3D im Browser

Eine Einführung in babylon.JS

@merelyChristina & @merelyAnna

Who are we?

CHRISTINA

merelyChristina

ANNA

merelyAnna

codecentric

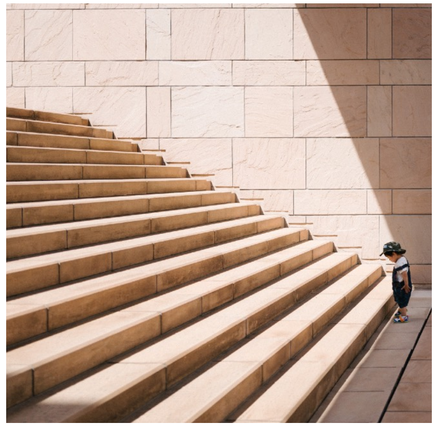
Who are you?

I am Cevapcici Christina :D

I am Ananas Anna :D

What are we doing today?

3D (Duh…)

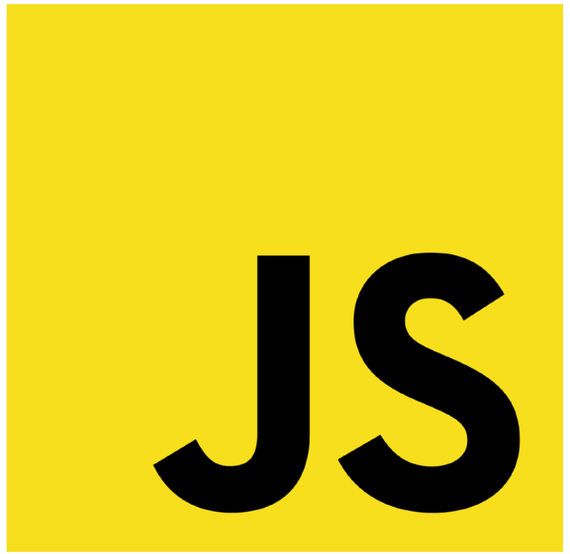


Basics

¯_(ツ)_/¯

Programming experience?

Experience with JavaScript/TypeScript?



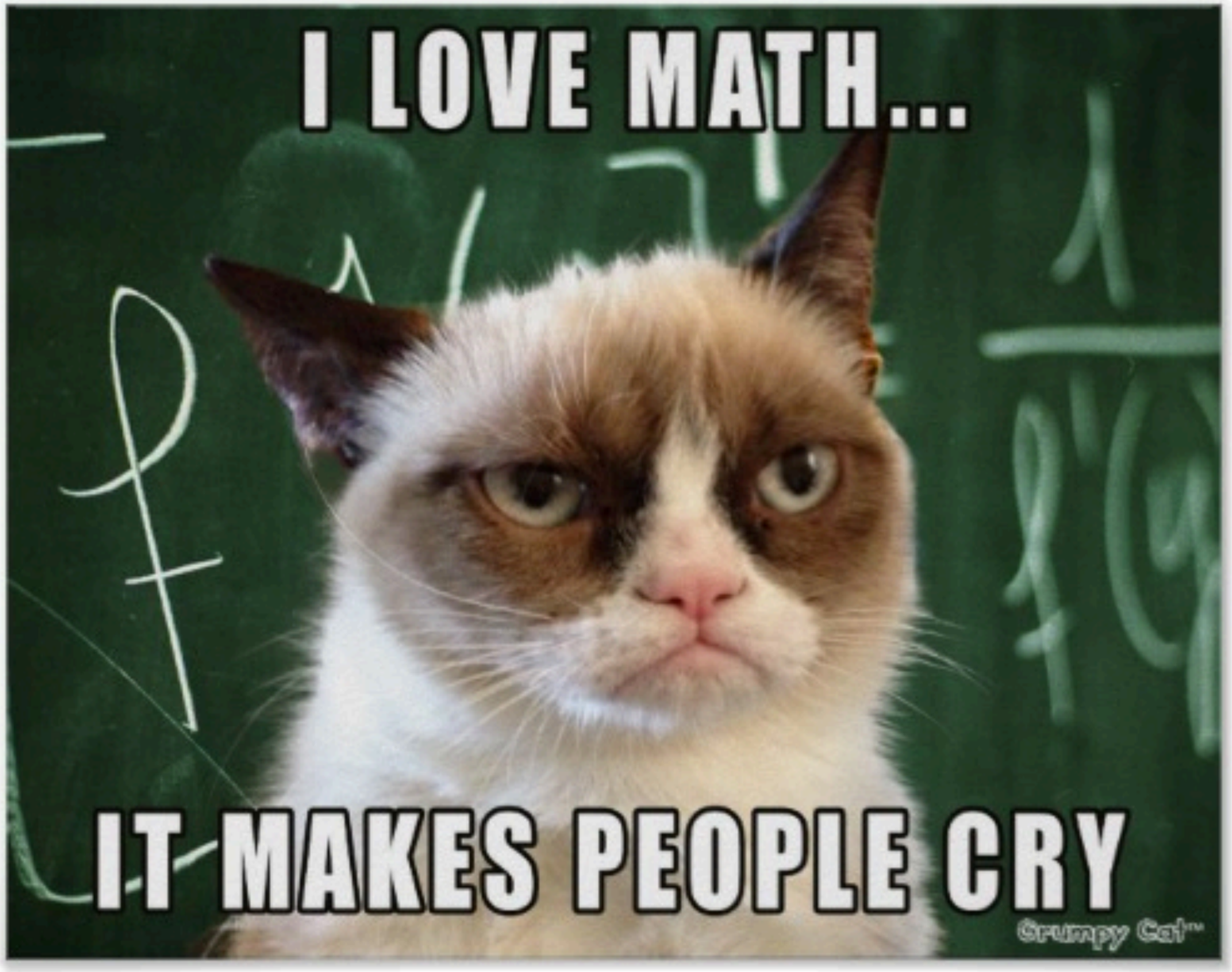
Experience with git?

Access to the github repo?

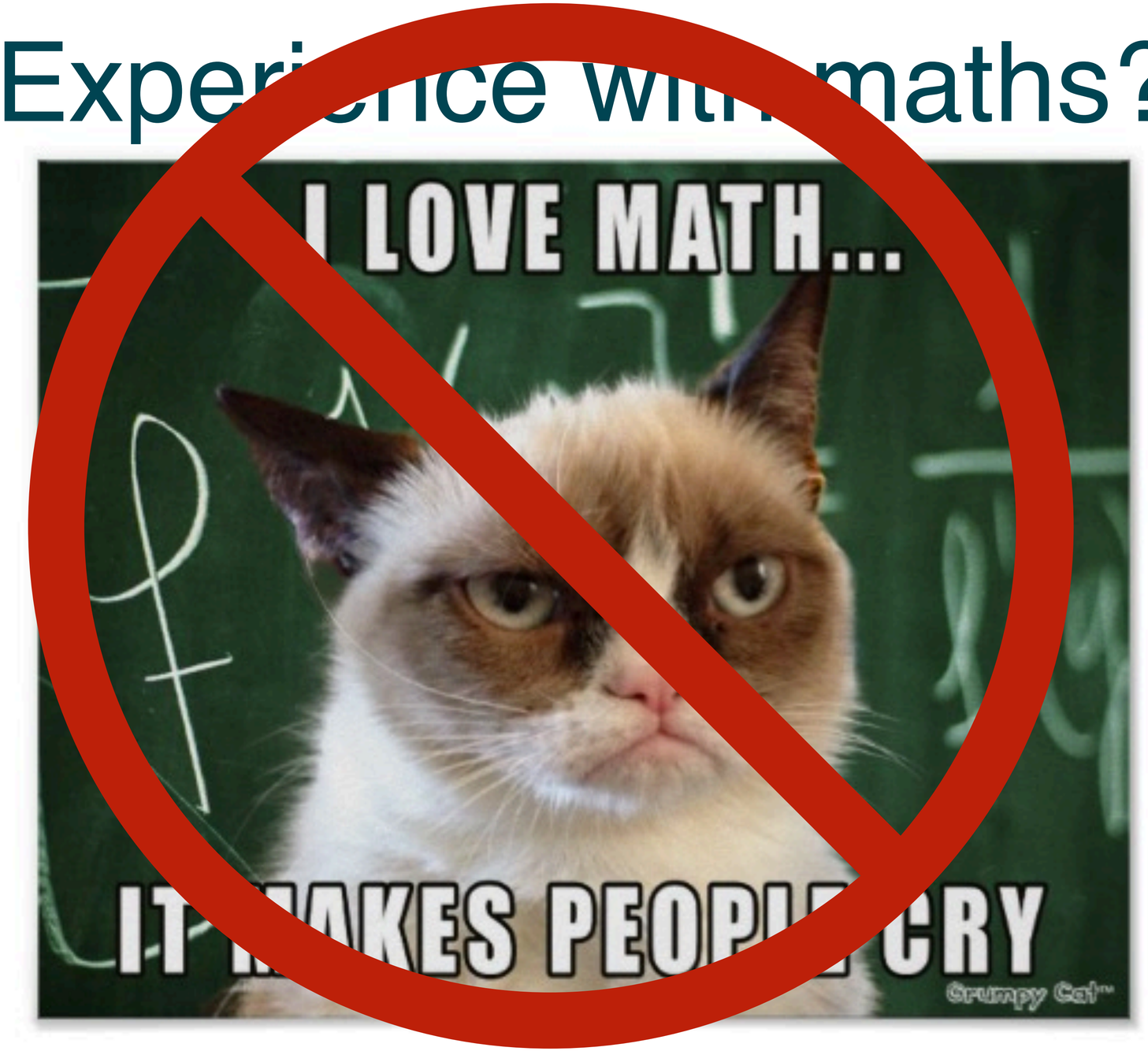
https://github.com/merelyAnna/spartakiade-babylonJSworkshop

OR https://bit.ly/2Gzep1l

Experience with maths?



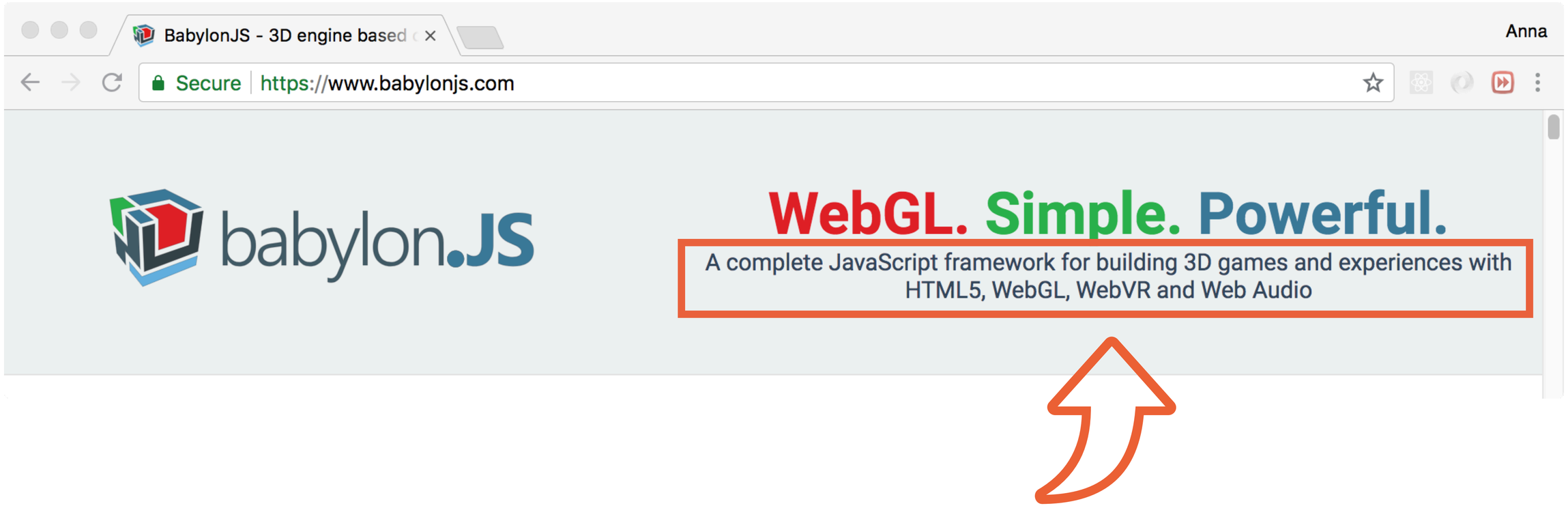
Experience with maths?



babylon.JS

What is it?

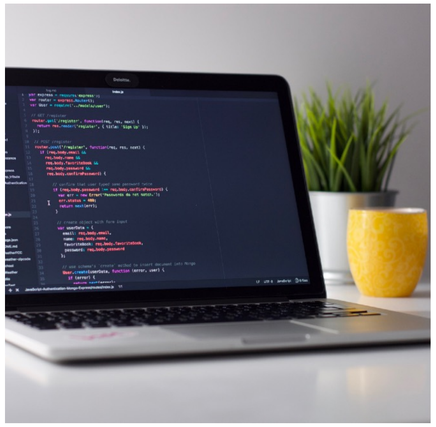
What is babylon.JS?



What is babylon.JS?

3D game engine
running in a browser

What can it do?



Game Engine

Plattform

Runs in a browser

What else can it do?

Animation

Special FX

Debug Layer

Why use it?

Getting started

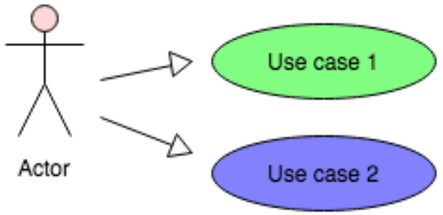
Runs in a browser

Extensive

Why not use it?

Extensive

Limits

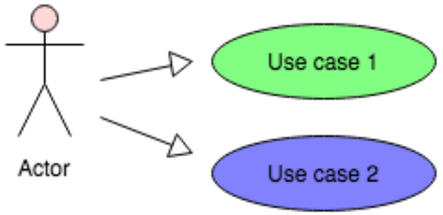


Use Case

Why not use it?

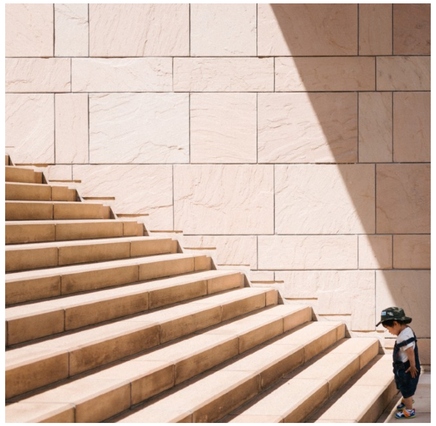
Extensive

Limits



Use Case

Our experience

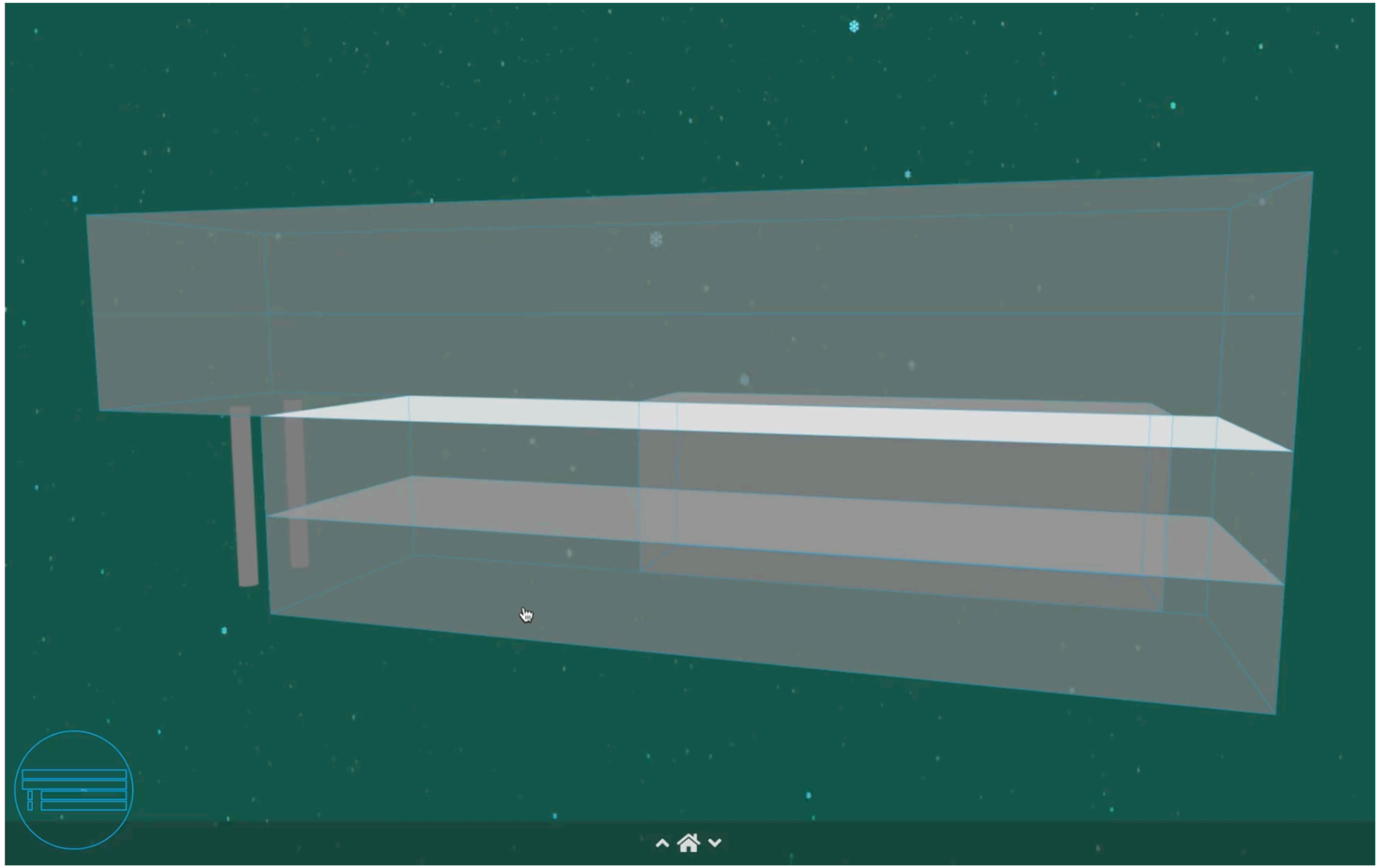


Learning curve

Performance

Community & Resources

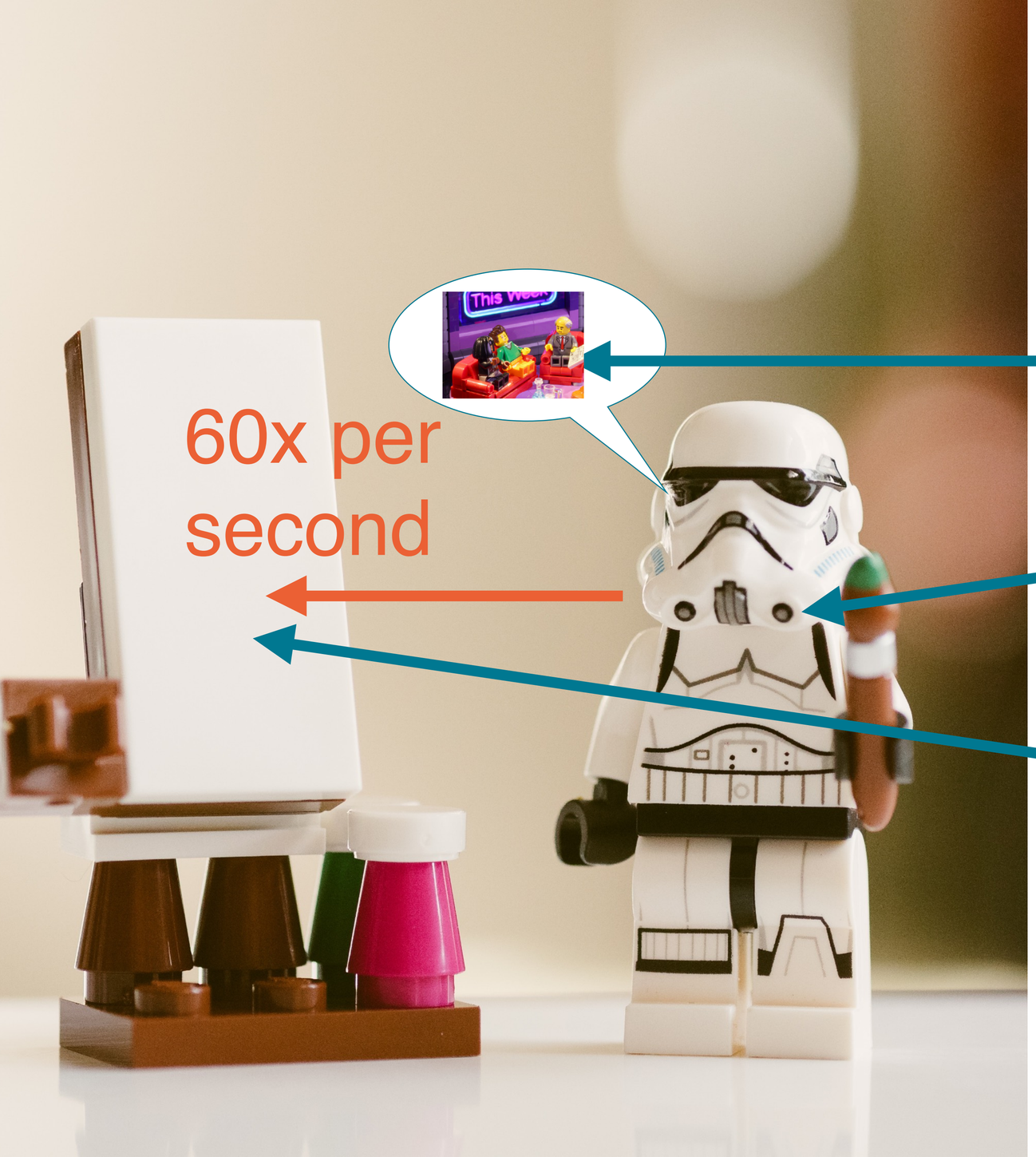
What do we use it for?



babylon.JS

How does it work?

How does it work?



Scene

Engine

Canvas

Canvas

```
<html>
  <head> </head>

  <body style="overflow: hidden; margin: 0px">
    <canvas style="width: 100%; height: 100%;" id="babylon-canvas"></canvas>
  </body>
</html>
```

Getting the engine running

```
const initGame = () => {
  const canvas = document.getElementById("babylon-canvas");

  const engine = new Engine(canvas, true);
  const scene = new Scene(engine);

  engine.runRenderLoop(function() {
    scene.render();
  });
};

initGame();
```

All this is already in the repo.

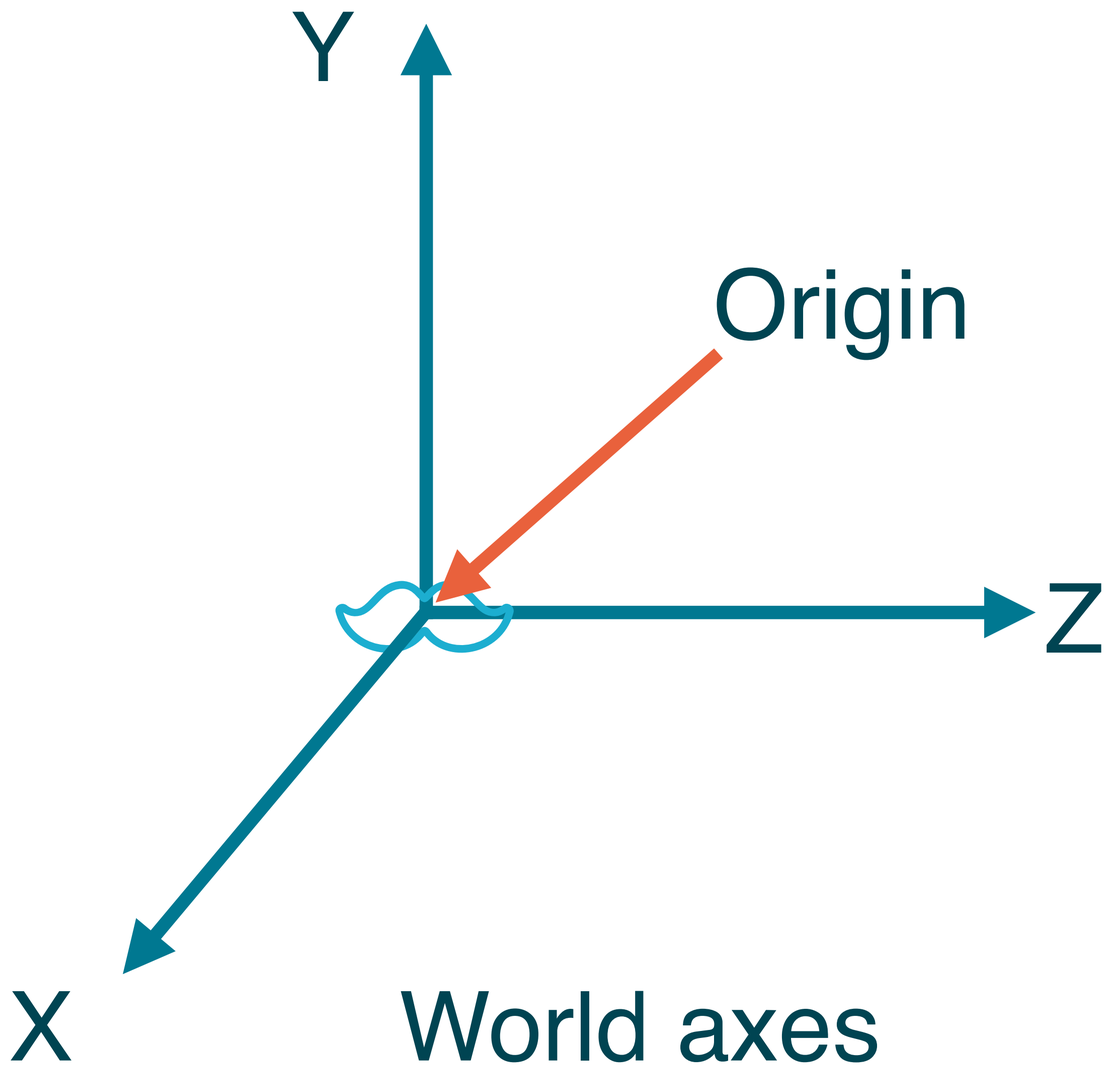

I want to see
something
on the canvas

Parameters

```
const thing = new Thing("name", { }, scene);
```

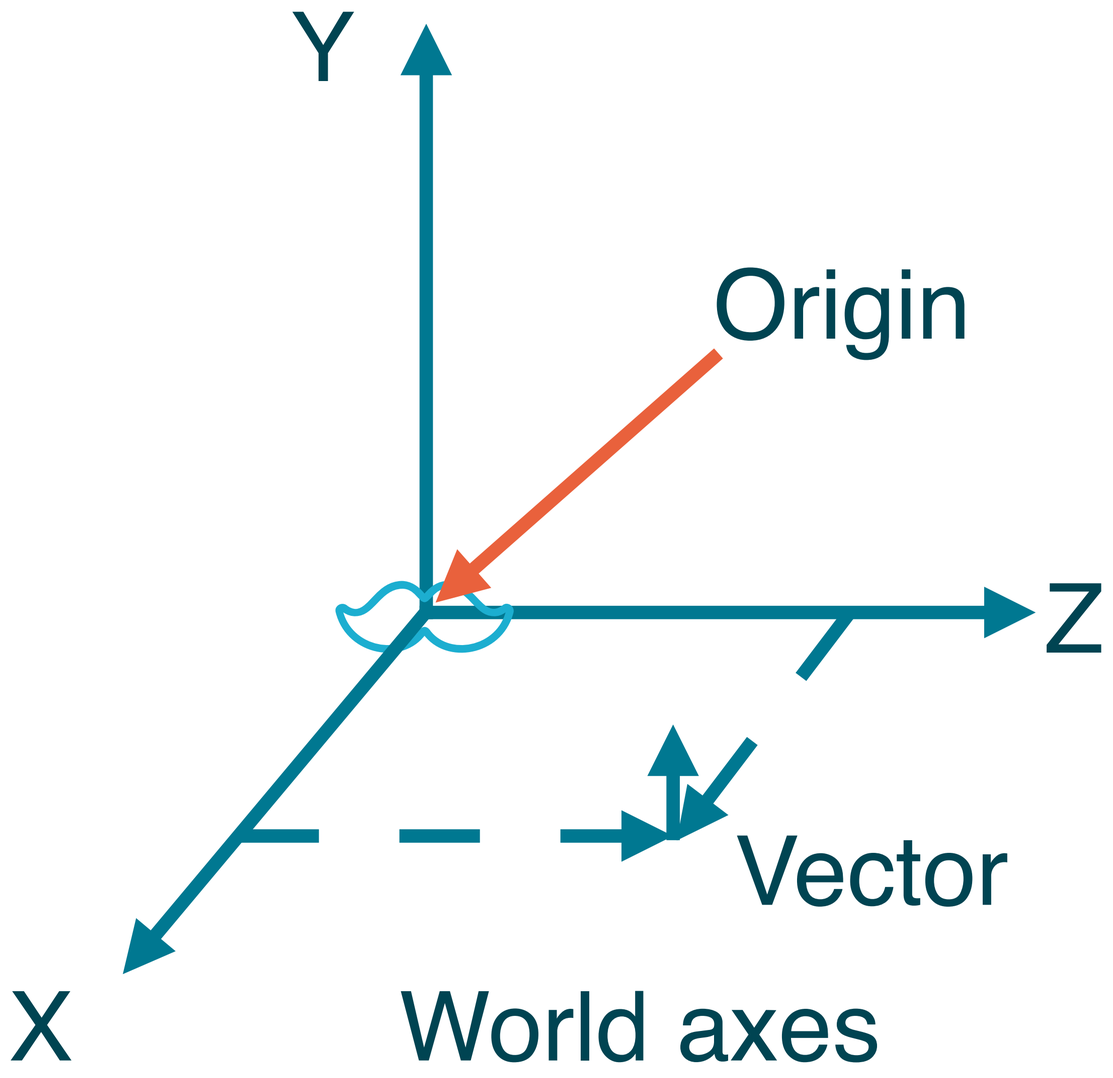
class name options scene

Positioning

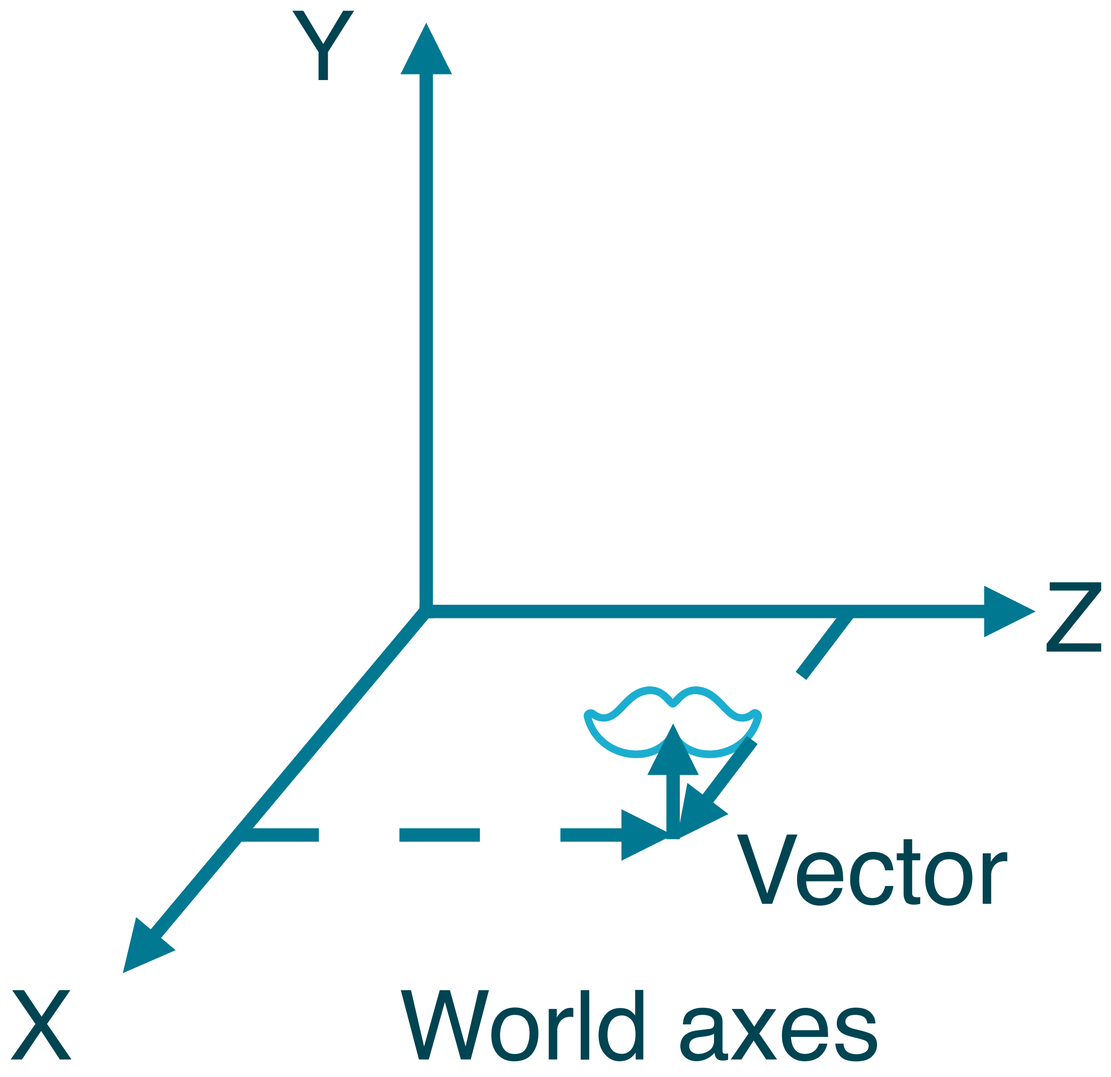


World axes

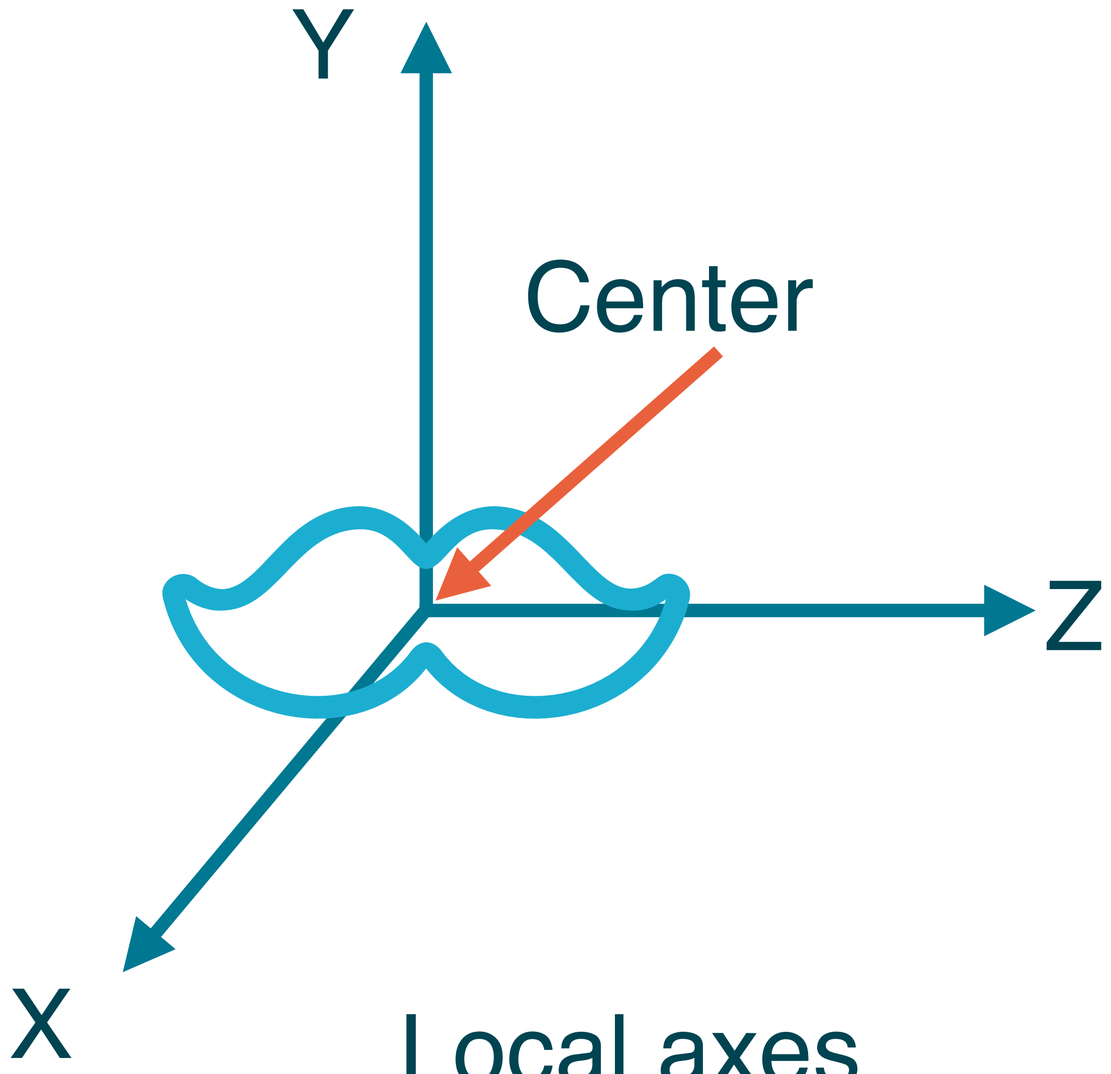
Positioning



Positioning

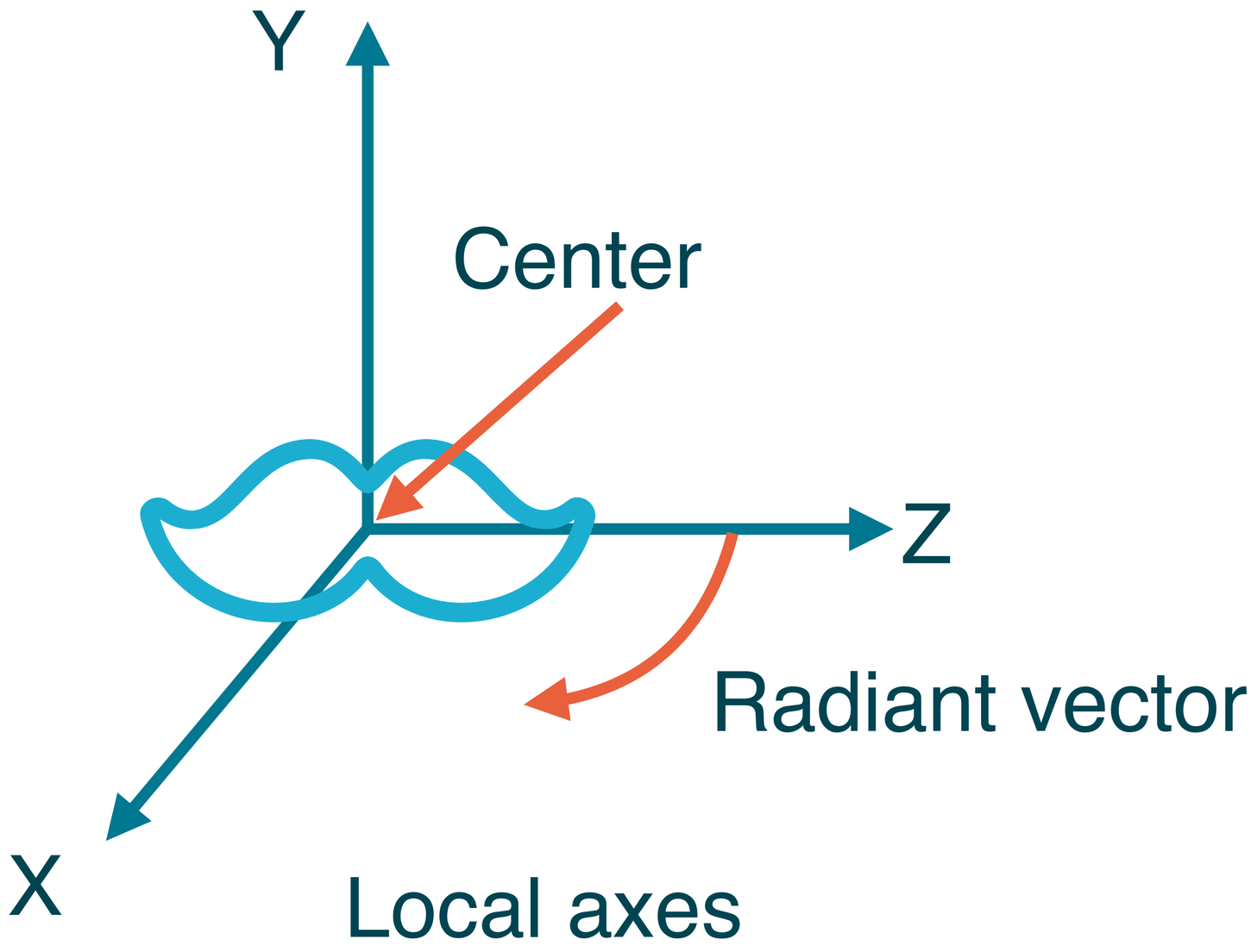


Rotation

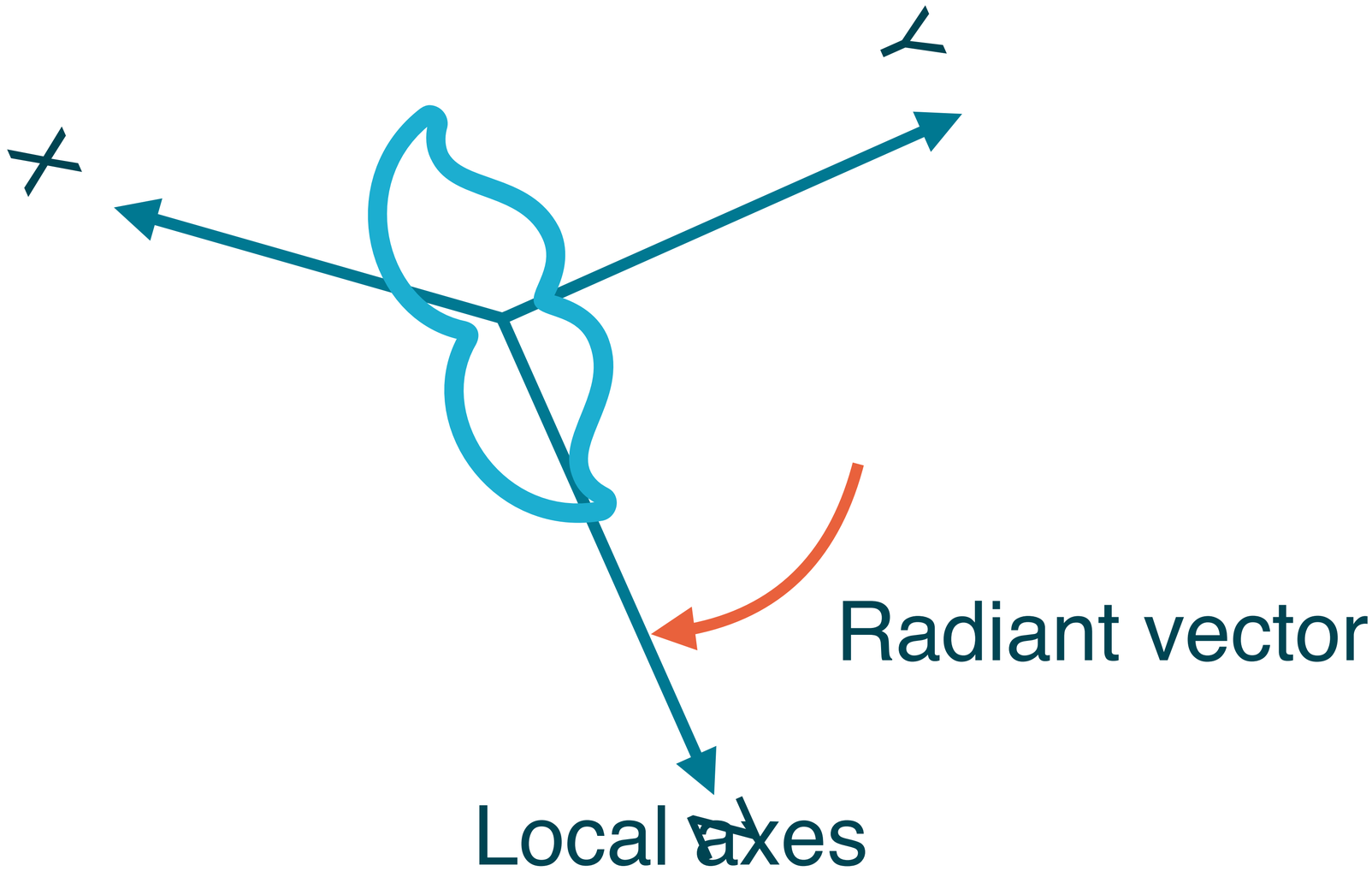


Local axes

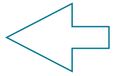
Rotation

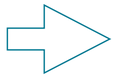


Rotation



Universal camera


Left

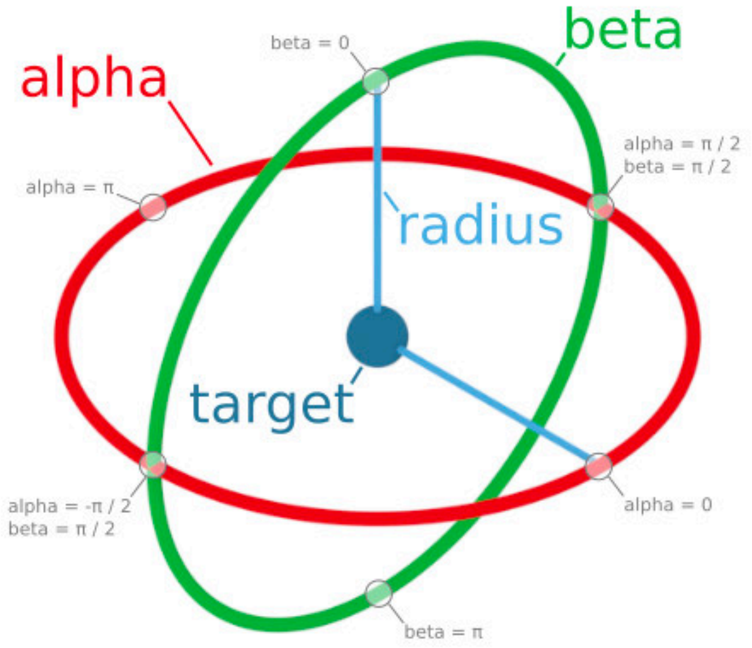

Right


Forward


Backward

```
const camera = new UniversalCamera("camera",
new Vector3(0, 0, -10), scene);

// This targets the camera to scene origin
camera.setTarget(Vector3.Zero());

// This attaches the camera to the canvas
camera.attachControl(canvas, true);
```

Arc rotate camera

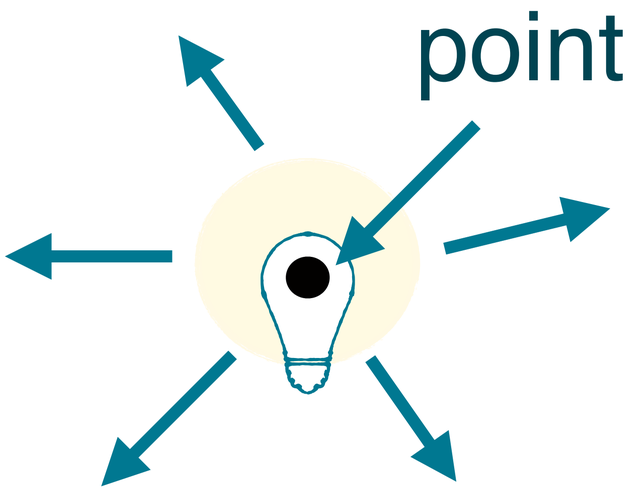



```
const camera = new ArcRotateCamera("camera", 0,
0, 20, new Vector3(0, 0, 0), scene);

// This attaches the camera to the canvas
camera.attachControl(canvas, true);
```

Image source: https://doc.babylonjs.com/babylon101/cameras#arc-rotate-camera

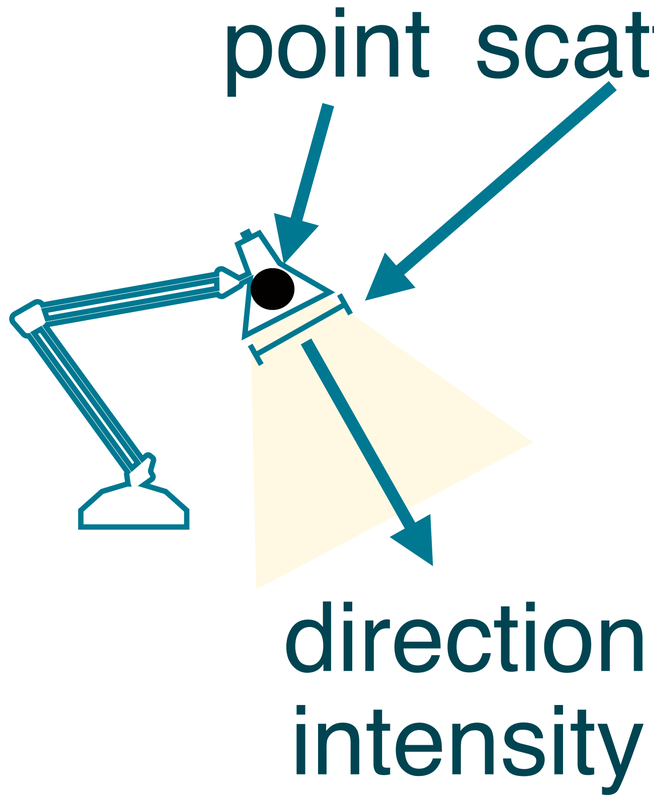
Without light
everything
is black

Point light

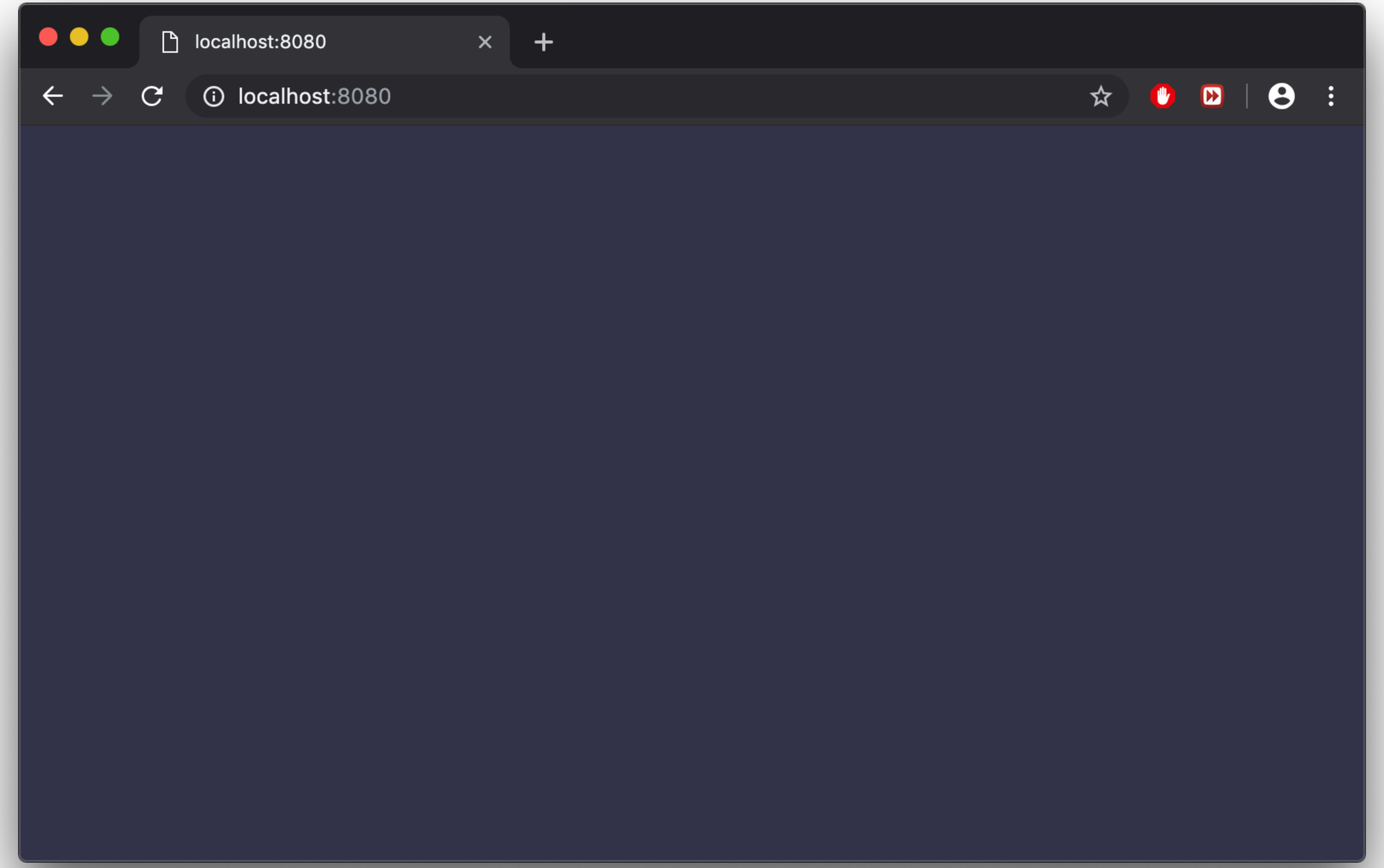



```
const light = new PointLight(
  "pointLight",
  new Vector3(0, 10, 0),
  scene
);
```

Spot light




```
const light = new SpotLight(
  "spotLight",
  new Vector3(0, 10, 0),
  new Vector3(0, -1, 0),
  Math.PI / 2,
  10,
  scene
);
```

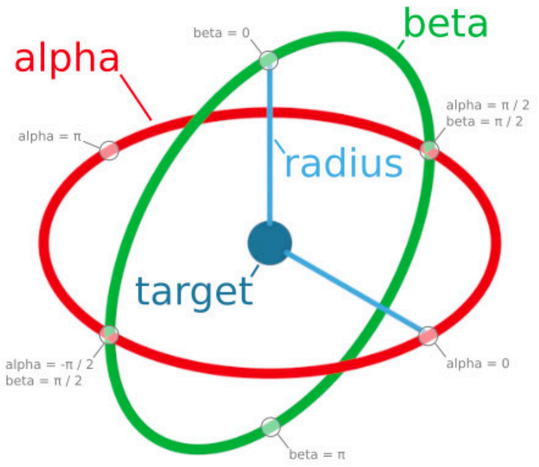

localhost:8080
localhost:8080

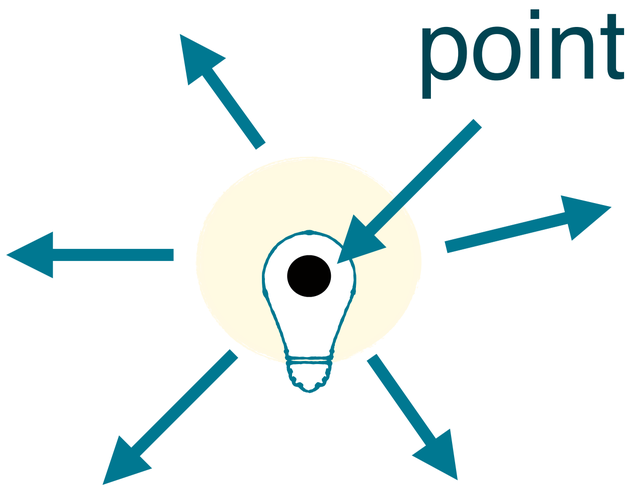
Task 1: Let there be light

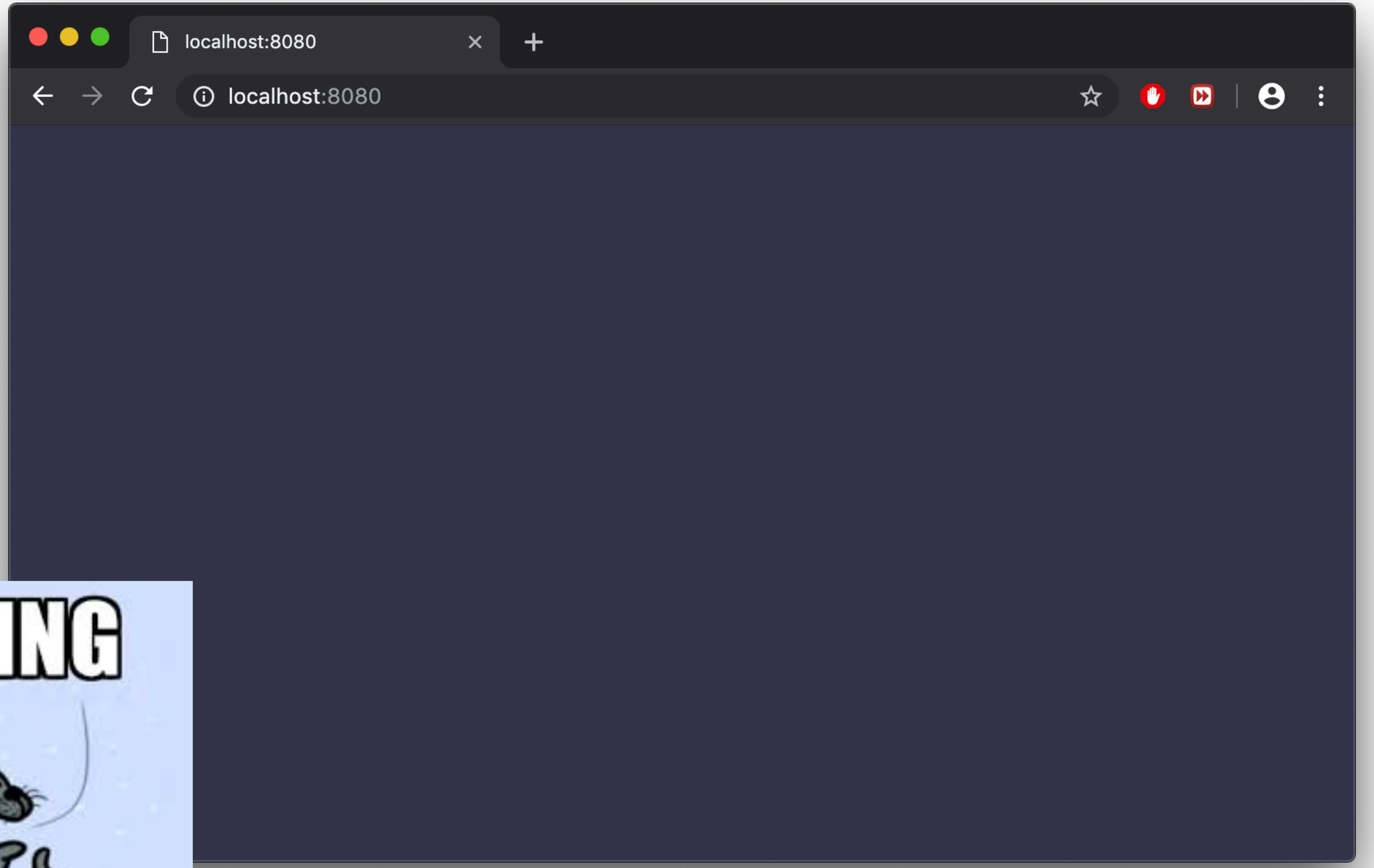
Add an **ArcRotateCamera**.

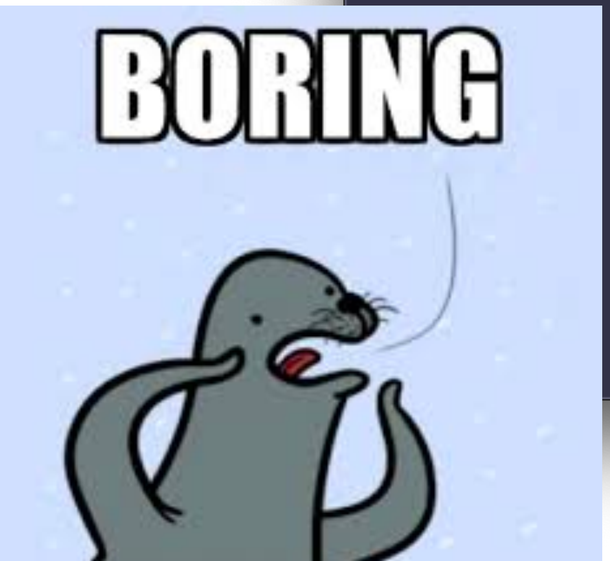
Point the camera to the Origin.

Add a **PointLight**.

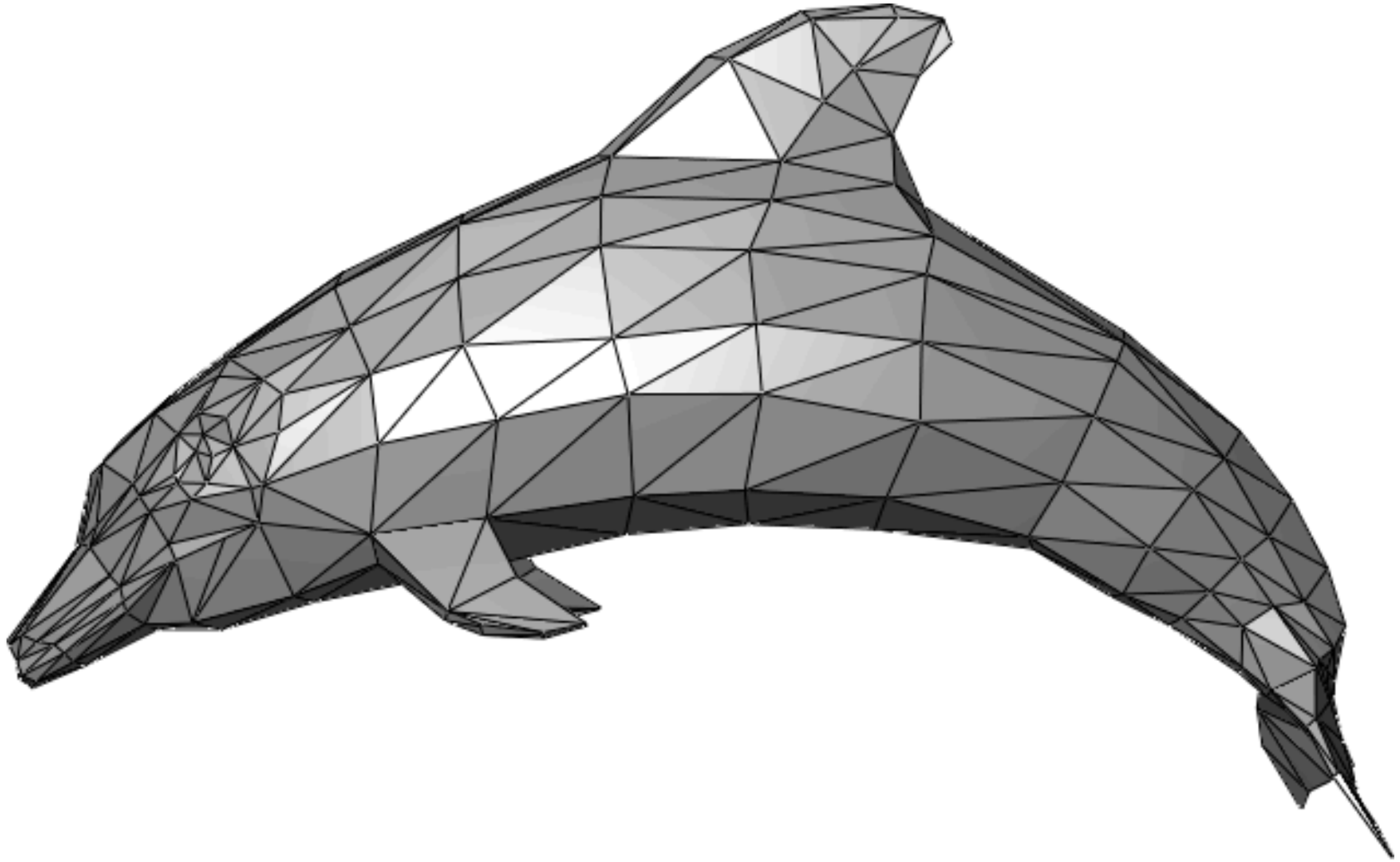




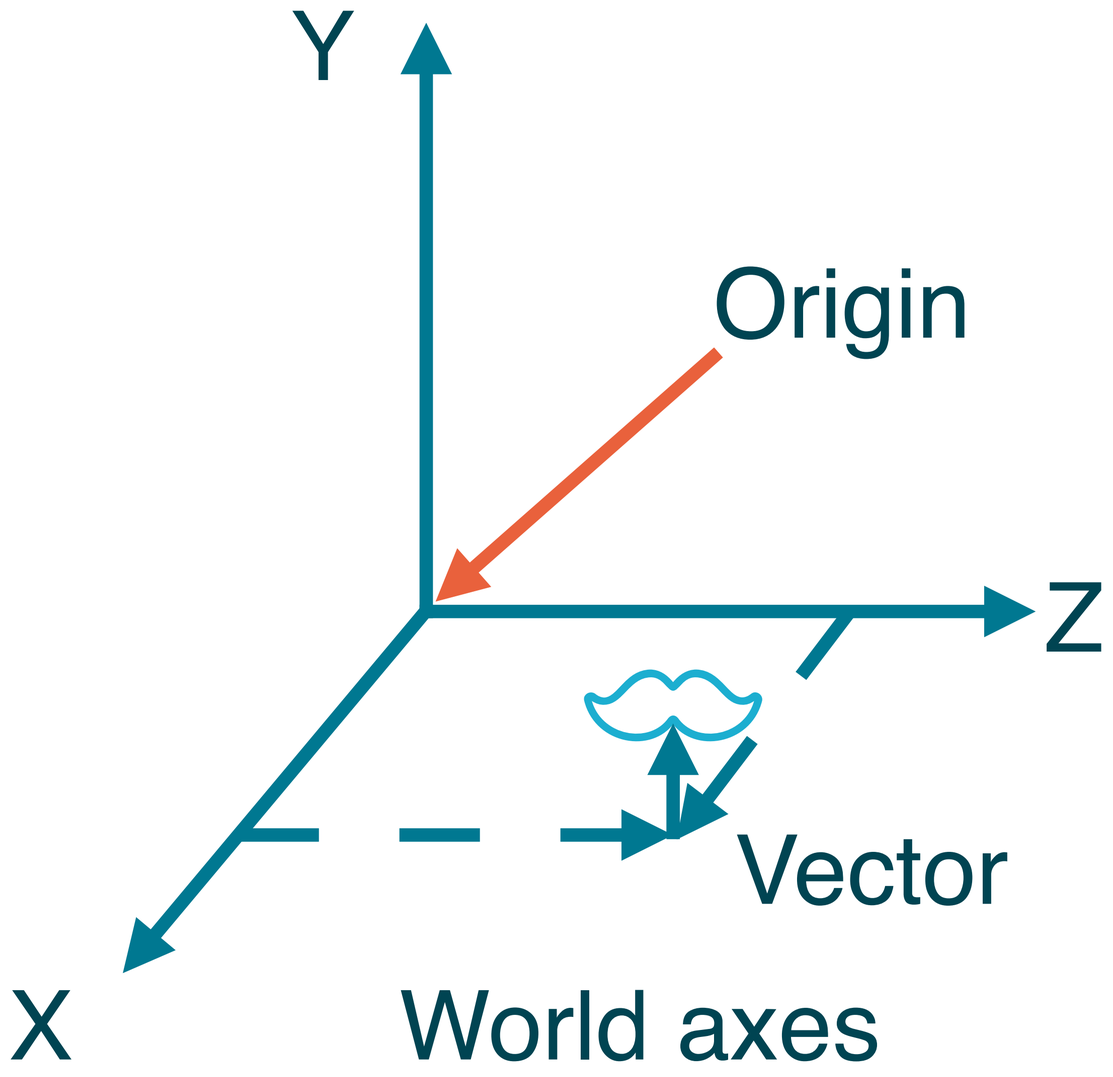

localhost:8080
localhost:8080


BORING

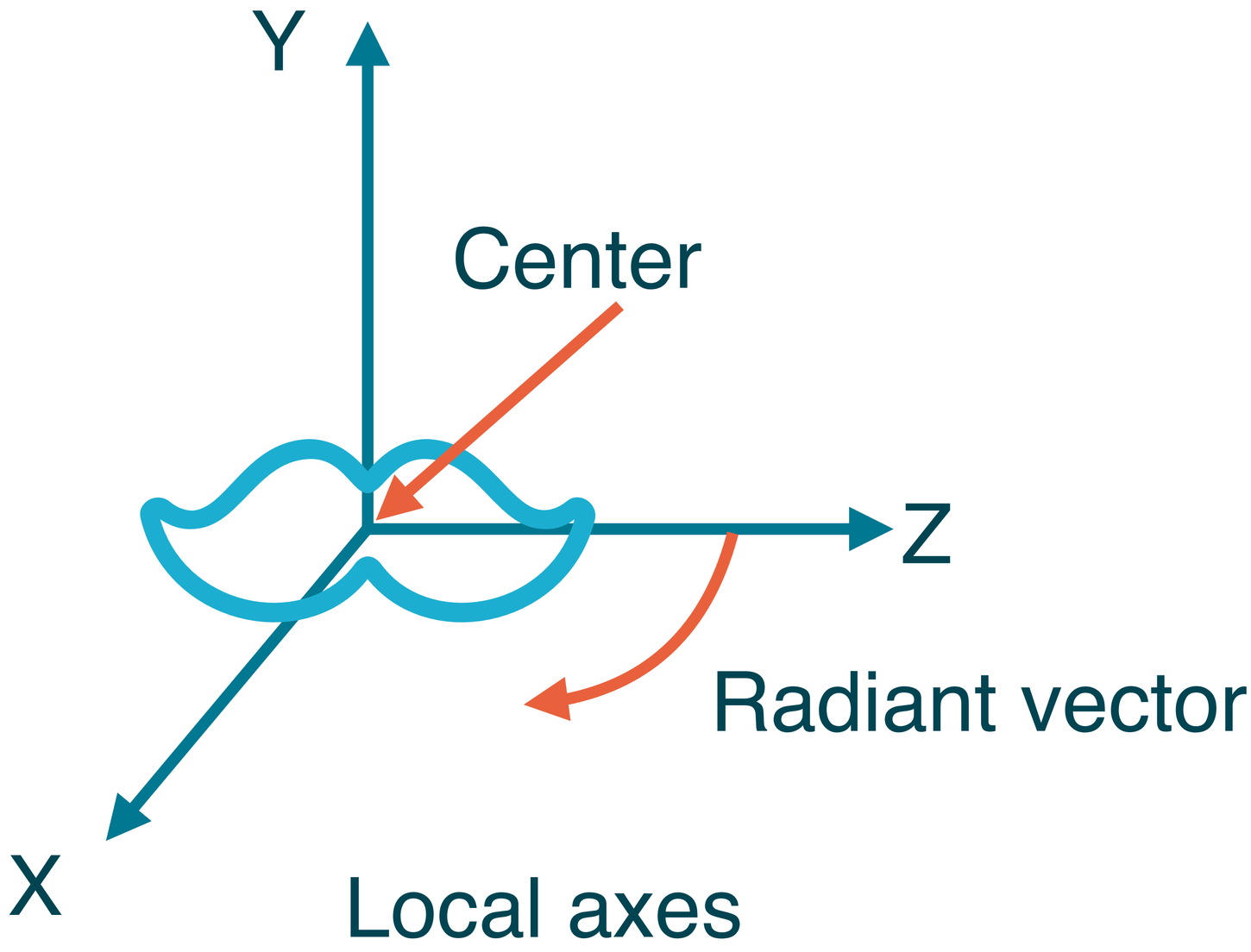
Mesh



Positioning



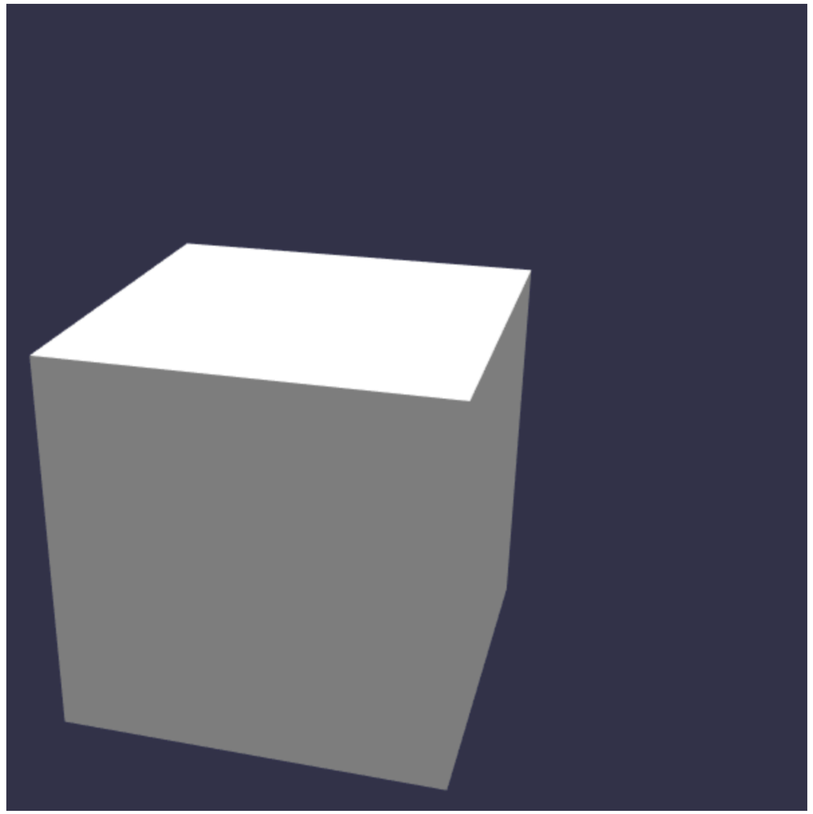
Rotation

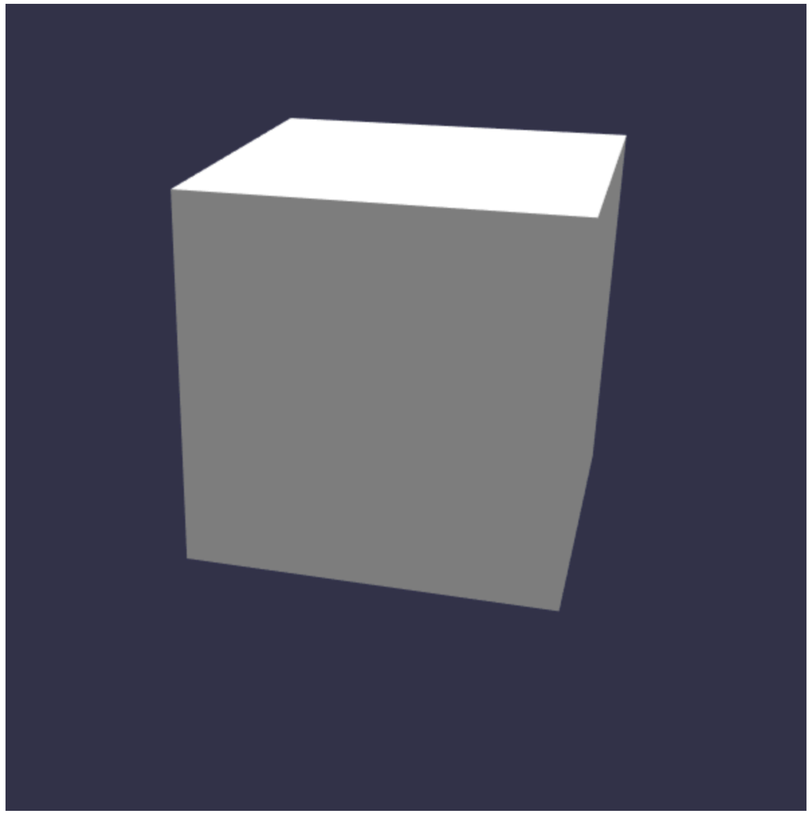


Example

```
//Creating the box mesh
const box = MeshBuilder.CreateBox(
    "box",
    { width: 10, height: 10, depth: 10 },
    scene
);

// Positioning the box
box.position.x = 7;
box.position.y = 8;
box.position.z = 9;

// Rotating the box
box.rotation = new Vector3(2, 3, 4);
```

Example



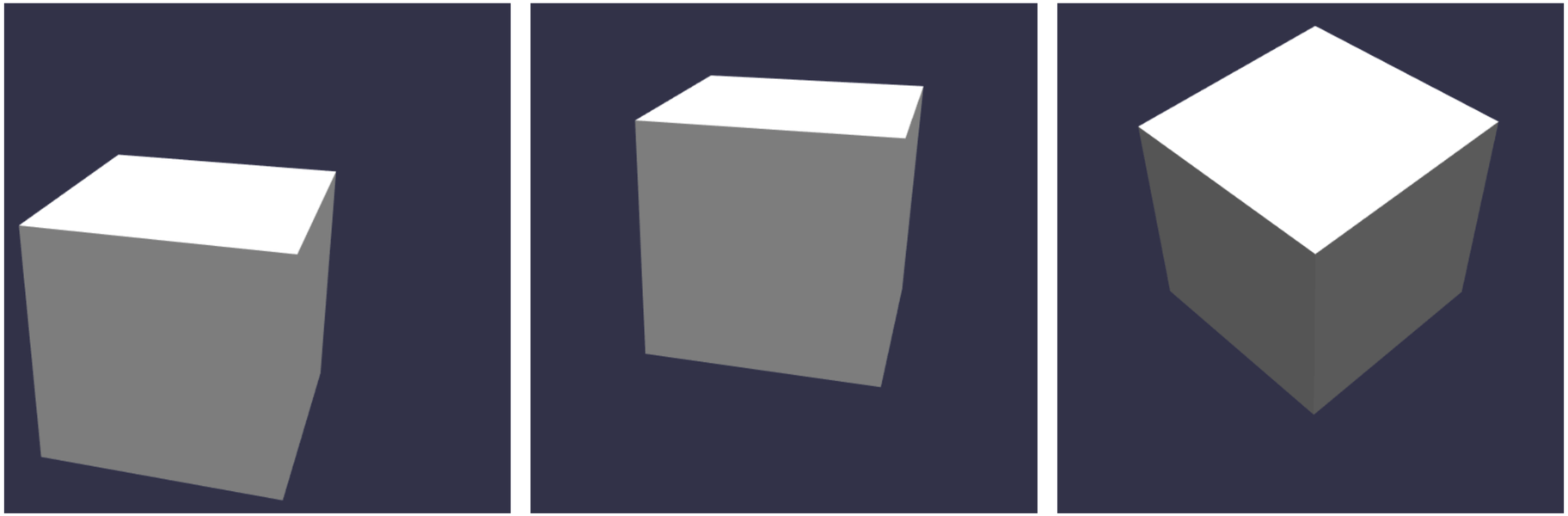


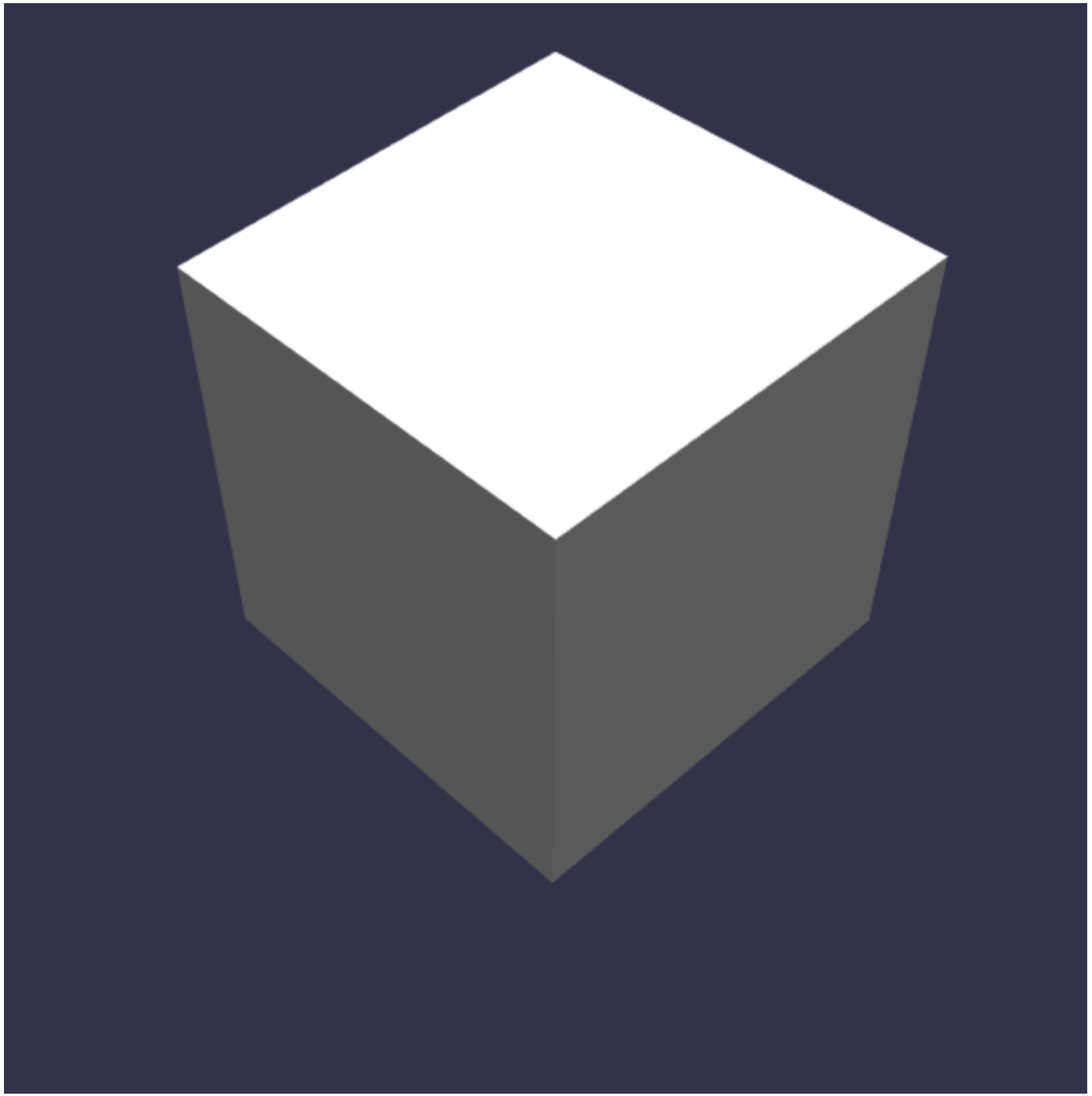
Task 2: A box

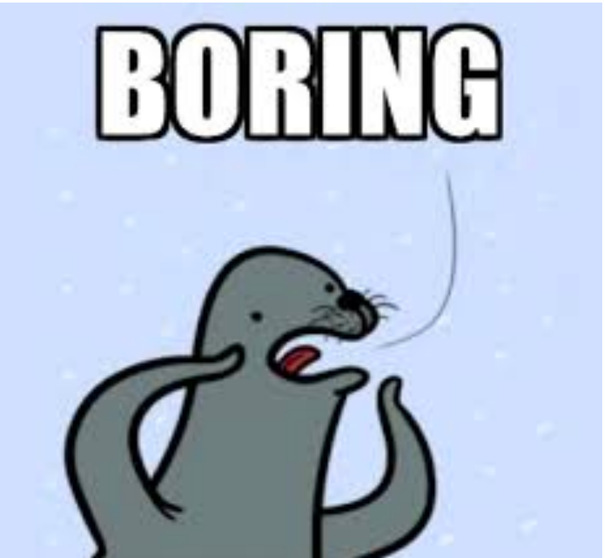
Use the **MeshBuilder** to create a **box**.

Position the box.

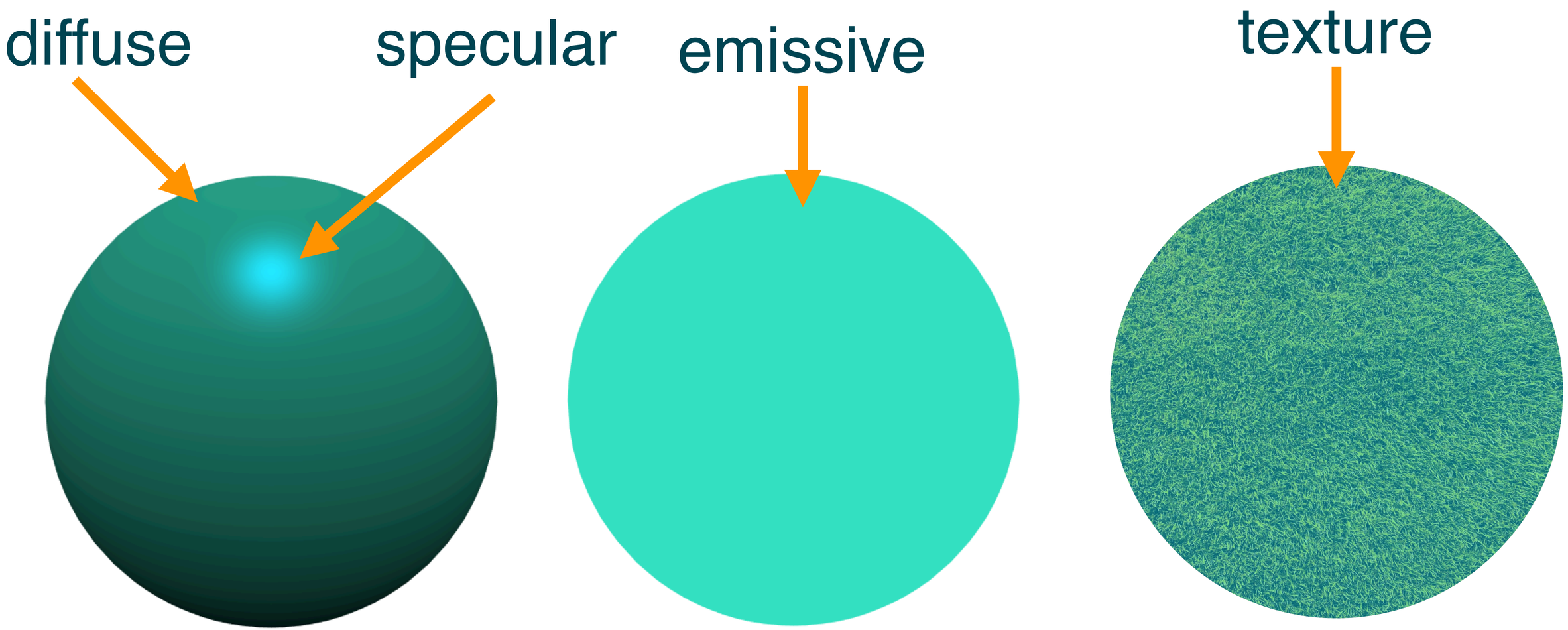
Rotate the box.






BORING

Materials

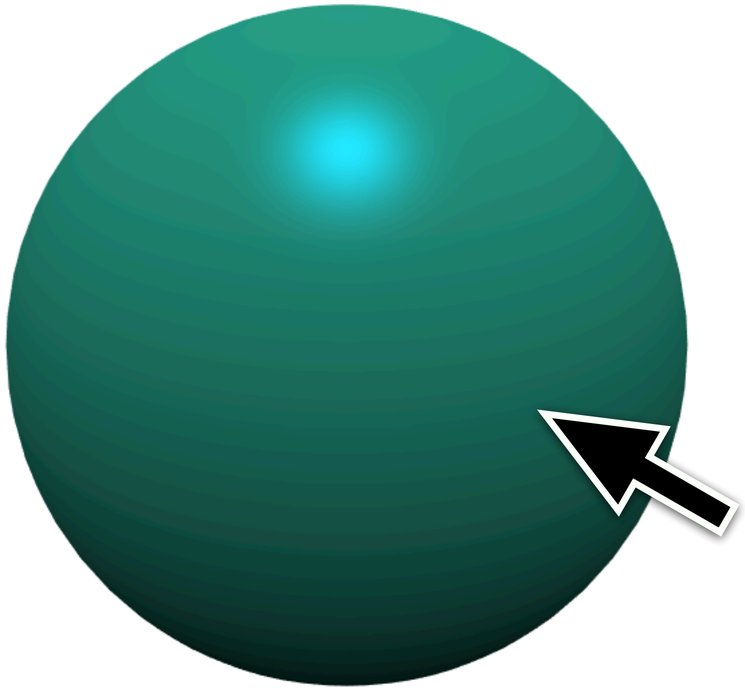


Materials

```
// Creating the material
const material = new StandardMaterial("material");
material.diffuseColor = new Color3(0.22, 0.89, 0.76);
material.specularColor = new Color3(0, 0.5, 0.76);

box.material = material;
```

User interaction

Action Manager



Actions

Actions Typ

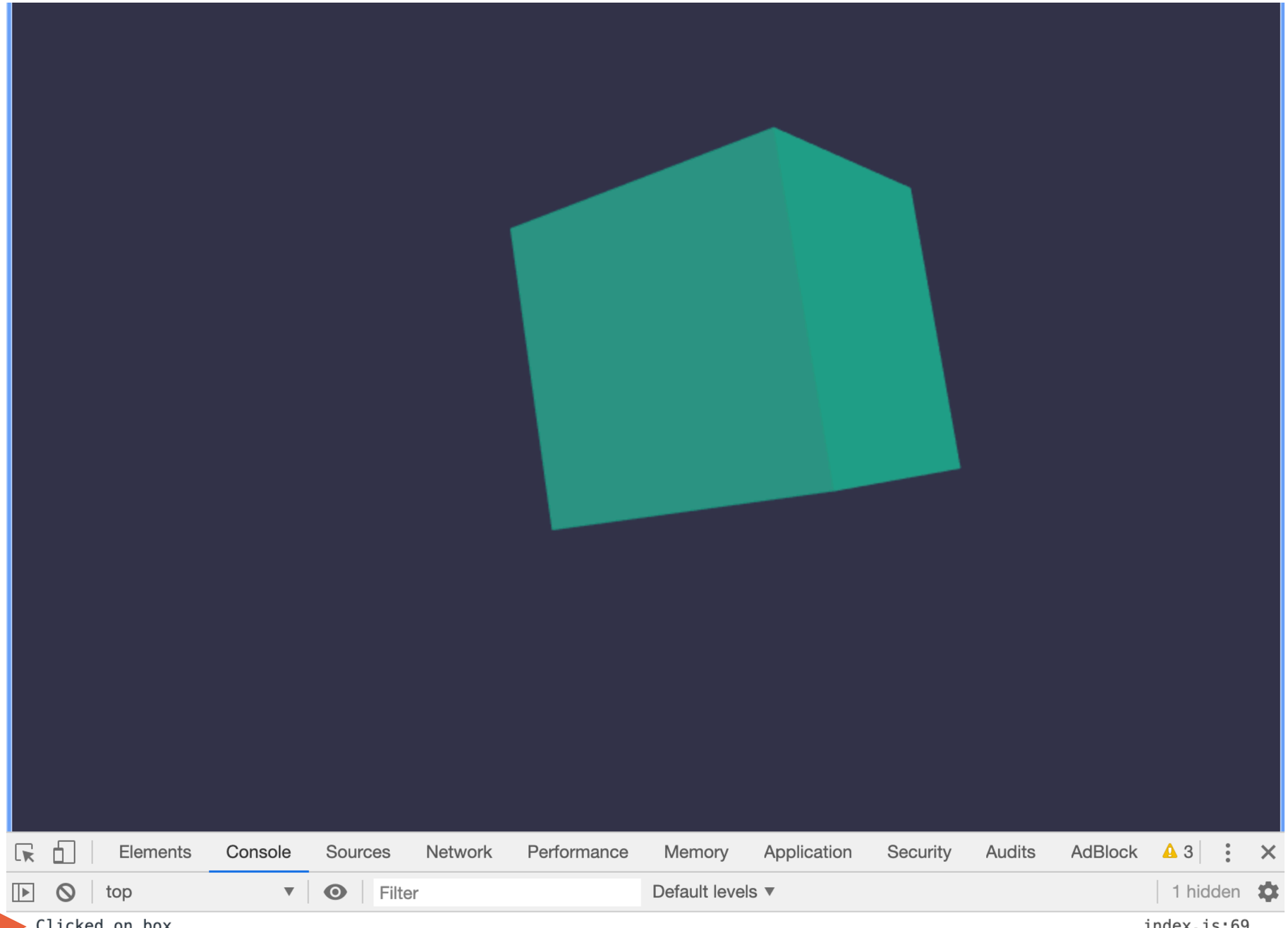
- Set value
- Play sound
- Execute code
- …

Trigger

- Click
- Double-click
- Press key
- …

Example

```
// Adding an action
box.actionManager = new ActionManager(scene);

box.actionManager.registerAction(
    new ExecuteCodeAction(ActionManager.OnPickTrigger, () =>
    {
        console.log("Clicked on box");
        box.material = material;
    })
);
```

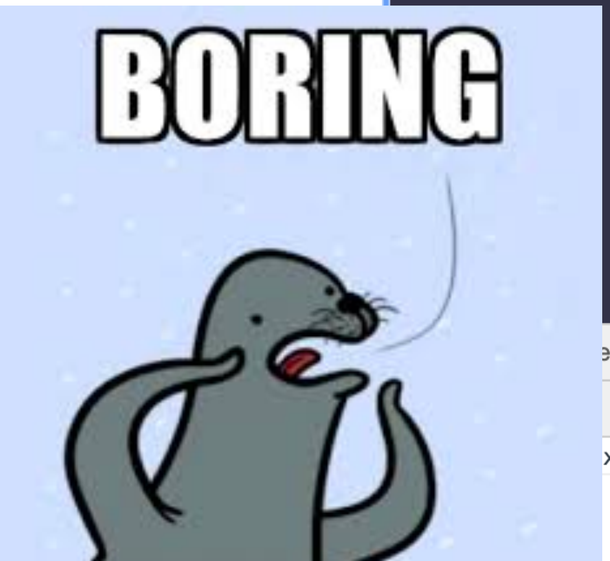

Elements
Console
Sources
Network
Performance
Memory
Application
Security
Audits
AdBlock
3
top
Filter
Default levels
1 hidden
Clicked on box
index.js:69

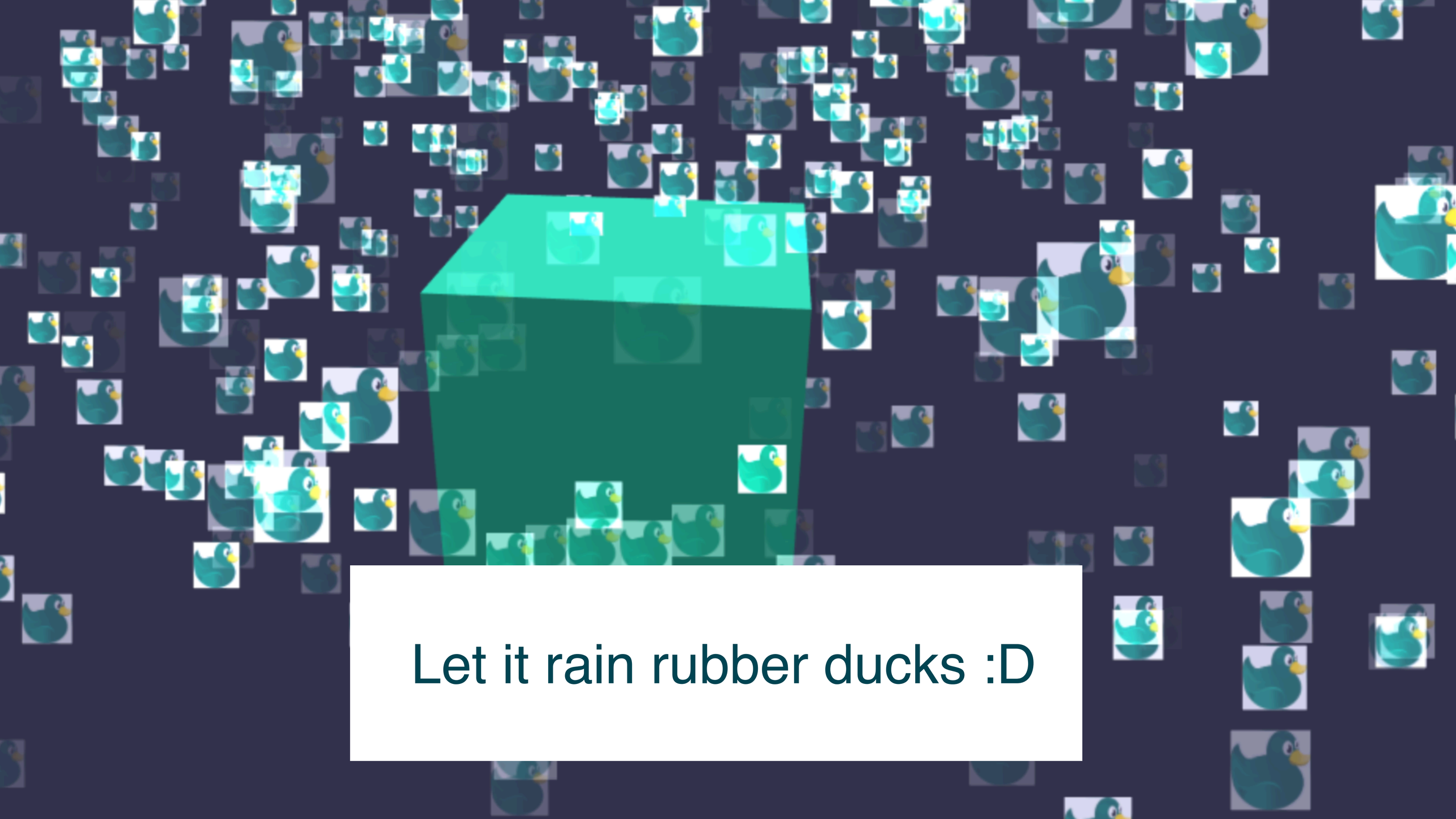
Task 3: Colors :D

Use a **StandardMaterial** to make the box colorful.

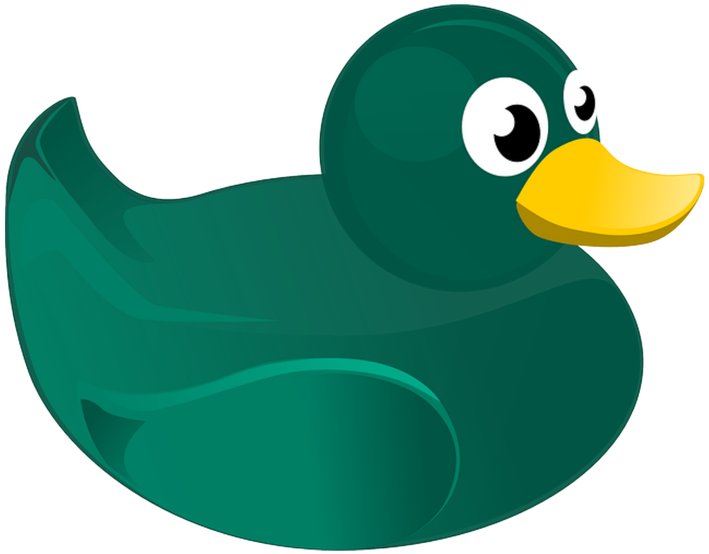
Create an **ExecuteCodeAction** with the **ActionManager** that logs a message on the console when **clicking** on the box.


ents
Console
Sources
Network
Performance
Memory
Application
Security
Audits
AdBlock
3
Filter
Default levels
1 hidden
x
index.js:69


BORING


Let it rain rubber ducks :D

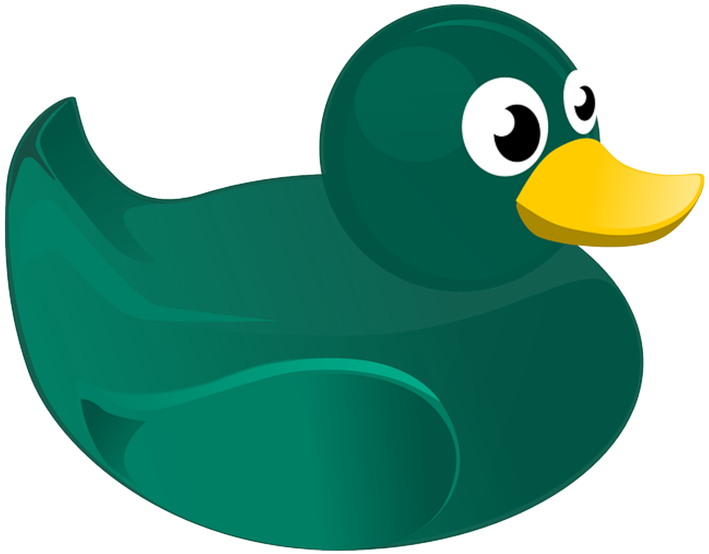
Particle System



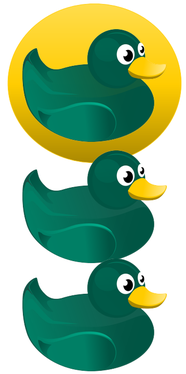
2D Sprite

Emitter

Particle System

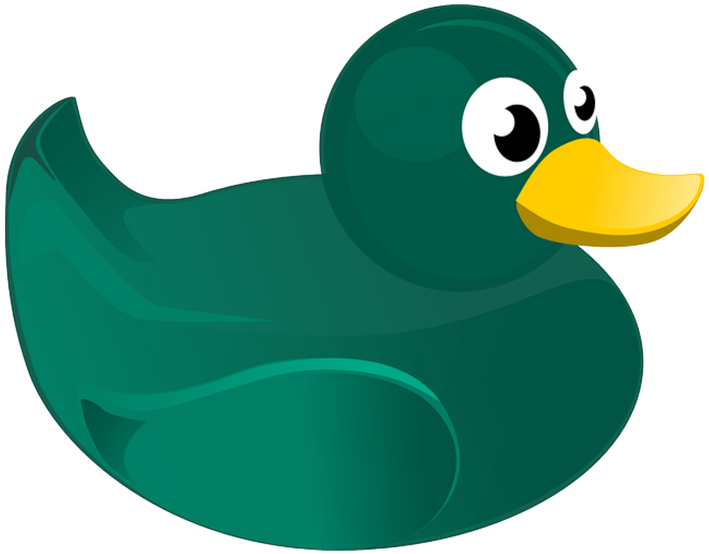


2D Sprite

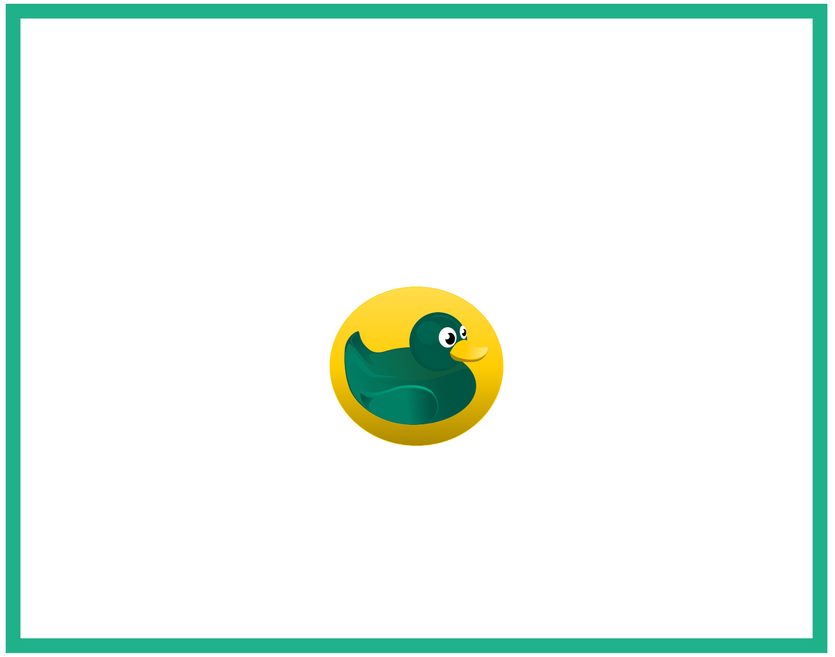


Emitter

Particle System

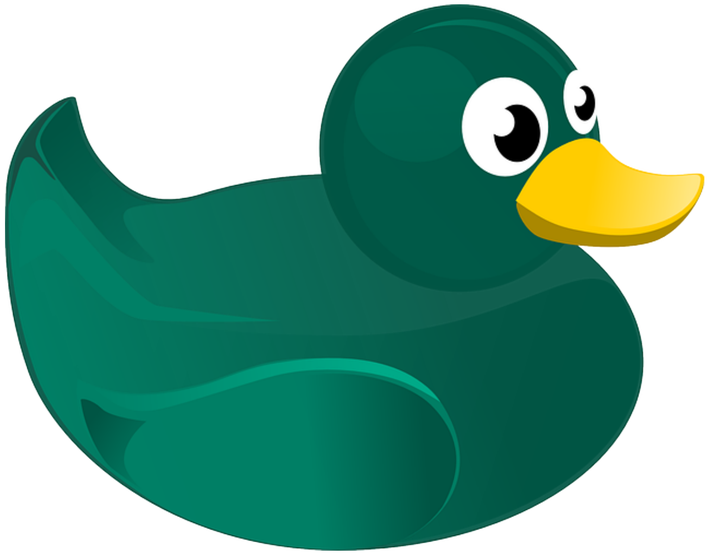


2D Sprite

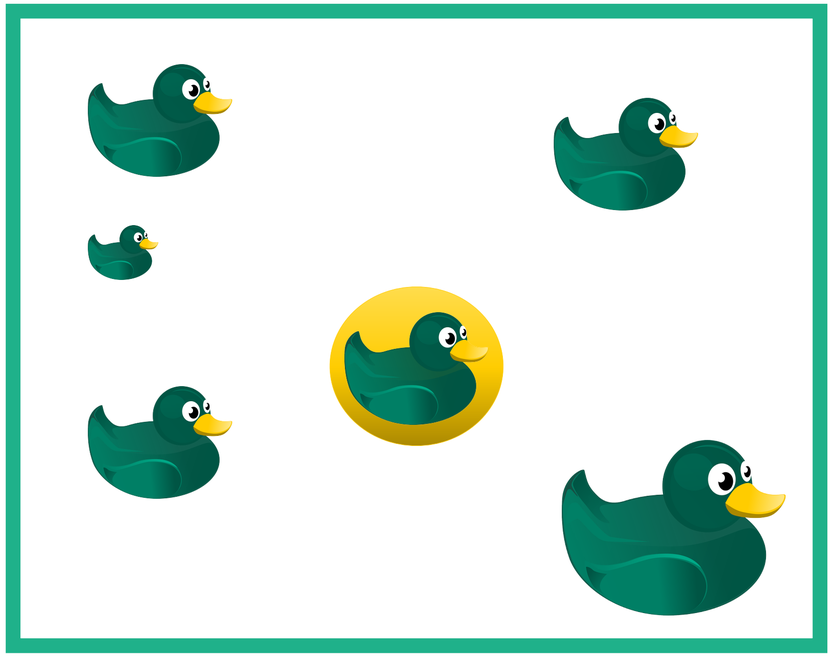


Emitter

Particle System



2D Sprite



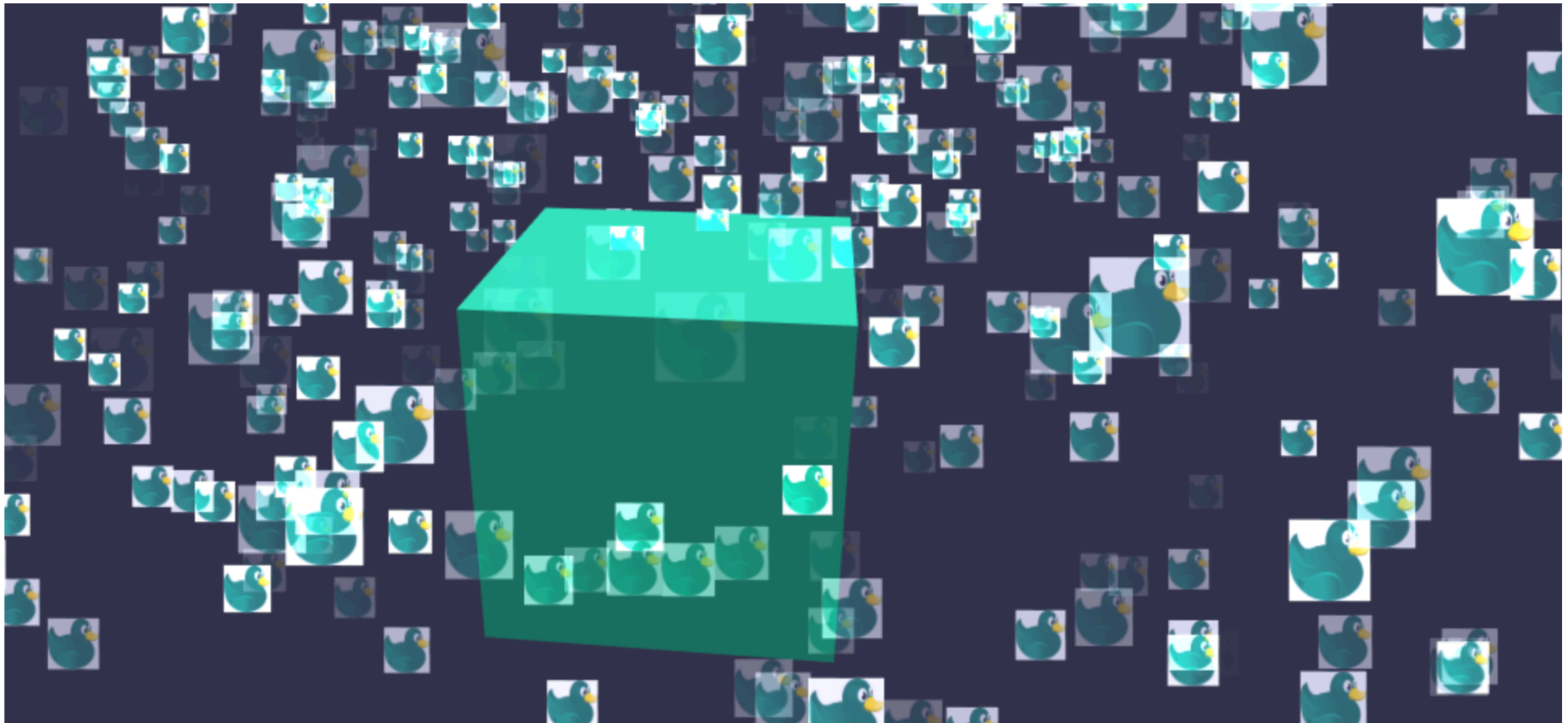
Emitter

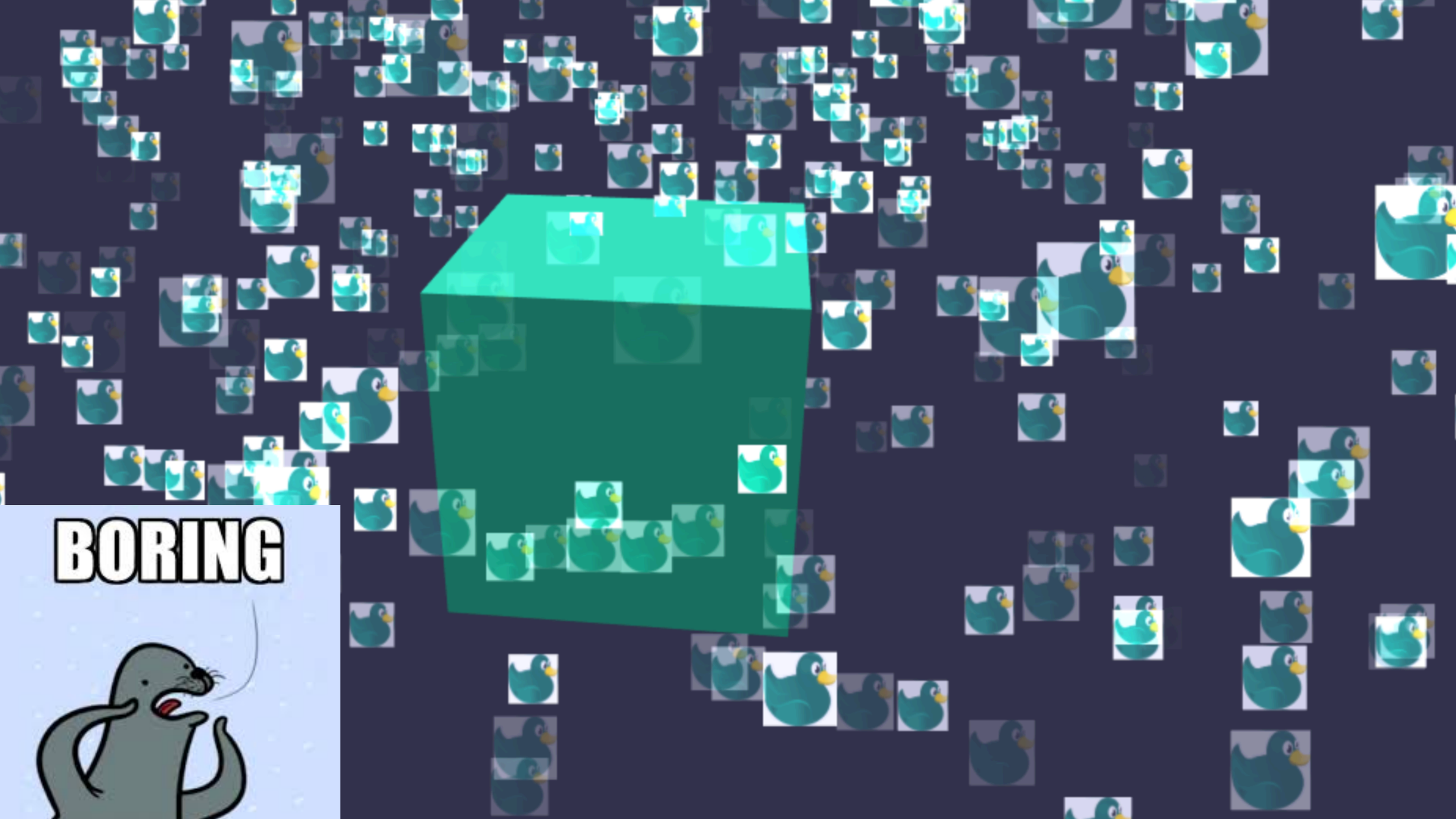
Particle Systems

```
const duckParticleSystem = new ParticleSystem(„ducks", 2000, scene);

duckParticleSystem.particleTexture = new Texture("/assets/duck.png", scene);
duckParticleSystem.emitter = new Vector3(0, 20, 0);
duckParticleSystem.minEmitBox = new Vector3(-20, -20, -20);
duckParticleSystem.maxEmitBox = new Vector3(20, 20, 20);
duckParticleSystem.gravity = new Vector3(0, -10, 0);
duckParticleSystem.emitRate = 999;

duckParticleSystem.start();
```

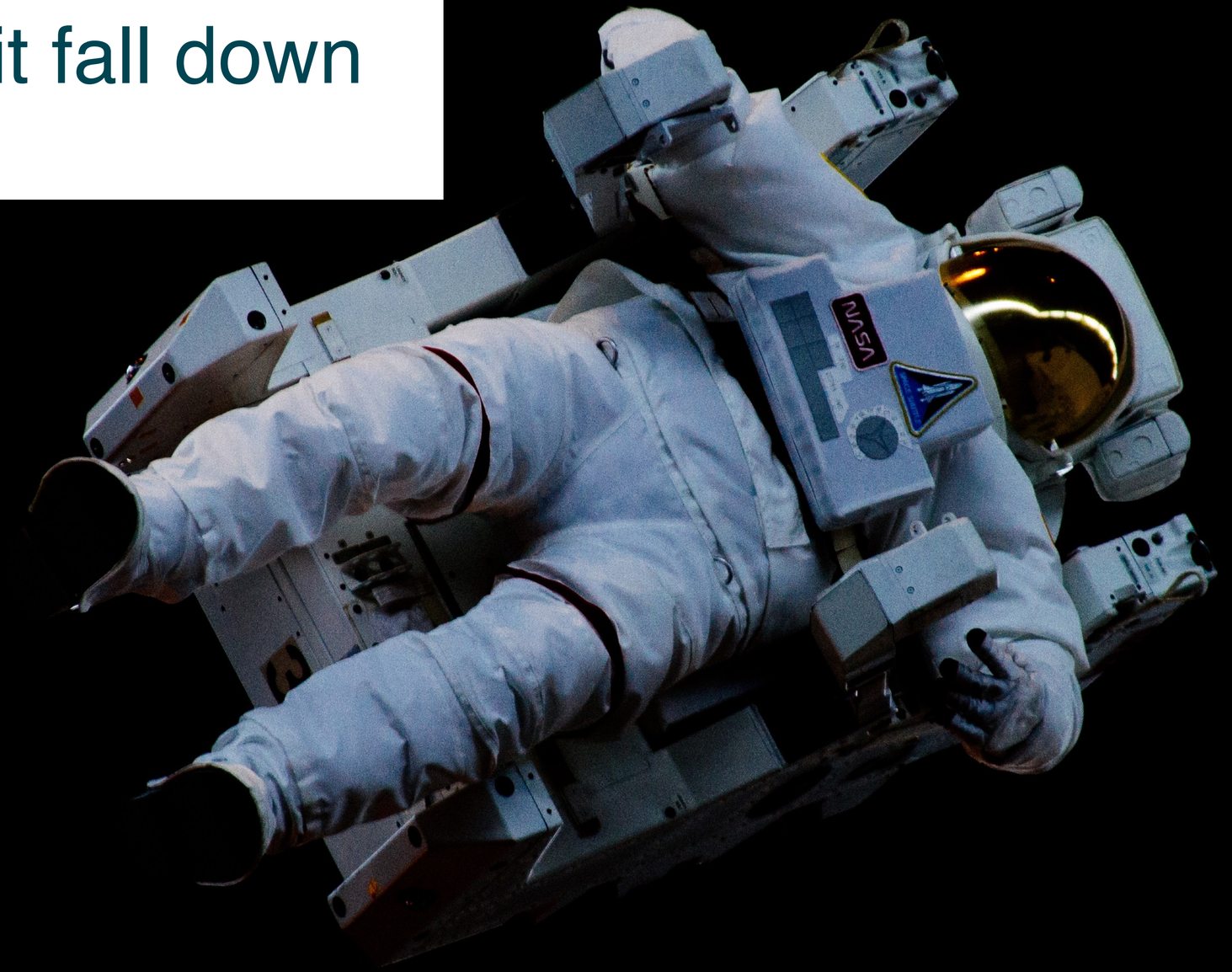
Task 4: Rain

Use the **ParticleSystem** with a **Texture** of your choice.




BORING

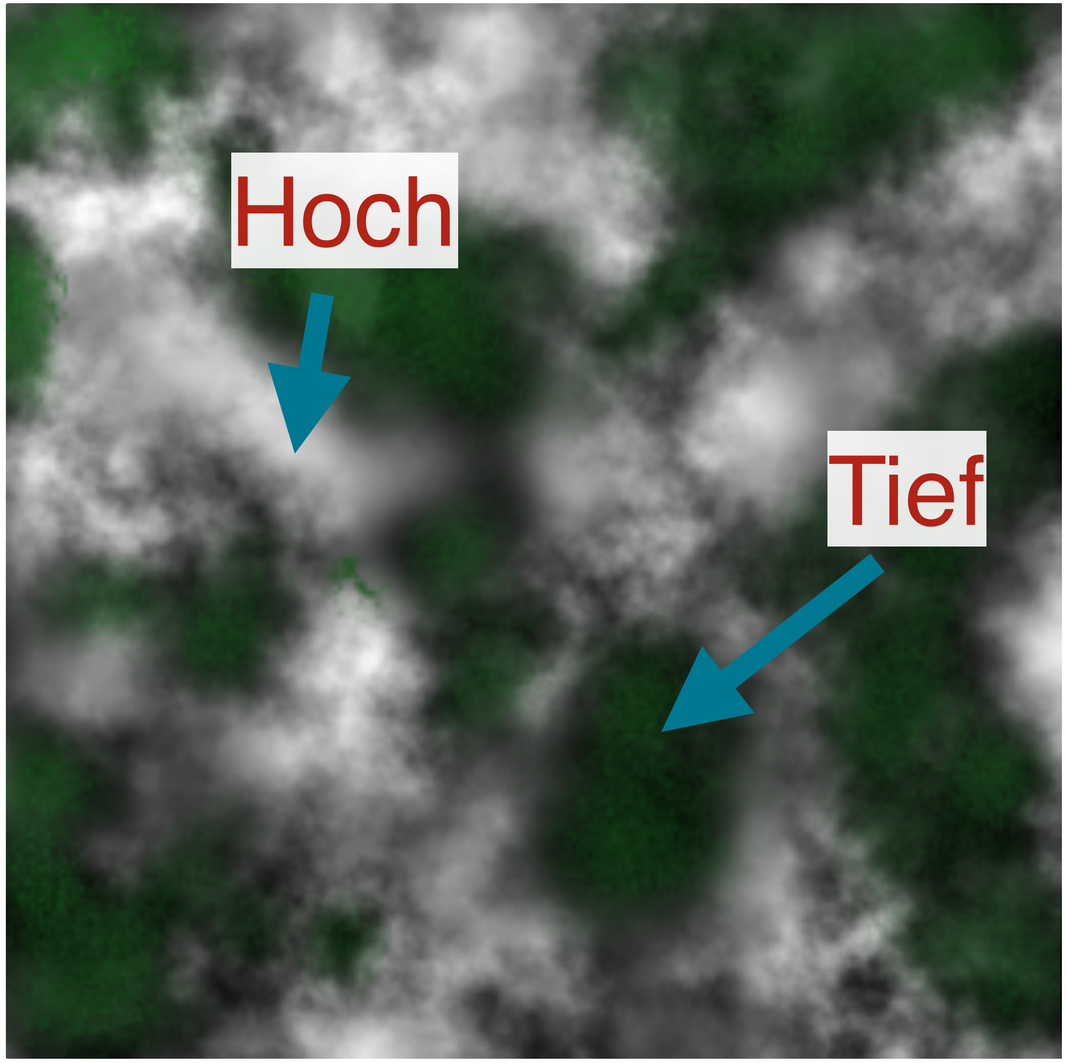
Let it fall down

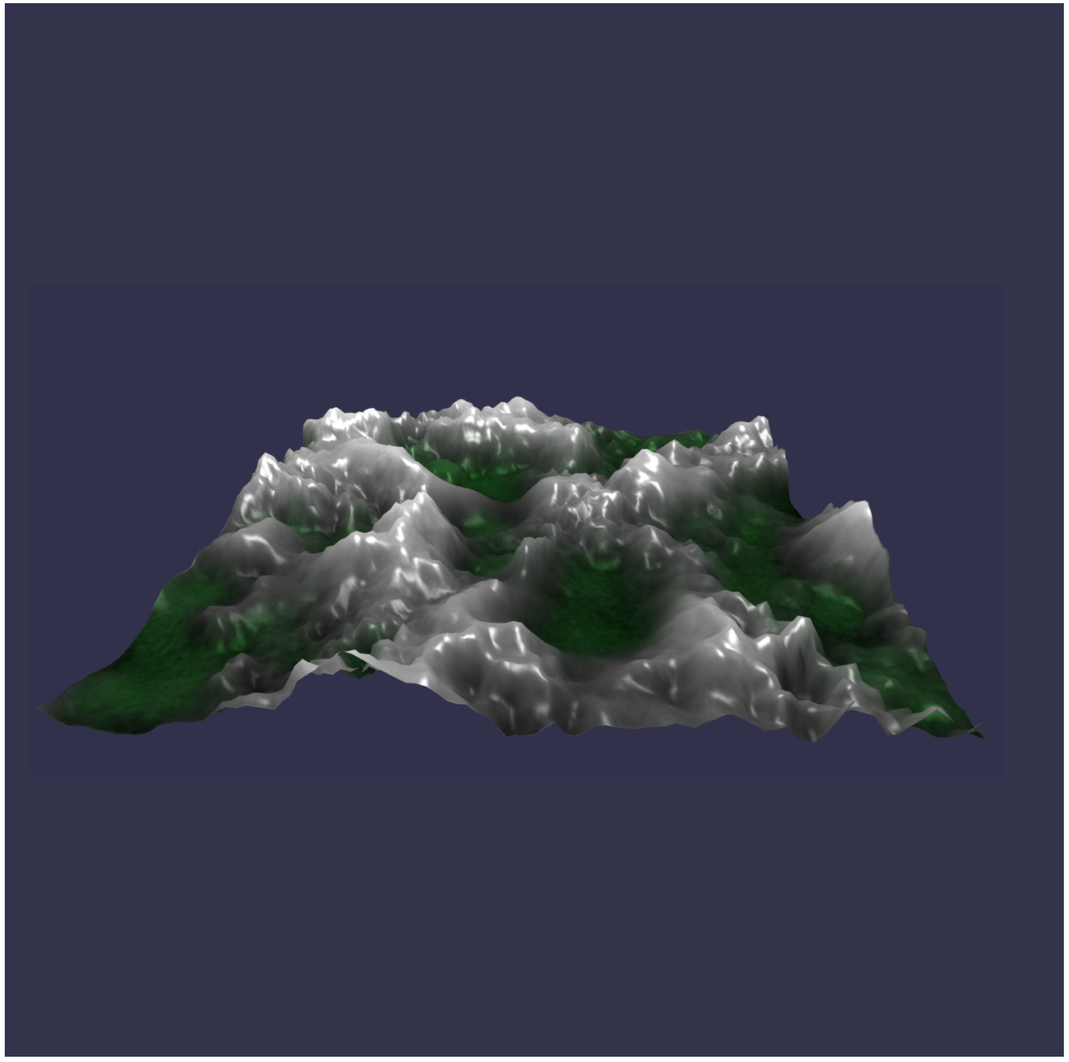


Gravity

```
scene.enablePhysics();

sphere.physicsImpostor = new PhysicsImpostor(
  sphere,
  PhysicsImpostor.SphereImpostor,
  { mass: 2, restitution: 0.9 },
  scene
);

groundPlane.physicsImpostor = new PhysicsImpostor(
  groundPlane,
  PhysicsImpostor.PlaneImpostor,
  { mass: 0, restitution: 0.1 },
  scene
);
```

Height Map





Height Map / Ground

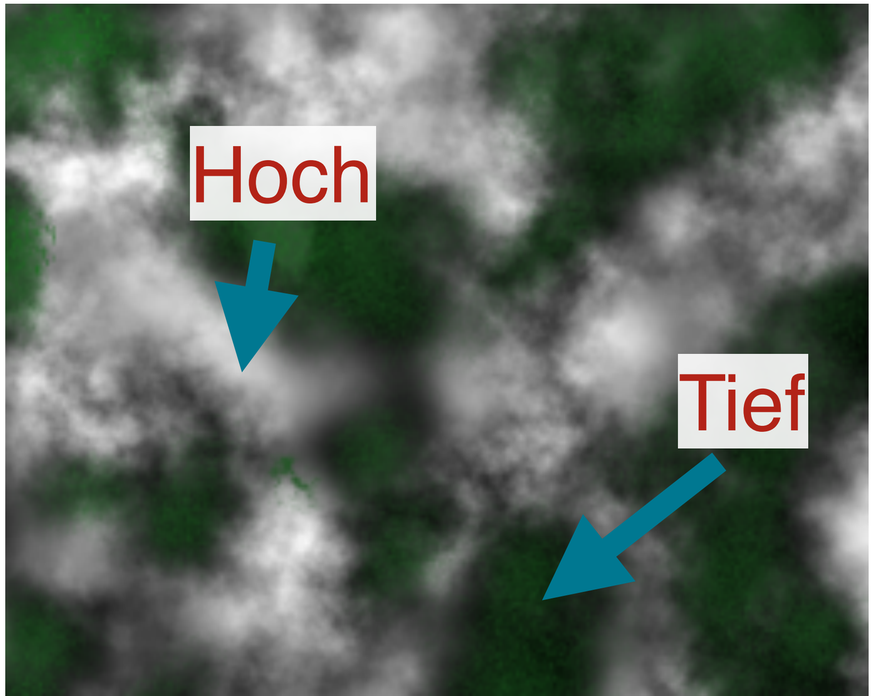
```
const ground = new Mesh.CreateGroundFromHeightMap(
  "ground",
  "assets/Heightmap.png",
  400, //width
  400, //height
  100, //subdivide
  -5, //min height
  50, // max height
  scene,
  false, //updatable
  () => {
    //function on success
  }
);
```

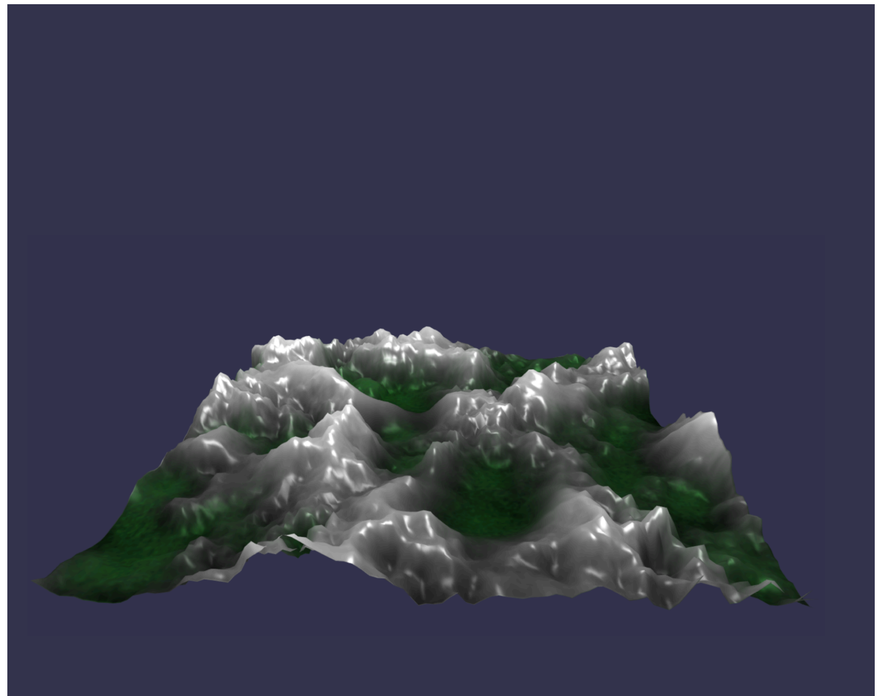
Task 5: A whole new world

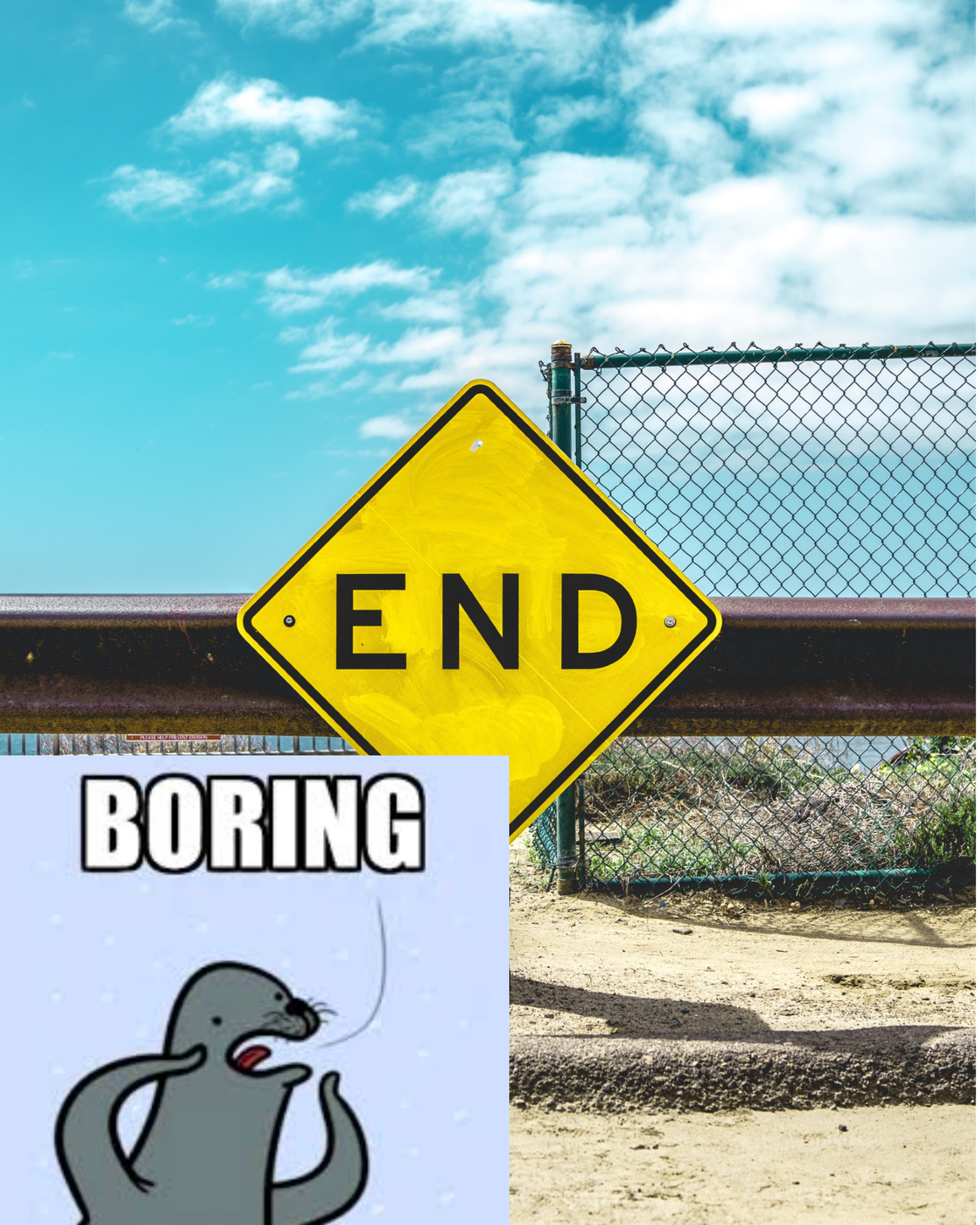
Create a ground from a **HeightMap** using the **MeshBuilder**.

Activate gravity.

Create a sphere that falls down using the **PhysicsImposter**.







…of the boring part

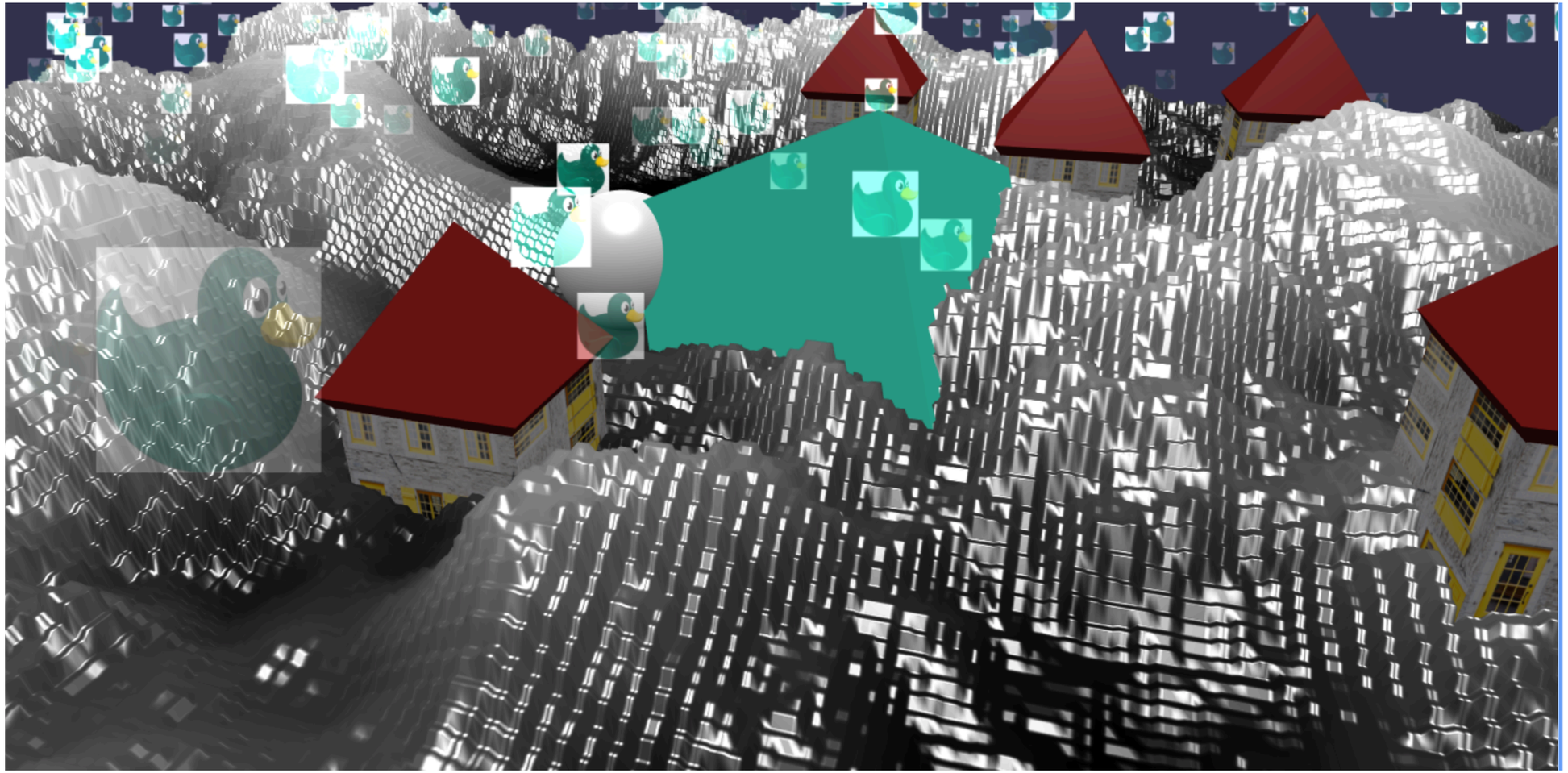
…of the
~~boring part~~
tutorial part…

…and start of the fun part :D

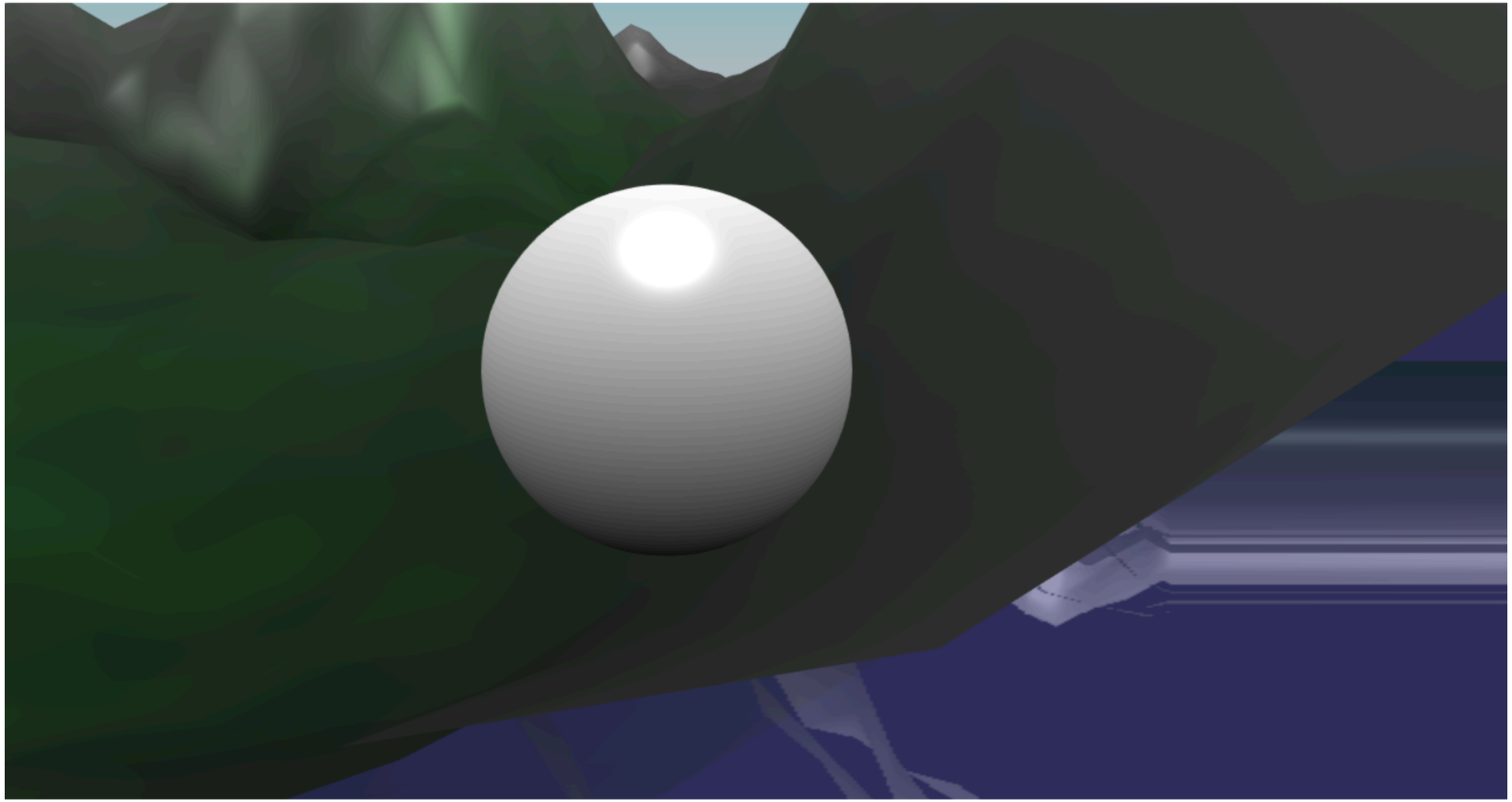
Your idea!

We help you build it.

Inspiration



Inspiration (aka Live Demo)



QUESTIONS?

COME TALK TO US! :)

CHRISTINA

merelyChristina

ANNA

merelyAnna