challenge
              </Button>
            ) : null}
          </TableCell>
        </TableRow>
      ))}
    </TableBody>
  </Table>
</div>
```

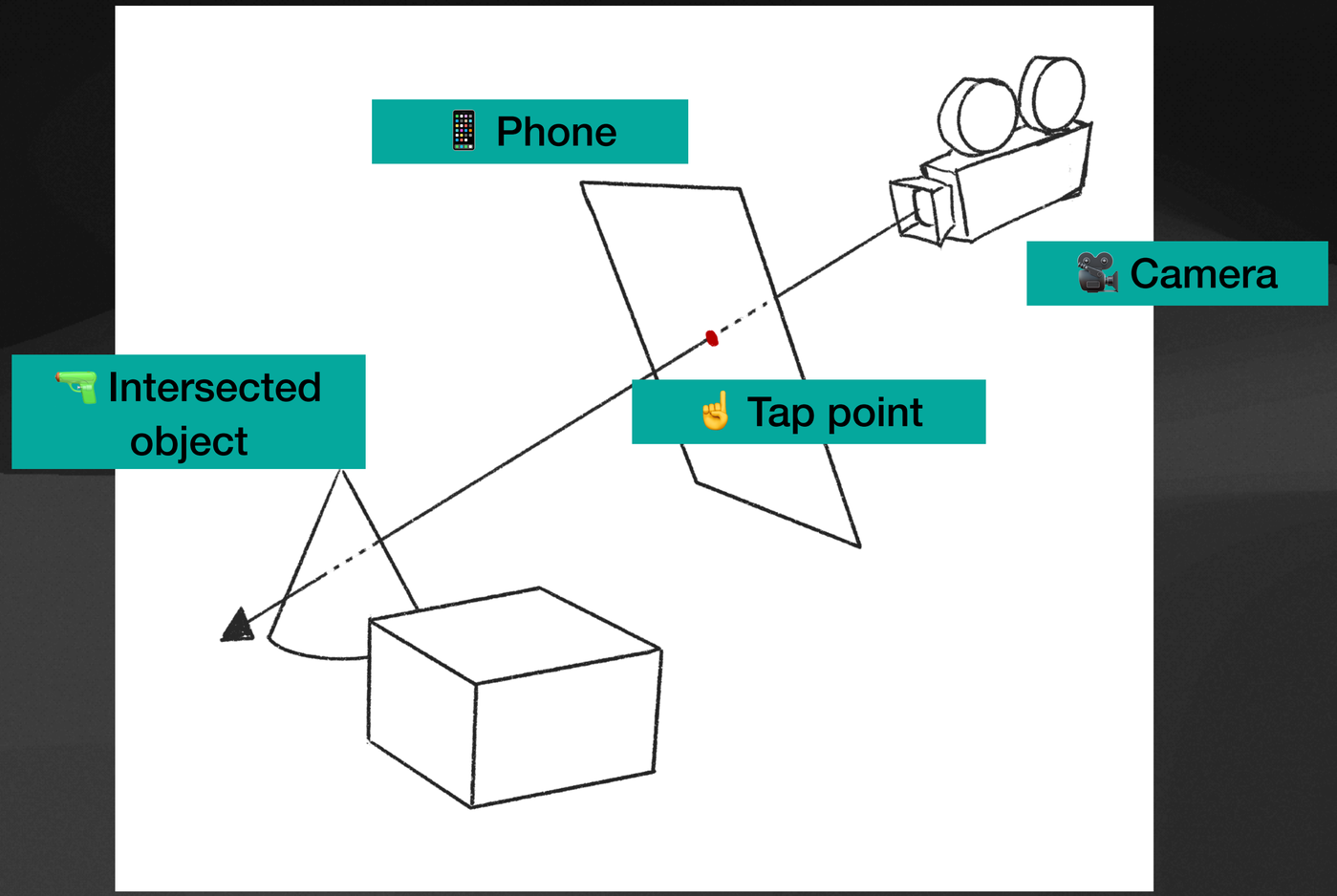
ReactLobby.jsx

```jsx
<div className="flex flex-col h-screen">
  <Table>
    <TableHeader>
      <TableRow>
        <TableHead>Name</TableHead>
        <TableHead>Status</TableHead>
      </TableRow>
    </TableHeader>
    <TableBody>
      {players.map((player) => (
        <TableRow key={player.id}>
          <TableCell>{player.name}</TableCell>
          <TableCell>
            {player.isPlaying ? (
              <Badge>Playing</Badge>
            ) : player.id !== game.currentPlayer?.id ? (
              <Button size="sm" onClick={() => onChallenge(player.id)}>
                Challenge
              </Button>
            ) : null}
          </TableCell>
        </TableRow>
      ))}
    </TableBody>
  </Table>
</div>
```

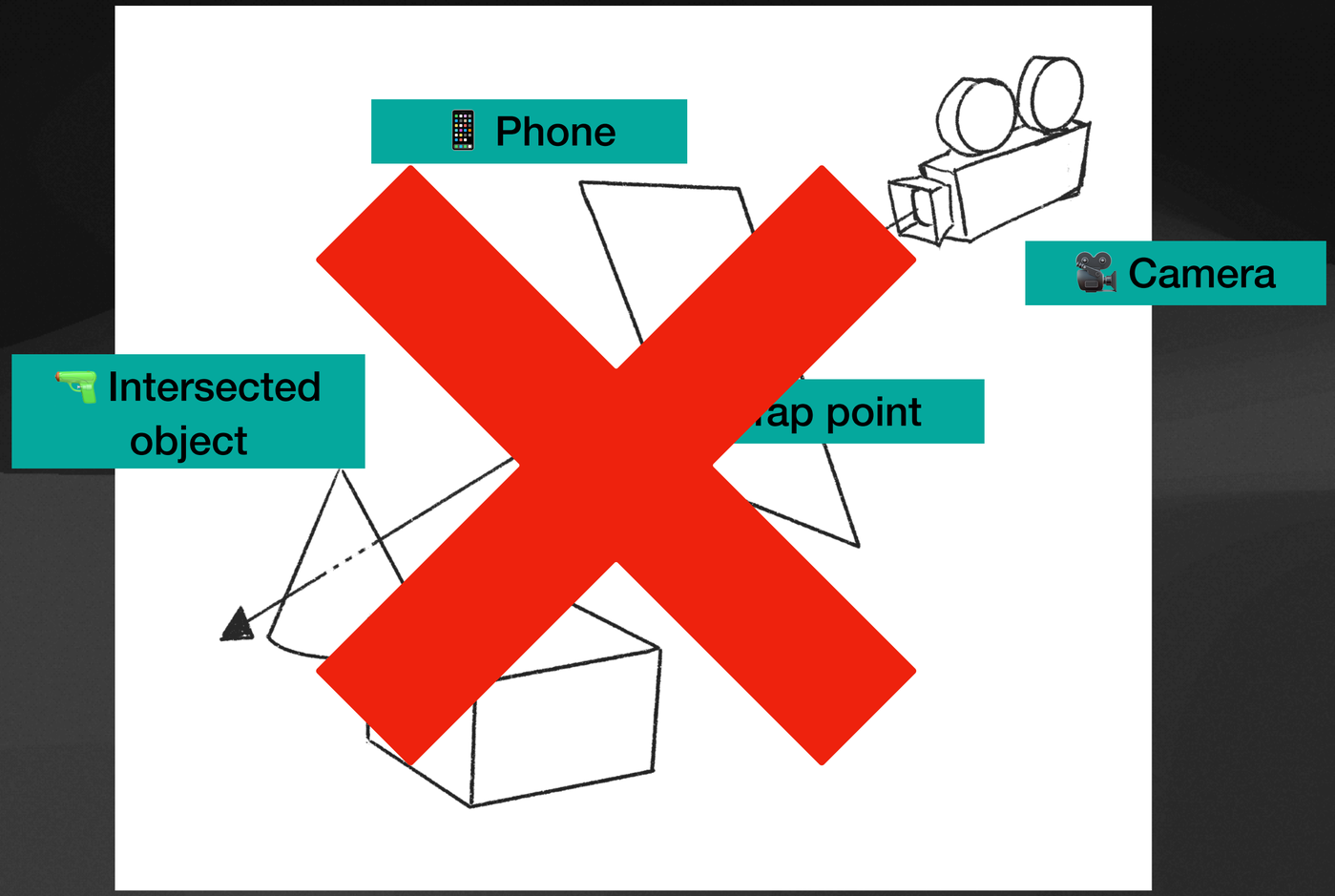
R3fLobby.jsx

```jsx
<>
  <ambientLight intensity={1} />
  <directionalLight position={[10, 10, 10]} intensity={3.5} castShadow />
  <OrbitControls />
  <WaterPlane
    ref={planeRef}
    size={100}
    rotation={[-Math.PI / 2, 0, 0]}
    position={[0, -0.1, 0]}
    onClick={handlePlaneClick}
  />
  <ControlledPlayer player={currentPlayer!} ref={playerRef} />
  {players.map((player) => (
    <OtherPlayer
      key={player.id}
      player={player}
      onChallenge={onChallenge}
    />
  ))}
</>
```

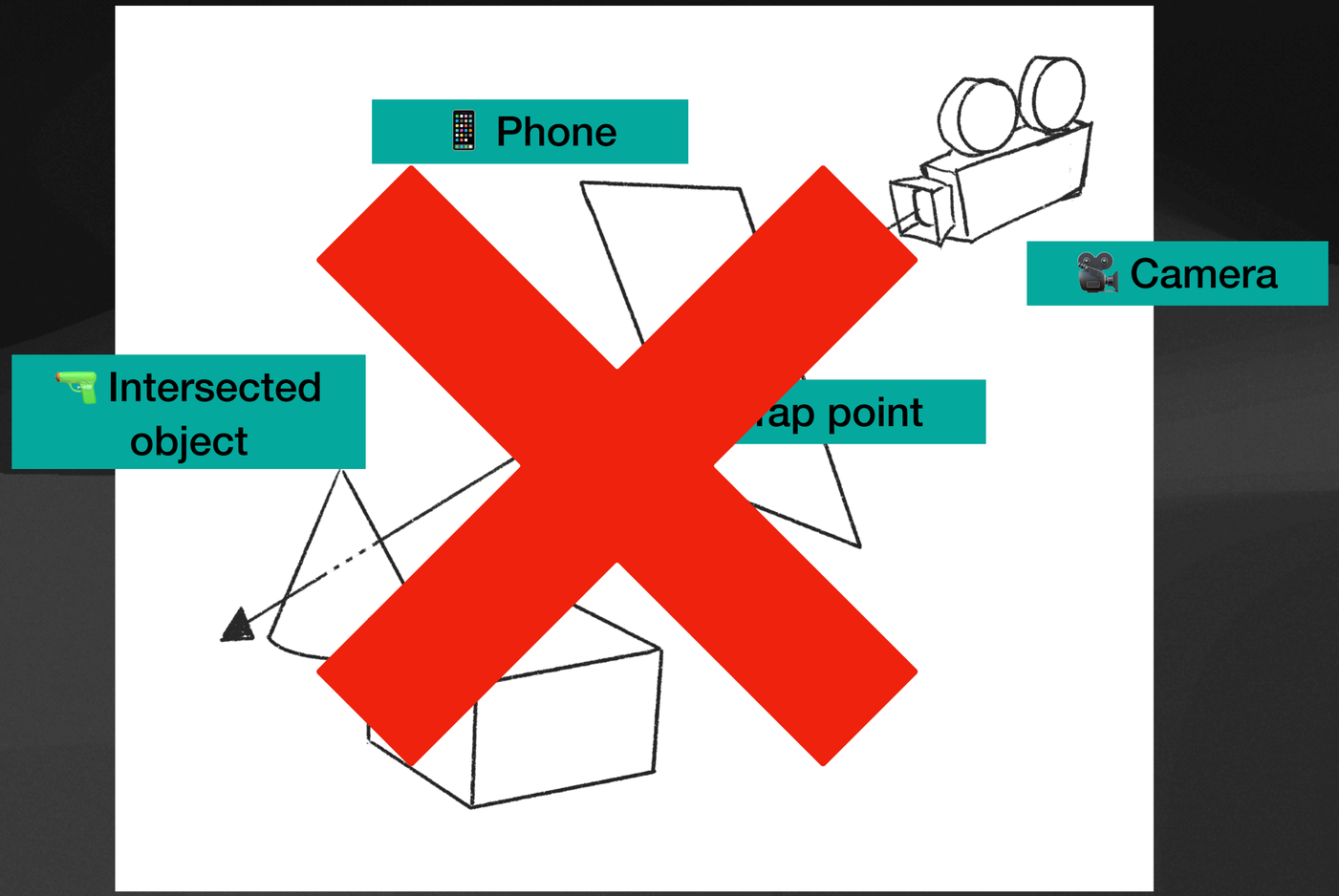
Raycasting or Clicking?



Raycasting or Clicking?

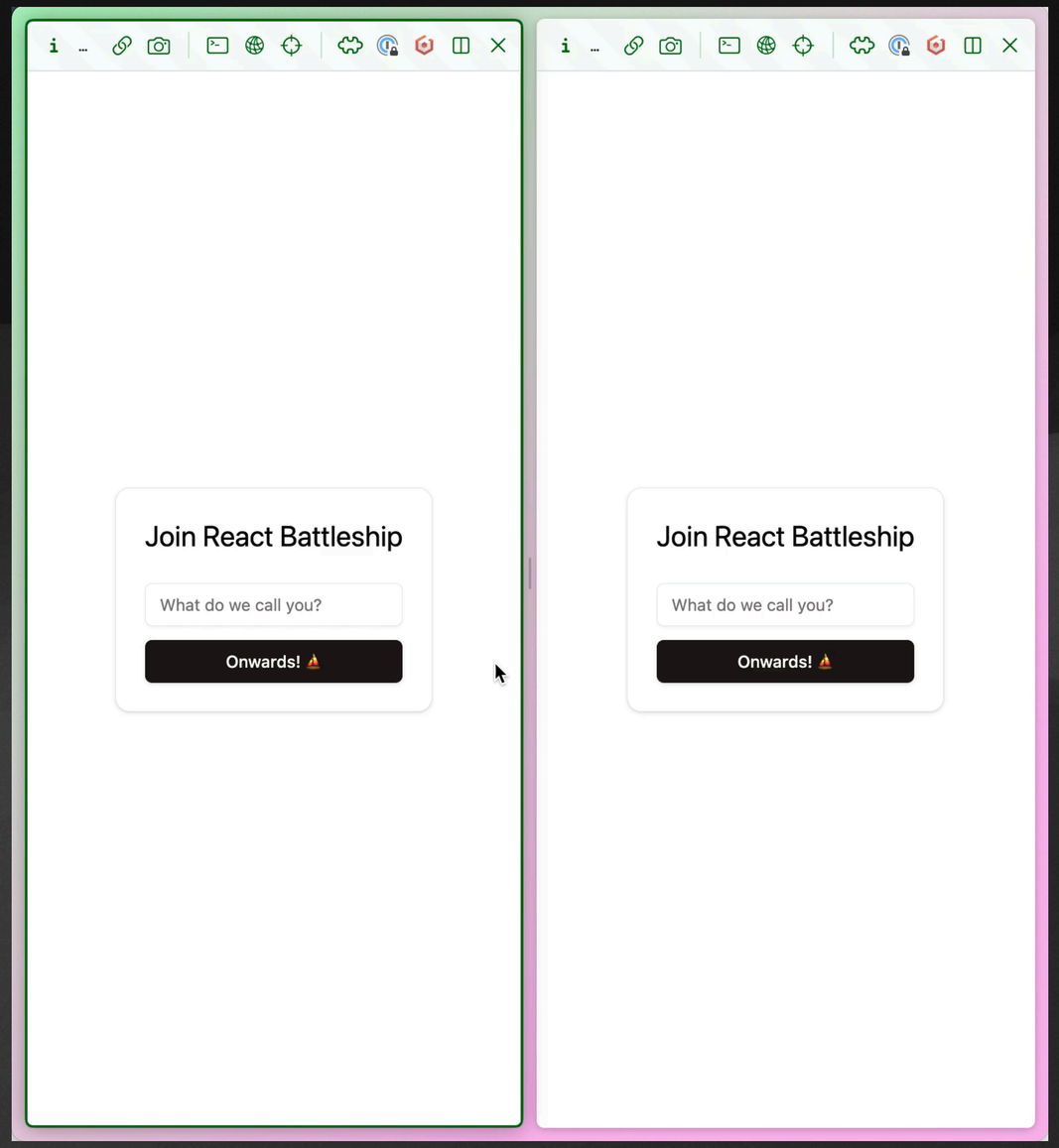


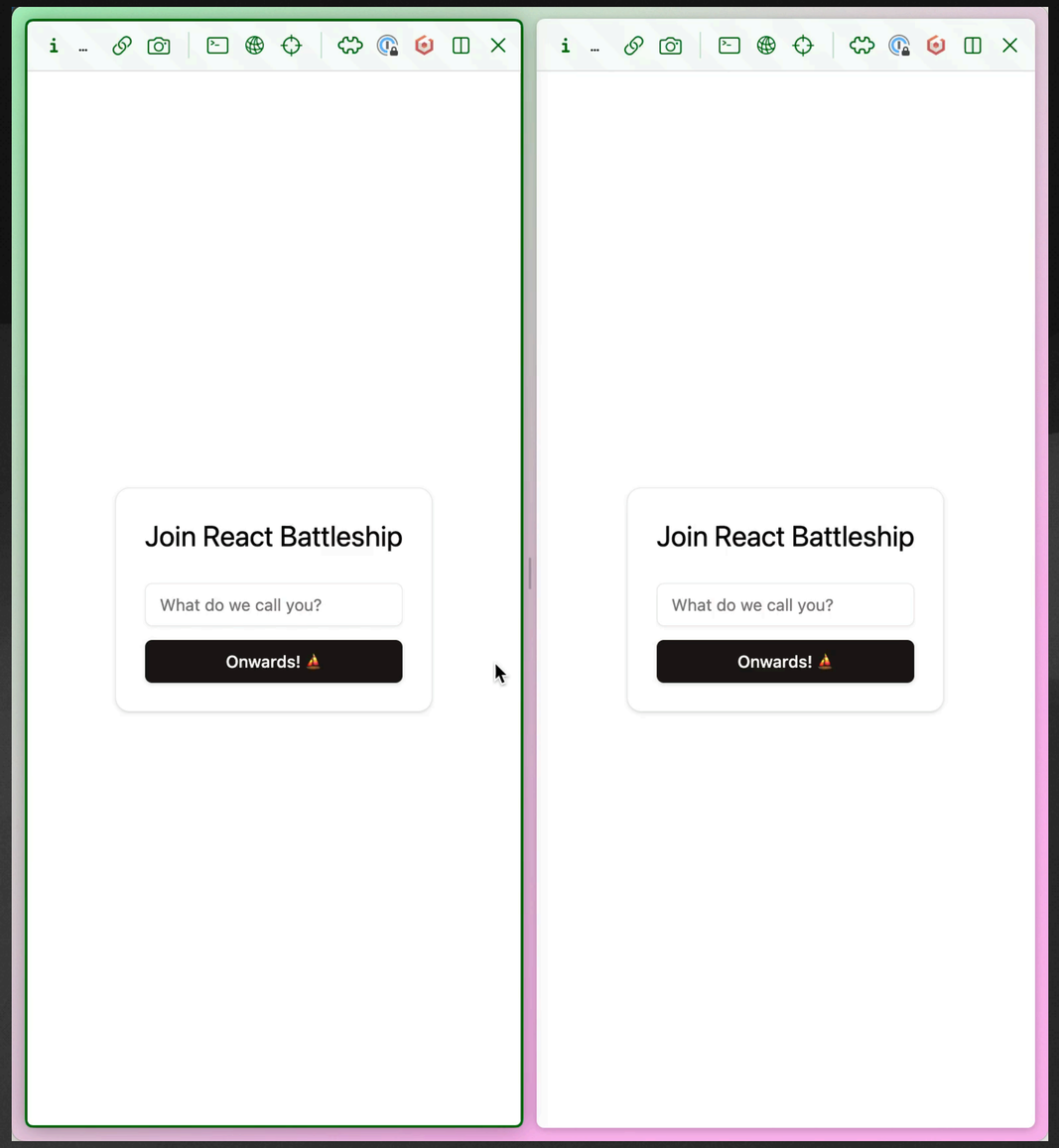
Raycasting or Clicking?

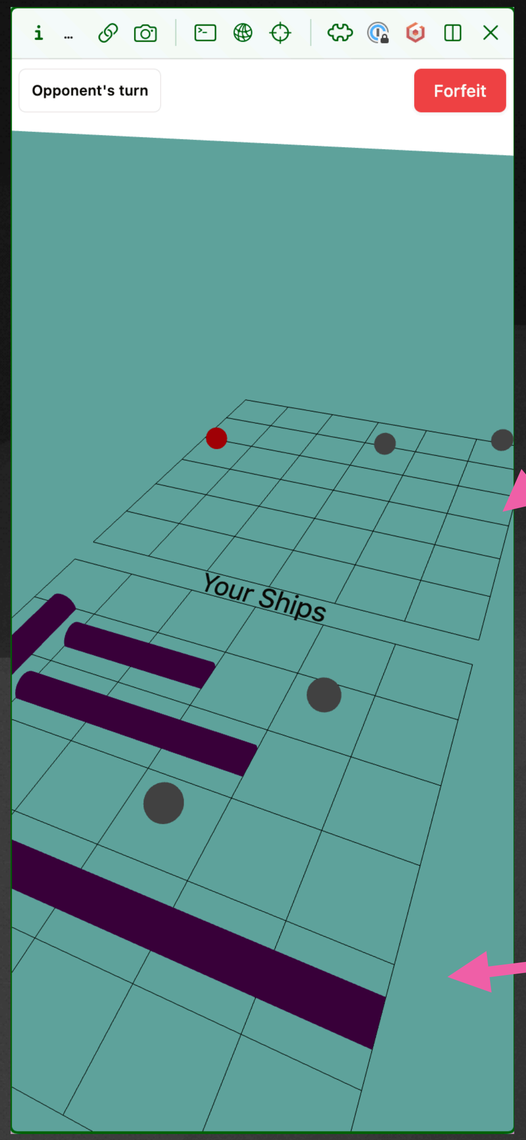


ReactLobby.jsx

```
<mesh
  rotation={[-Math.PI / 2, 0, 0]}
  position={[0, -0.1, 0]}
  ref={planeRef}
  onClick={(event: ThreeEvent<MouseEvent>) => {
    if (!planeRef.current || !currentPlayer) return;

    const {x, y, z} = currentPlayer.position;
    const direction = new Vector3().subVectors(
      event.point,
      new Vector3(x, y, z),
    );
    const distance = direction.length();
    direction.normalize();
    direction.multiplyScalar(Math.min(distance, 3));

    setTarget(direction.add(new Vector3(x, y, z)));
  }}
>
  <planeGeometry args={[100, 100]} />
  <meshStandardMaterial color="lightblue" />
</mesh>
```


Join React Battleship
What do we call you?
Onwards!
Join React Battleship
What do we call you?
Onwards!


Join React Battleship
What do we call you?
Onwards!
Join React Battleship
What do we call you?
Onwards!



GridHelper

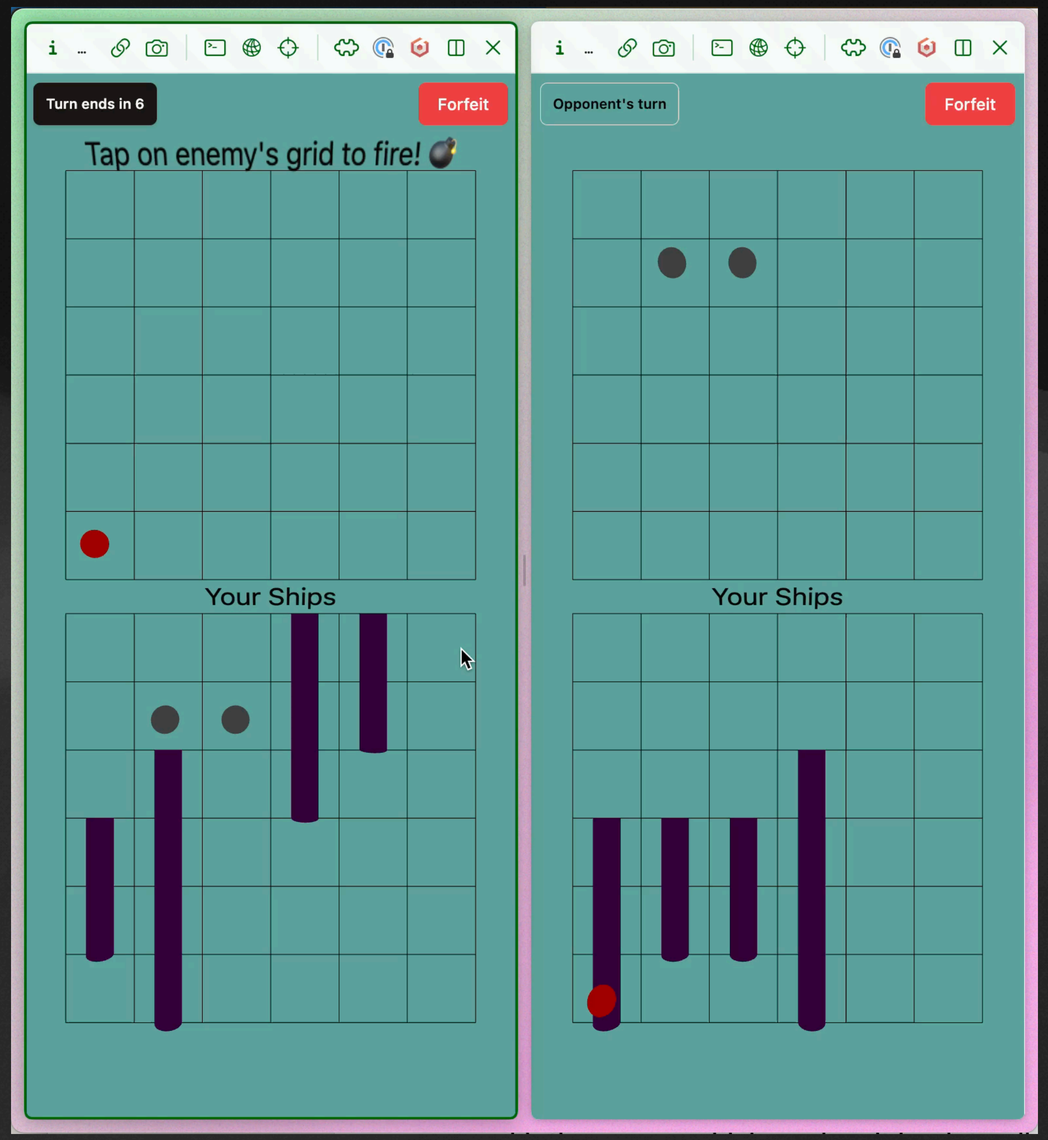
WaterBackground

Game.jsx

```
<WaterBackground
  onClick={(e: ThreeEvent<MouseEvent>) => {
    const x = e.point.x;
    const z = e.point.z + 3.25;
    const isInOpponentGrid =
      x >= -GRID_WIDTH / 2 &&
      x <= GRID_WIDTH / 2 &&
      z >= -GRID_WIDTH / 2 &&
      z <= GRID_WIDTH / 2;

    if (!isInOpponentGrid || !gameState?.yourTurn || showCannon) {
      return;
    }

    const col = Math.floor((x + GRID_WIDTH / 2) / CELL_SIZE);
    const row = Math.floor((z + GRID_WIDTH / 2) / CELL_SIZE);

    setShowCannon(true);
    setCannonEnd(e.point.clone());

    setTimeout(() => {
      setShowCannon(false);
      onCannonFired(row, col);
      onFired();
    }, cannonDuration + 200);
  }}
/>
```

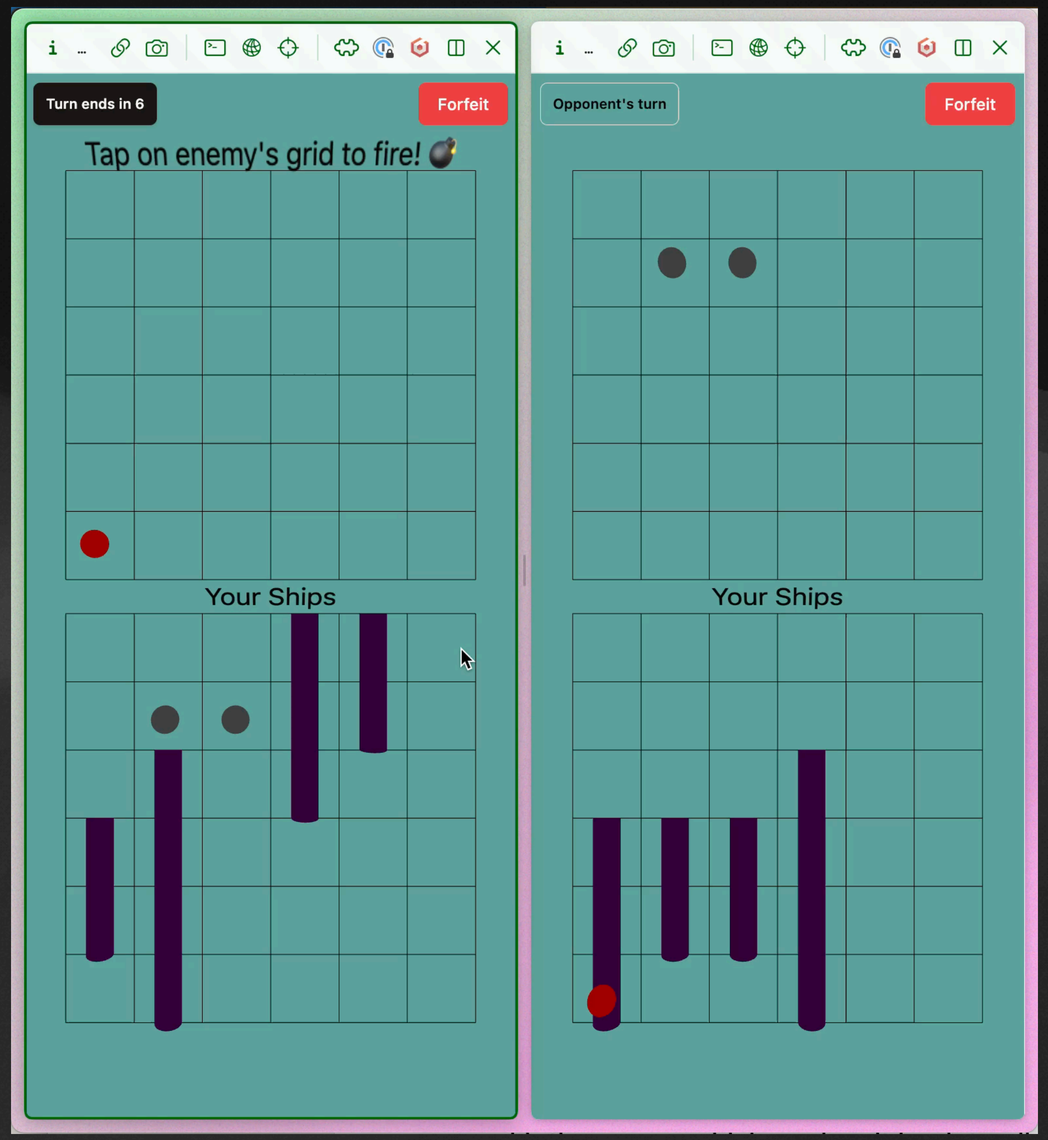
Game.jsx

```jsx
<Grid grid={gameState.yourGrid} position={[0, 0, 3.25]}>
  {gameState.yourShipPositions.map((ship) => {
    const rotationY = ship.direction === "vertical" ? 0 : -Math.PI / 2;

    let positionZ = -GRID_SIZE / 2;
    let positionX = -GRID_SIZE / 2;

    // offset based on size of ship
    positionZ +=
      ship.direction === "horizontal"
        ? (SHIP_SIZE[ship.type] * CELL_SIZE) / 2
        : 0;
    positionX +=
      ship.direction === "vertical"
        ? (SHIP_SIZE[ship.type] * CELL_SIZE) / 2
        : 0;

    // offset based on start position
    positionZ += ship.start.x * CELL_SIZE;
    positionX += ship.start.y * CELL_SIZE;

    // offset to center of cell based on direction
    positionZ += ship.direction === "vertical" ? CELL_SIZE / 2 : 0;
    positionX += ship.direction === "horizontal" ? CELL_SIZE / 2 : 0;

    const position = new Vector3(positionX, 0, positionZ);
    return (
      <mesh
        position={position}
        rotation={[0, rotationY, -Math.PI / 2]}
        key={ship.type}
      >
        <cylinderGeometry args={[0.2, 0.2, SHIP_SIZE[ship.type], 32]} />
        <meshStandardMaterial color="purple" />
      </mesh>
    );
  })}
</Grid>
```



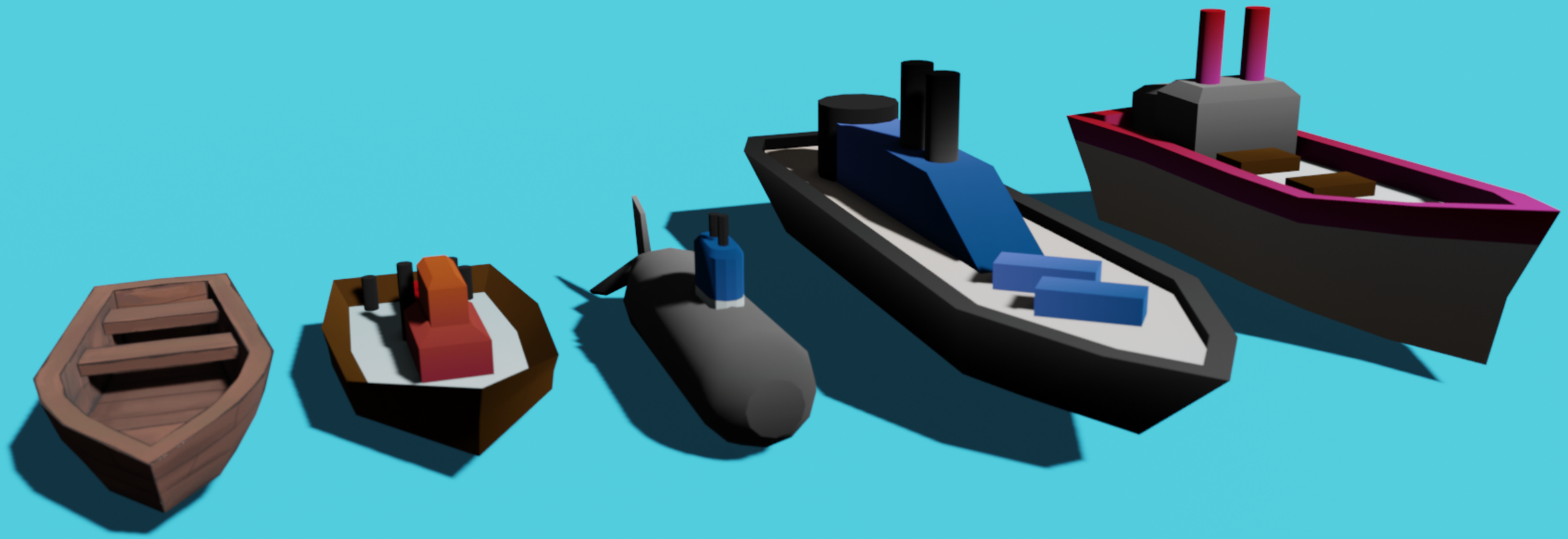
Game.jsx

```jsx
<Grid grid={gameState.yourGrid} position={[0, 0, 3.25]}>
  {gameState.yourShipPositions.map((ship) => {
    const rotationY = ship.direction === "vertical" ? 0 : -Math.PI / 2;

    let positionZ = -GRID_SIZE / 2;
    let positionX = -GRID_SIZE / 2;

    // offset based on size of ship
    positionZ +=
      ship.direction === "horizontal"
        ? (SHIP_SIZE[ship.type] * CELL_SIZE) / 2
        : 0;
    positionX +=
      ship.direction === "vertical"
        ? (SHIP_SIZE[ship.type] * CELL_SIZE) / 2
        : 0;

    // offset based on start position
    positionZ += ship.start.x * CELL_SIZE;
    positionX += ship.start.y * CELL_SIZE;

    // offset to center of cell based on direction
    positionZ += ship.direction === "vertical" ? CELL_SIZE / 2 : 0;
    positionX += ship.direction === "horizontal" ? CELL_SIZE / 2 : 0;

    const position = new Vector3(positionX, 0, positionZ);
    return (
      <mesh
        position={position}
        rotation={[0, rotationY, -Math.PI / 2]}
        key={ship.type}
      >
        <cylinderGeometry args={[0.2, 0.2, SHIP_SIZE[ship.type], 32]} />
        <meshStandardMaterial color="purple" />
      </mesh>
    );
  })}
</Grid>
```



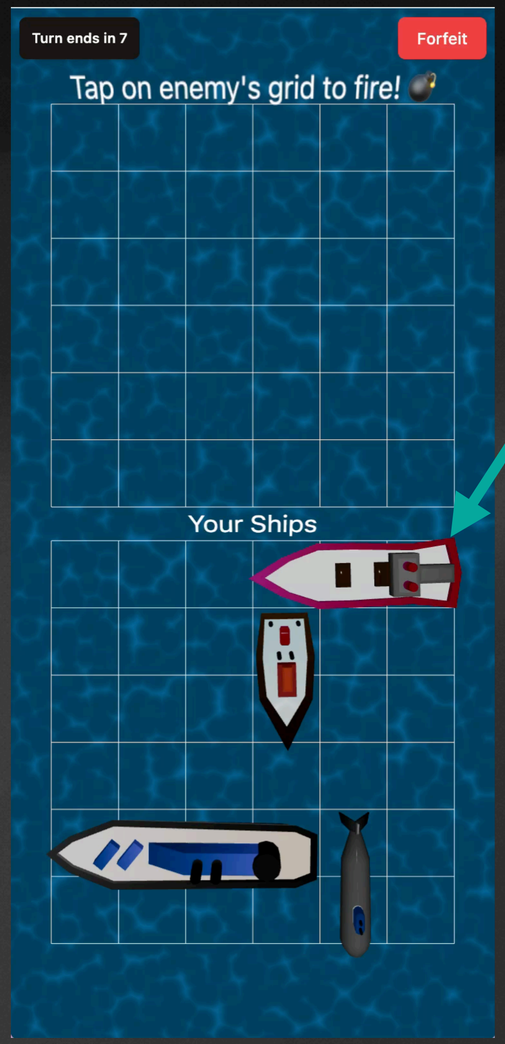
✨ Production

Models



Models in game

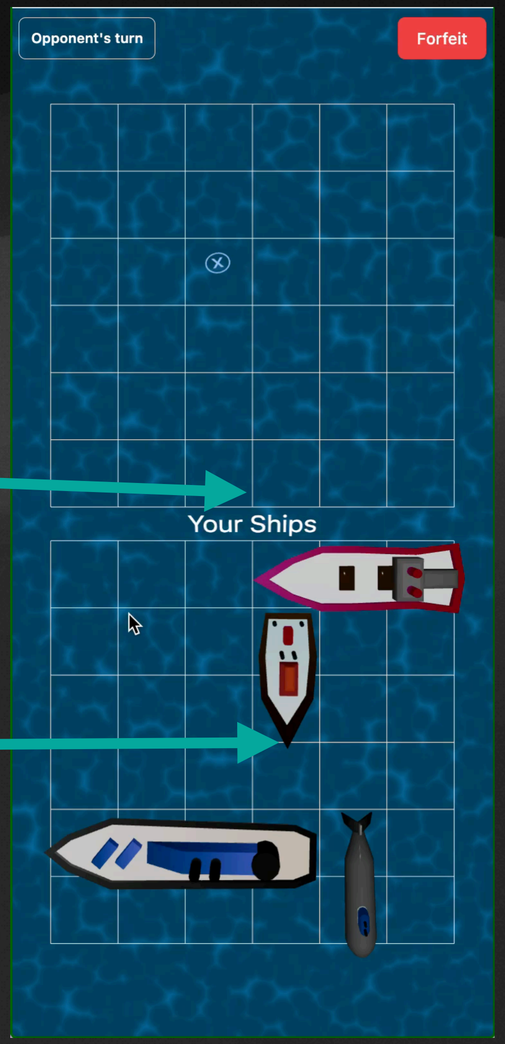
Animations



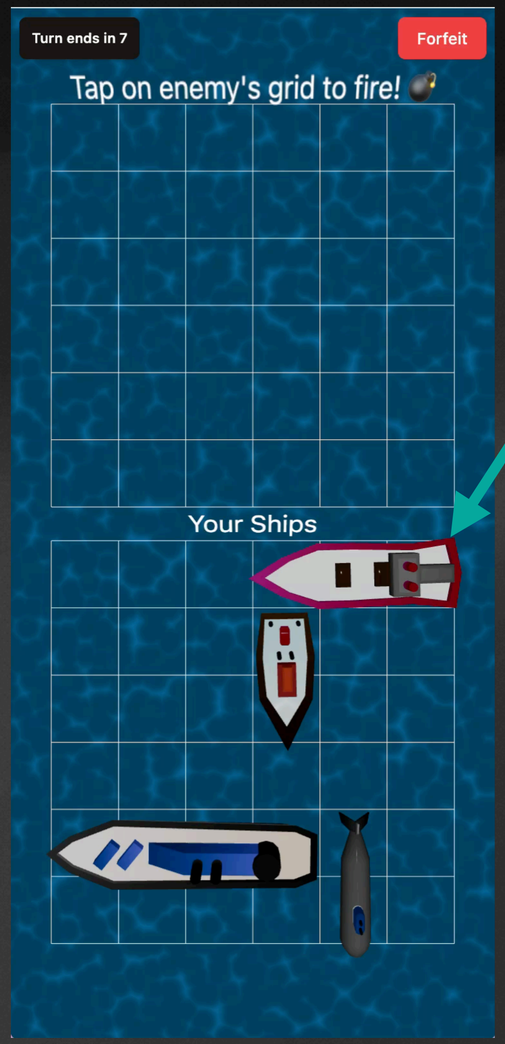
<Float>

useSpring

model animations



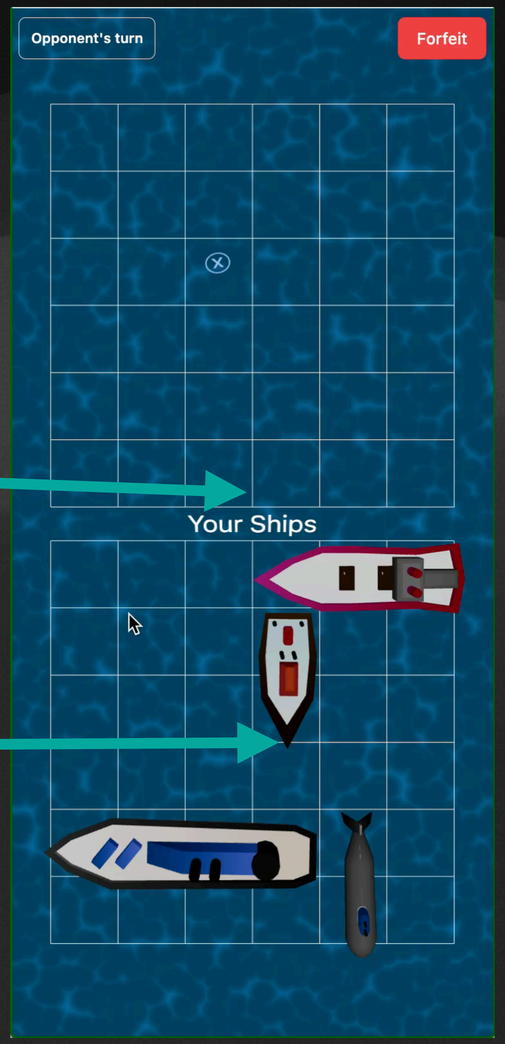
Animations



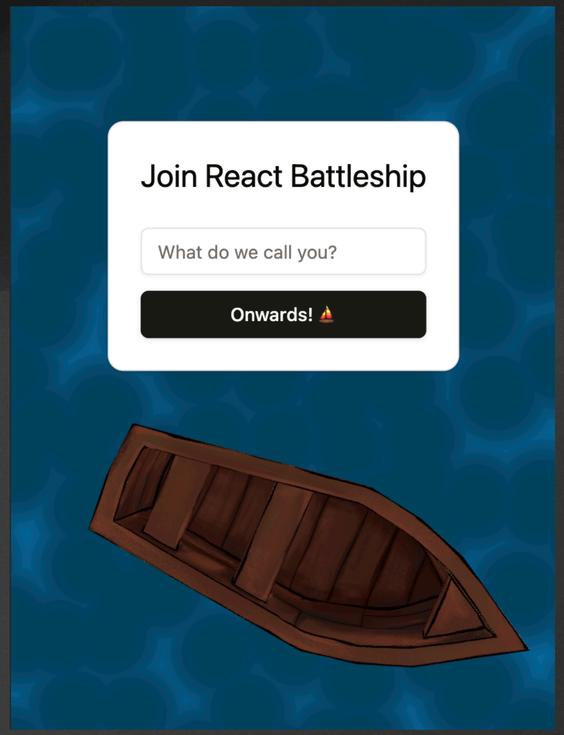
<Float>

useSpring

model animations



UI Improvements



⛵ Demo

bit.ly/react-battleship

Resources

- Wawa Sensei's React Three Fiber Ultimate Guide
- Bruno Simon's Three.js Journey
- WebGameDev Community (newsletter, discord server)

Thank you ✨

@maya_ndljk

@mayacoda