

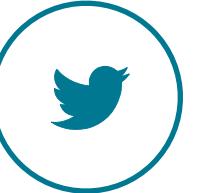
3D im Browser

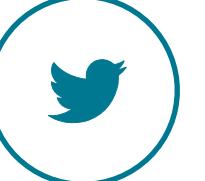
Eine Einführung in babylon.js

@merelyChristina & @merelyAnna

Who are we?



CHRISTINA
 [merelyChristina](https://twitter.com/merelyChristina)

ANNA
 [merelyAnna](https://twitter.com/merelyAnna)



Who are you?

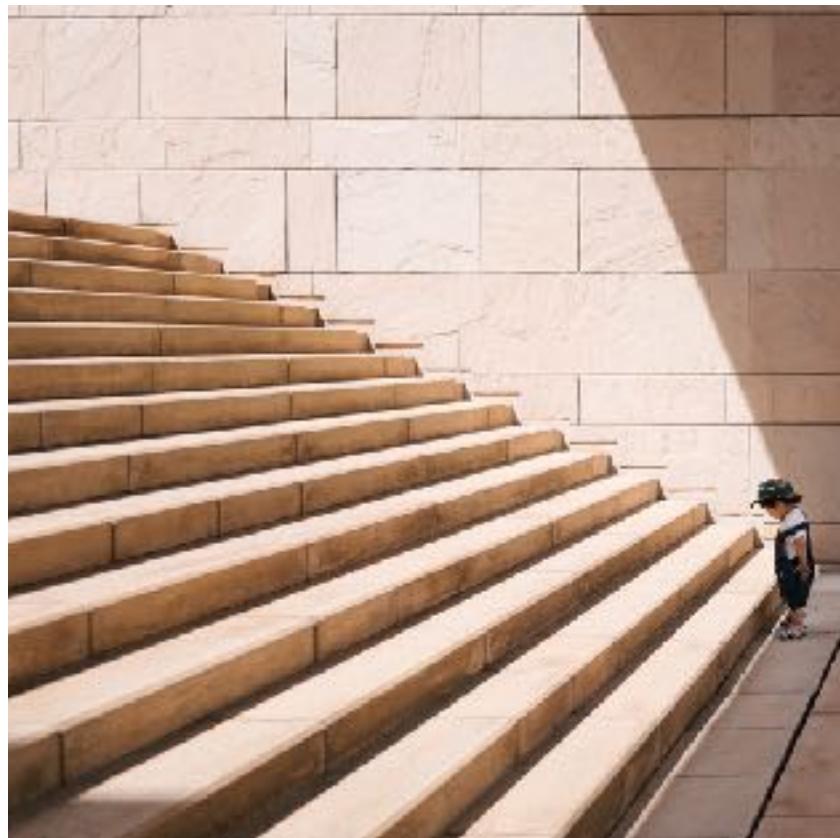


I am Cevapcici Christina :D



I am Ananas Anna :D

What are we doing today?



3D (Duh...)

Basics

↖(ツ)↗

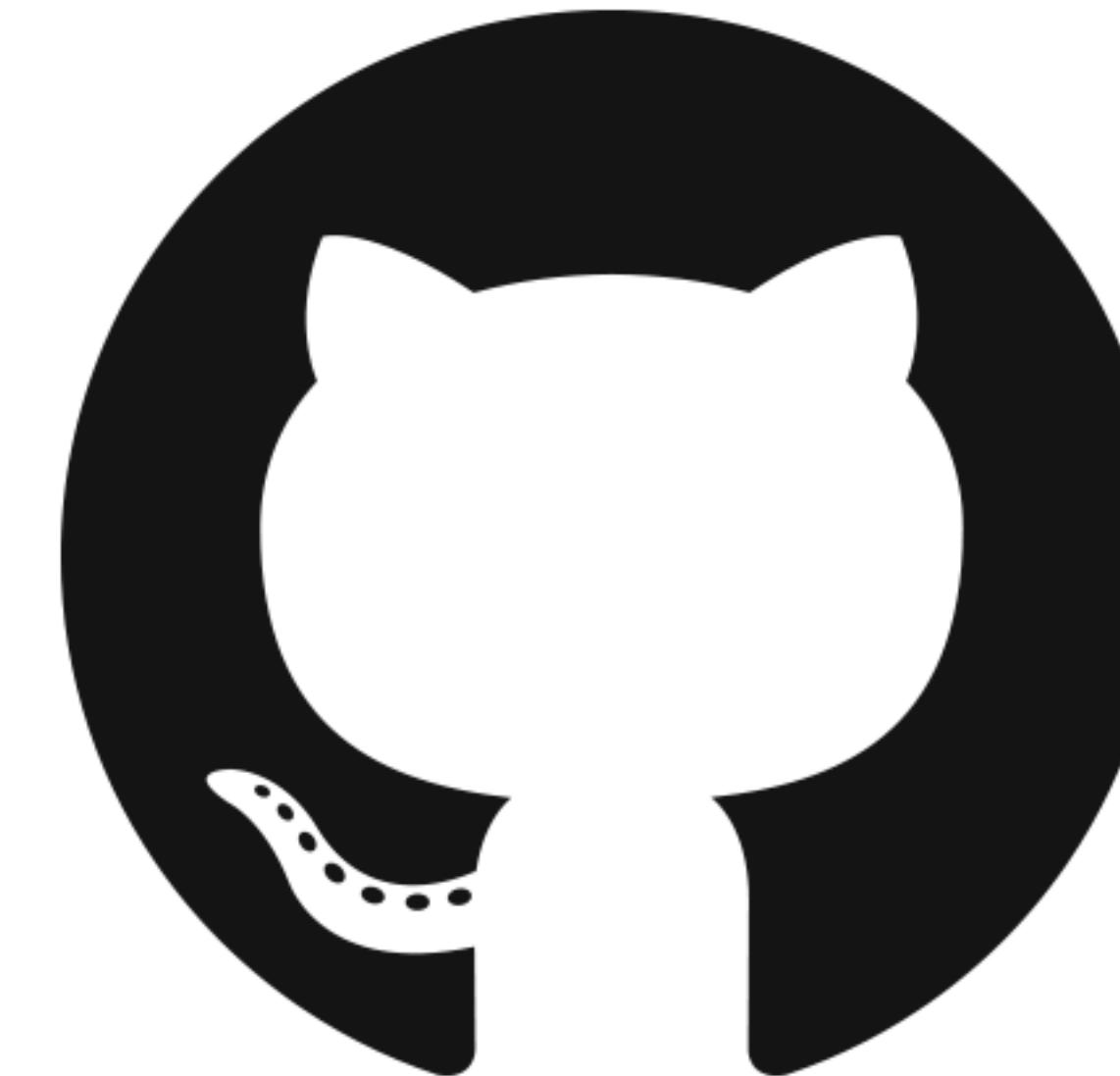
Programming experience?
Experience with
JavaScript/TypeScript?



Experience with git?



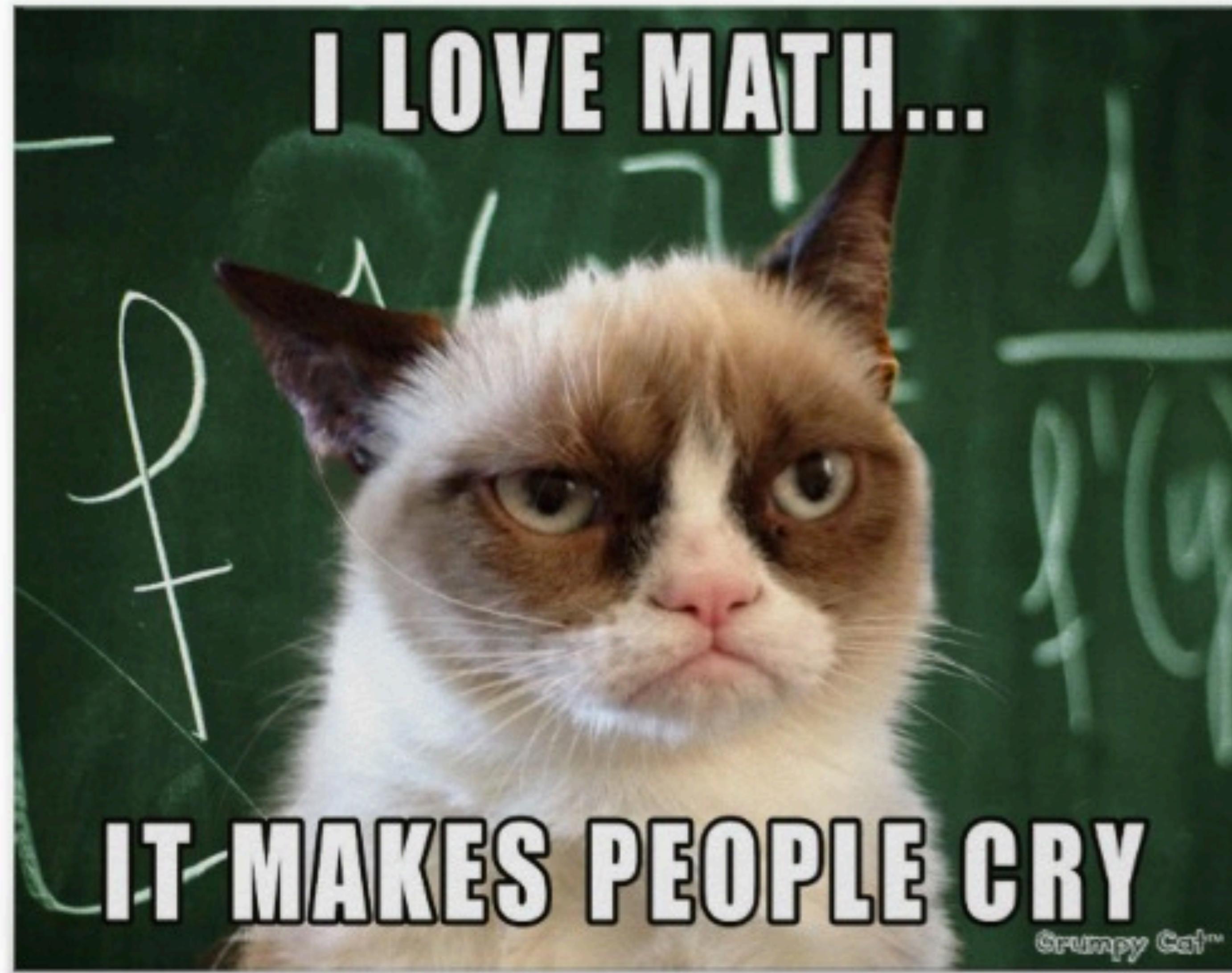
Access to the github repo?



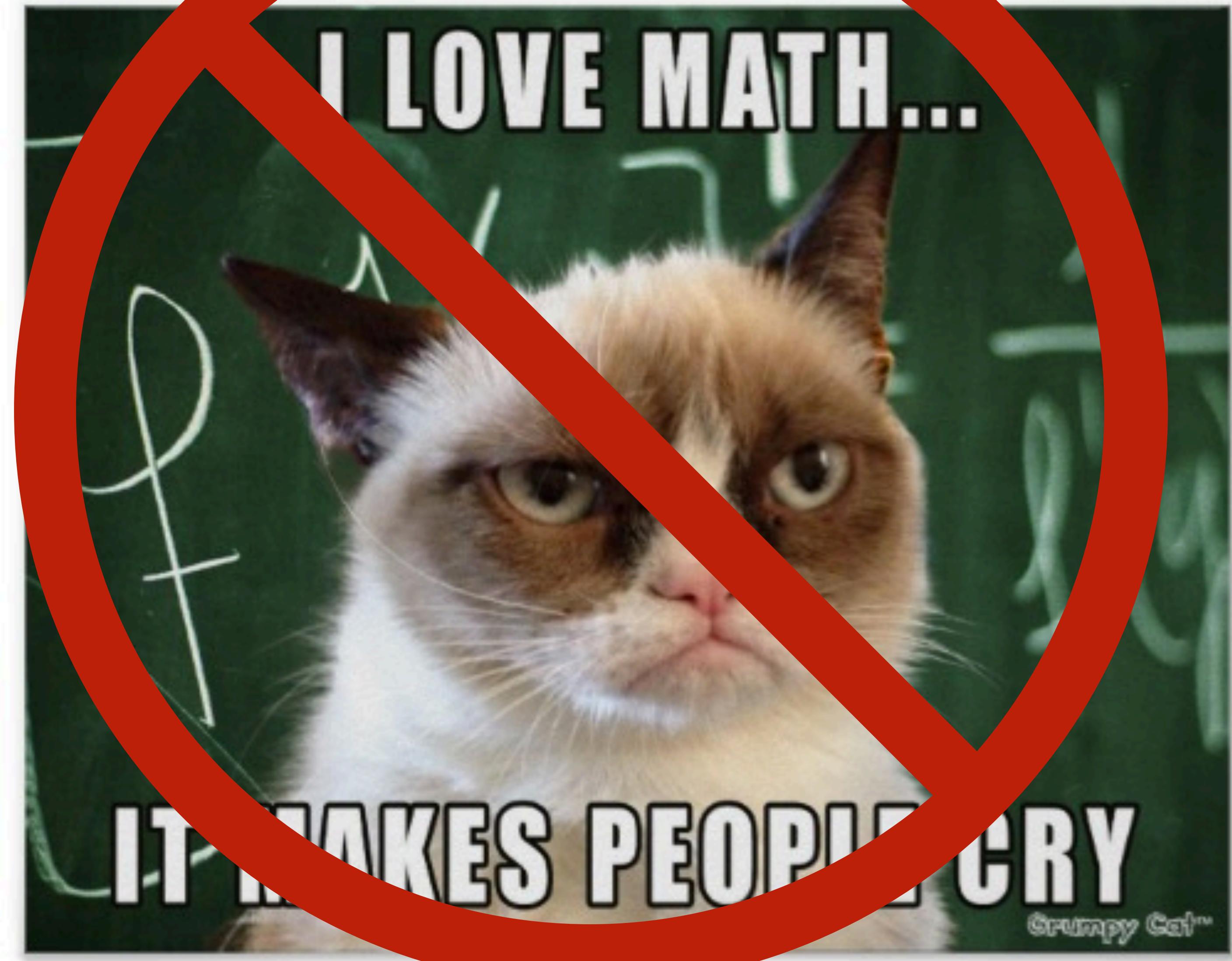
<https://github.com/merelyAnna/spartakiade-babylonJSworkshop>

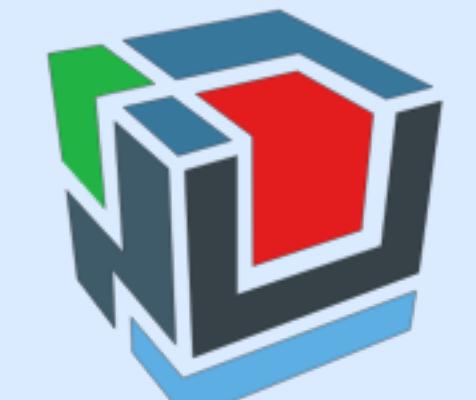
OR <https://bit.ly/2Gzep1l>

Experience with maths?



Experience with maths?

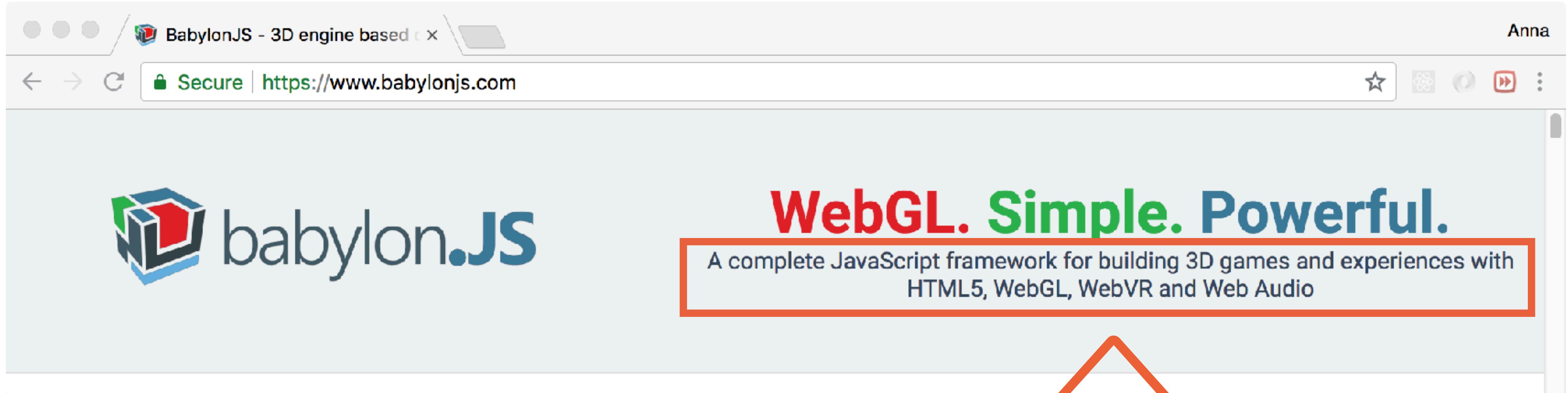




babylon.JS

What is it?

What is babylon.JS?



What is babylon.JS?

3D game engine
running in a browser

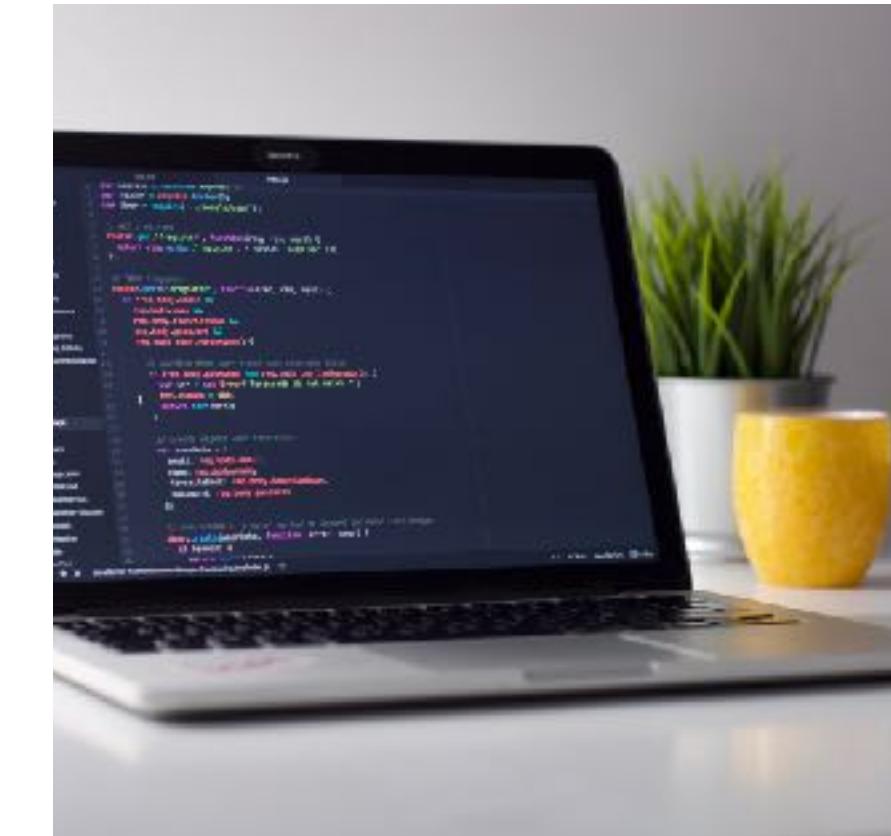
What can it do?



Game Engine



Plattform



Runs in a browser

What else can it do?



Animation

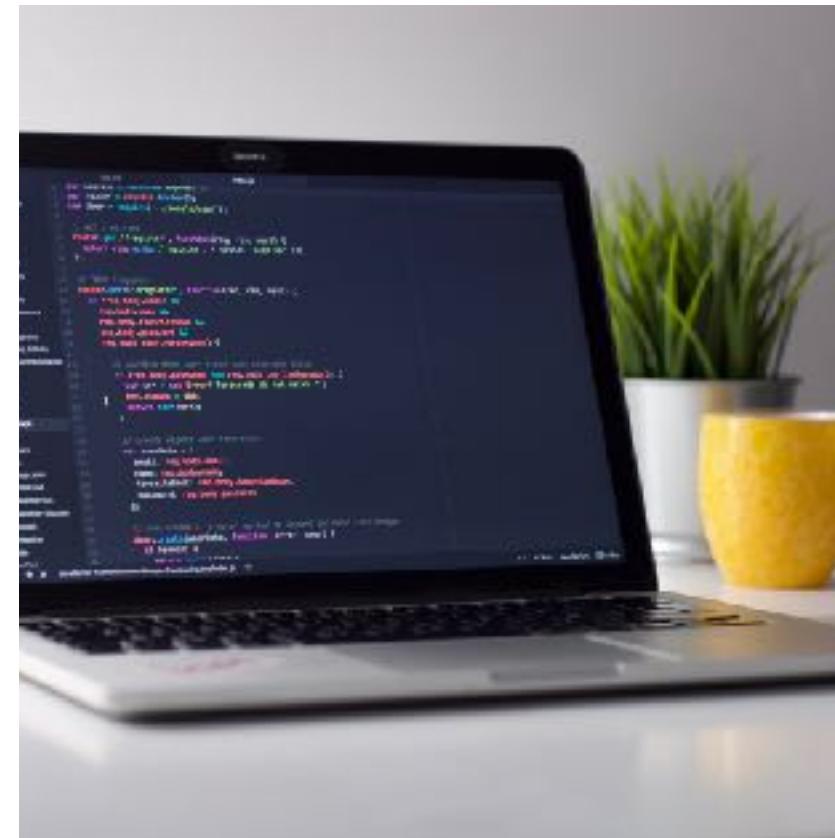
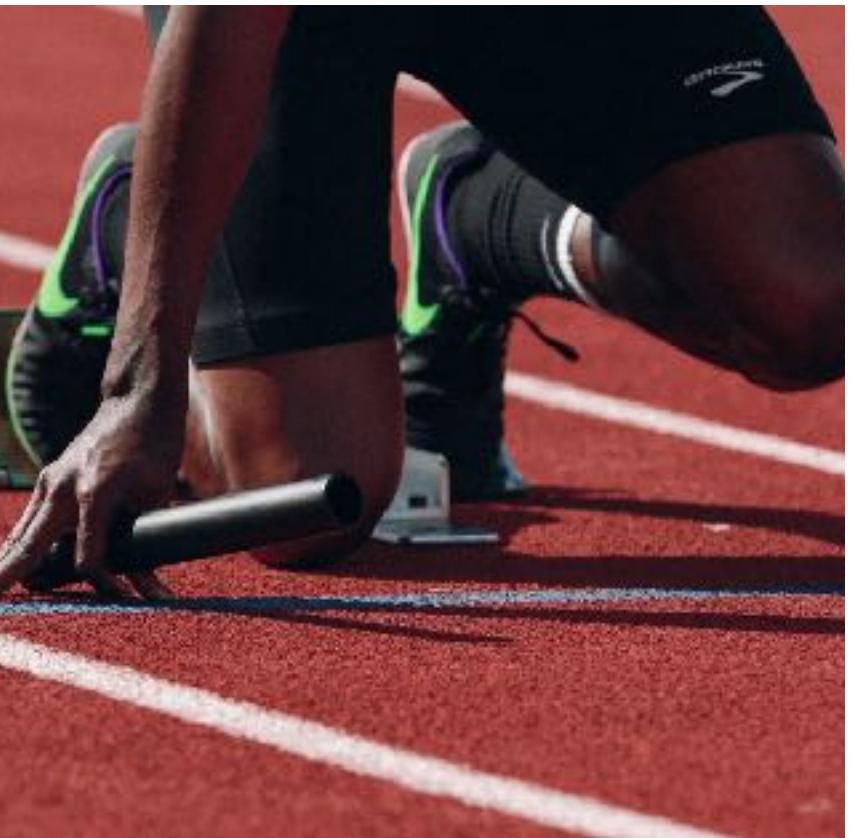


Special FX



Debug Layer

Why use it?

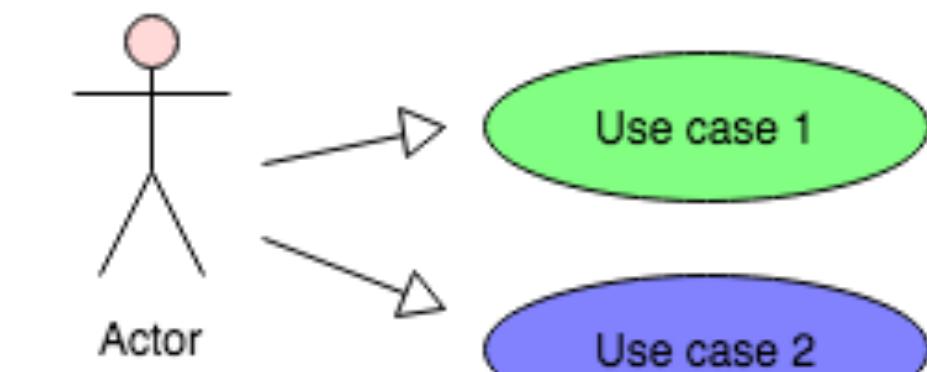


Getting started

Runs in a browser

Extensive

Why not use it?



Extensive

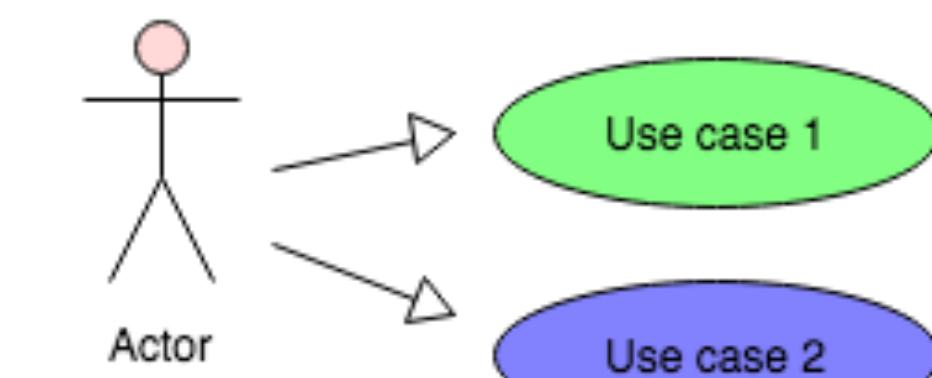
Limits

Use Case

Why not use it?

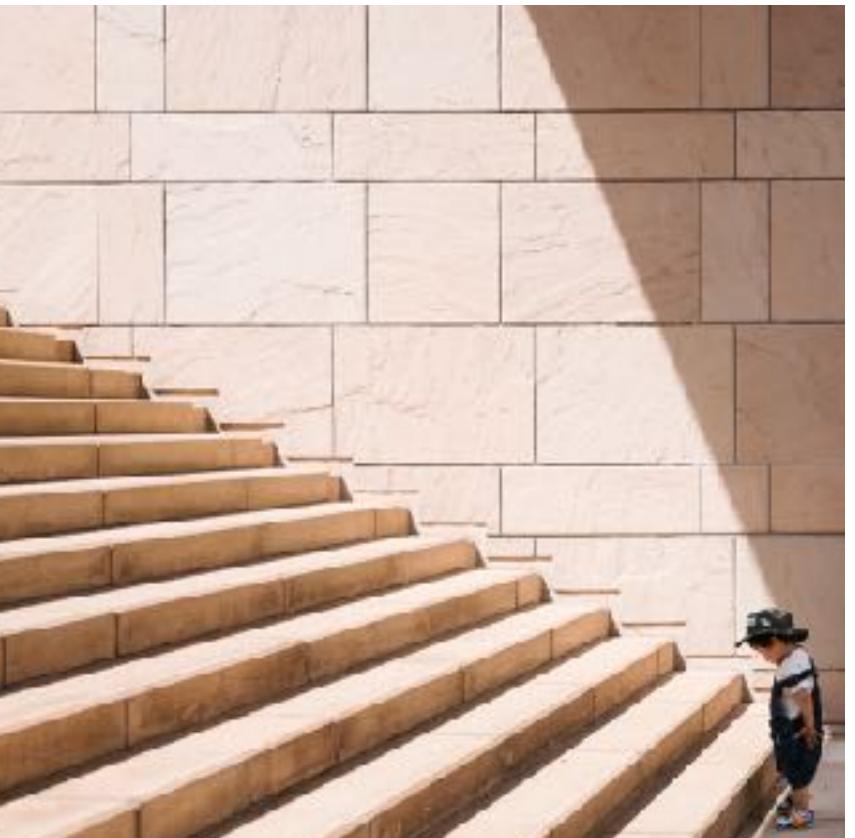


Limits



Use Case

Our experience



Learning curve

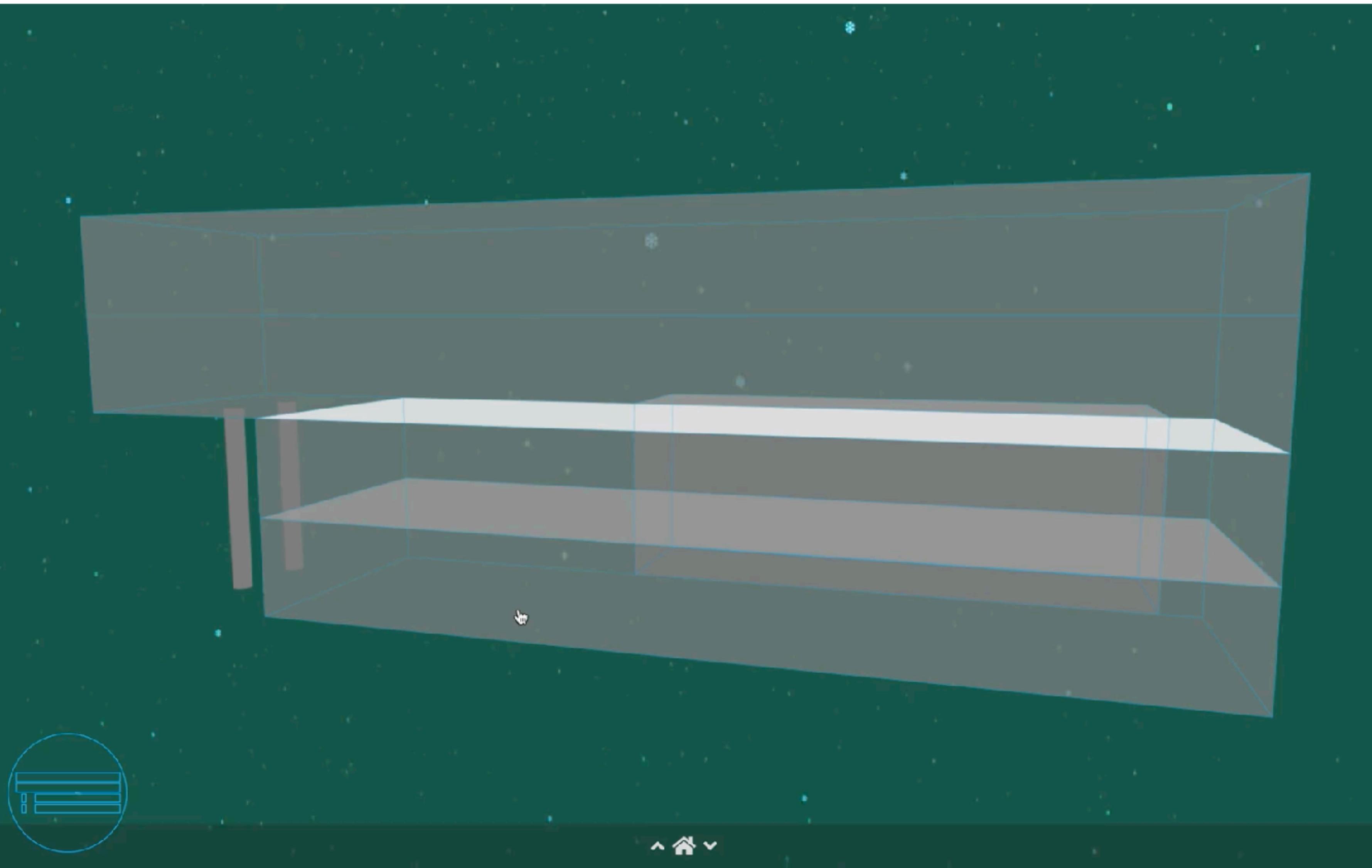


Performance



Community &
Resources

What do we use it for?



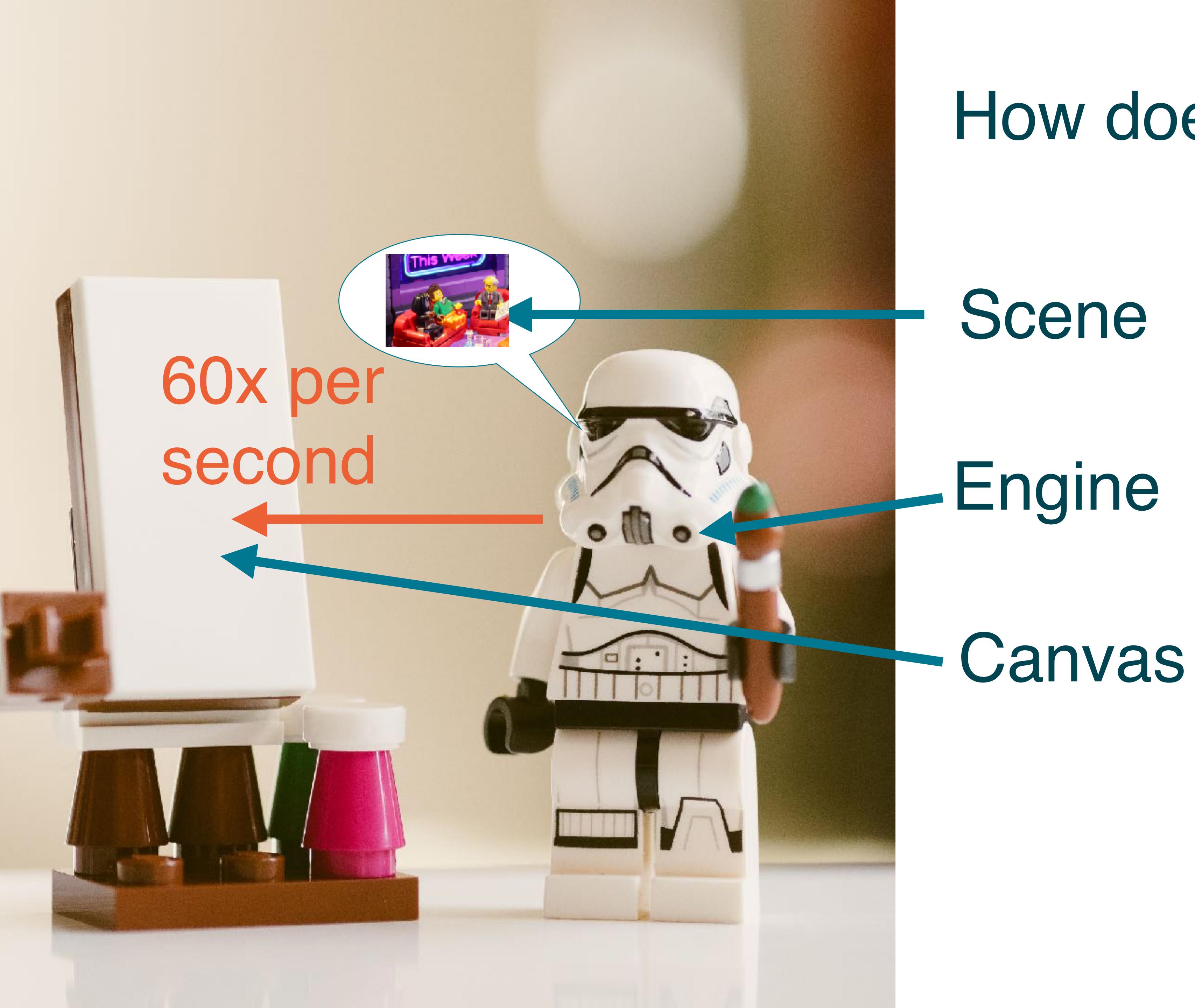
```
index
ngine.supportsUniformBuffers || (b=b.replace(/light\{X\}.(\w*)/g, function(e,t){return replace(a[0],h),a=s.exec(i)}},t.prototype._processPrecision=function(e){return -1==e.indexOf("precision highp float");},t.prototype._highPrecisionShaderSupported?"precision highp float;\n"+e:"precision mediump float;\n"+e:this._engine.getCaps().highPrecisionShadersSupported?"precision highp float; \n"+e:"precision mediump float")),e},t.prototype._rebuildProgram=function(t,i,n,r){var o=this;this._fragmentSourceCodeOverride=i,this.onError=function(e,t){r&&r(t)},this.onCompiled=function(){for(var t=o.getEngine();t;t=markAllMaterialsAsDirty(e.Material.TextureDirtyFlag);n&&n(o._program)},this._fallbacks=null,this._prepareEffect(),t.program=e{return this._engine.getUniforms(this._program,e)},t.prototype._prepareEffect=function(){var t=this._attributesNames,_valueCache={};var r=this._program;try{var o=this._engine;if(this._vertexSourceCodeOverride&&this._fragmentSourceCodeOverride){this._vertexSourceCodeOverride,void 0,this._transformFeedbackVaryings):this._program=this._fragmentSourceCode,i,void 0>this._transformFeedbackVaryings),this._program._SPECTOR_rebuildProgram=this._rebuildProgram(uniformBlock(s,this._uniformBuffersNames[s]));this._uniforms=o.getUniforms();for(a=0;a<this._samplers.length;a++)null==this.getUniform(this._samplers[a])&&(this._isReady=!0,this.onCompiled&&this.onCompiled(this),this.onCompileObservable());this._fallbacks.unBindMesh(),r&&this.getEngine()._deleteProgram(r)}catch(e){"Uniforms:"+this._uniformsNames.map(function(e){return"+e})),e.Tools.Error("Error:"+this._compilationError),this.onErrorObservable.notifyObservers(this)},n&&n(this._prepareEffect()):(this.onError&&this.onError(this,this._compilationError),this._fallbacks&&this._fallbacks.unBindMesh())},Object.defineProperty(r,0,configurable:!0}),t.prototype._bindTexture=function(e,t){this._engine.setTexture(this._samplers.indexOf(e),this.getUniform(e),t)},t.prototype._setTextureFromPostProcess=function(e,t){this._engine.setTextureFromPostProcess(this._valueCache[e],n=t.updateFlag);return(void 0==i||i!=n)&&(this._valueCache[e]=n,i=t)}},t.prototype._cacheFloat2=function(e,t,i){var n=this._valueCache[e];if(!n)return n=[t,i],this._valueCache[e]=n,!0;var r=!1;return r},t.prototype._cacheFloat3=function(e,t,i,n){var r=this._valueCache[e];if(!r)return r=[t,i,n],this._valueCache[e]=r,!1;var s=r[1]!==i&&(r[1]==i,o==!0),r[2]!==n&&(r[2]==n,o==!0),o},t.prototype._cacheFloat4=function(e,t,i,n,r){var o=this._valueCache[e];if(!o)o=[t,s==!0],this._valueCache[e]=o,!0;var s=!1;return o[0]!==t&&(o[0]==t,s==!0),o[1]!==i&&(o[1]==i,s==!0),o[2]!==n&&(o[2]==n,s==!0),o[3]!==r&&(o[3]==r,s==!0)},t.bindUniformBuffer=function(e,i){var n=this._uniformBuffersNames[i];void 0==n&&t._baseCache[n]==e&&(t._baseCache[n]=e)},t.prototype.bindUniformBlock=function(e,t){this._engine.bindUniformBlock(this._program,e,t)},t.prototype.setInt=function(e,t){this._engine.setInt(this._program,e,t)},t.prototype.setFloat=function(e,t){this._engine.setFloat(this._program,e,t)},t.prototype.setVector3=function(e,t){this._engine.setVector3(this._program,e,t)},t.prototype.setMatrix=function(e,t){this._engine.setMatrix(this._program,e,t)}
```



babylon.js

How does it work?

How does it work?



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.



A Playmobil figure with a brown fedora-style hat and a green vest over a white shirt with a red and green camouflage pattern on the shoulders. The figure is holding a large blue telescope with a tripod. The background is a blurred green field.

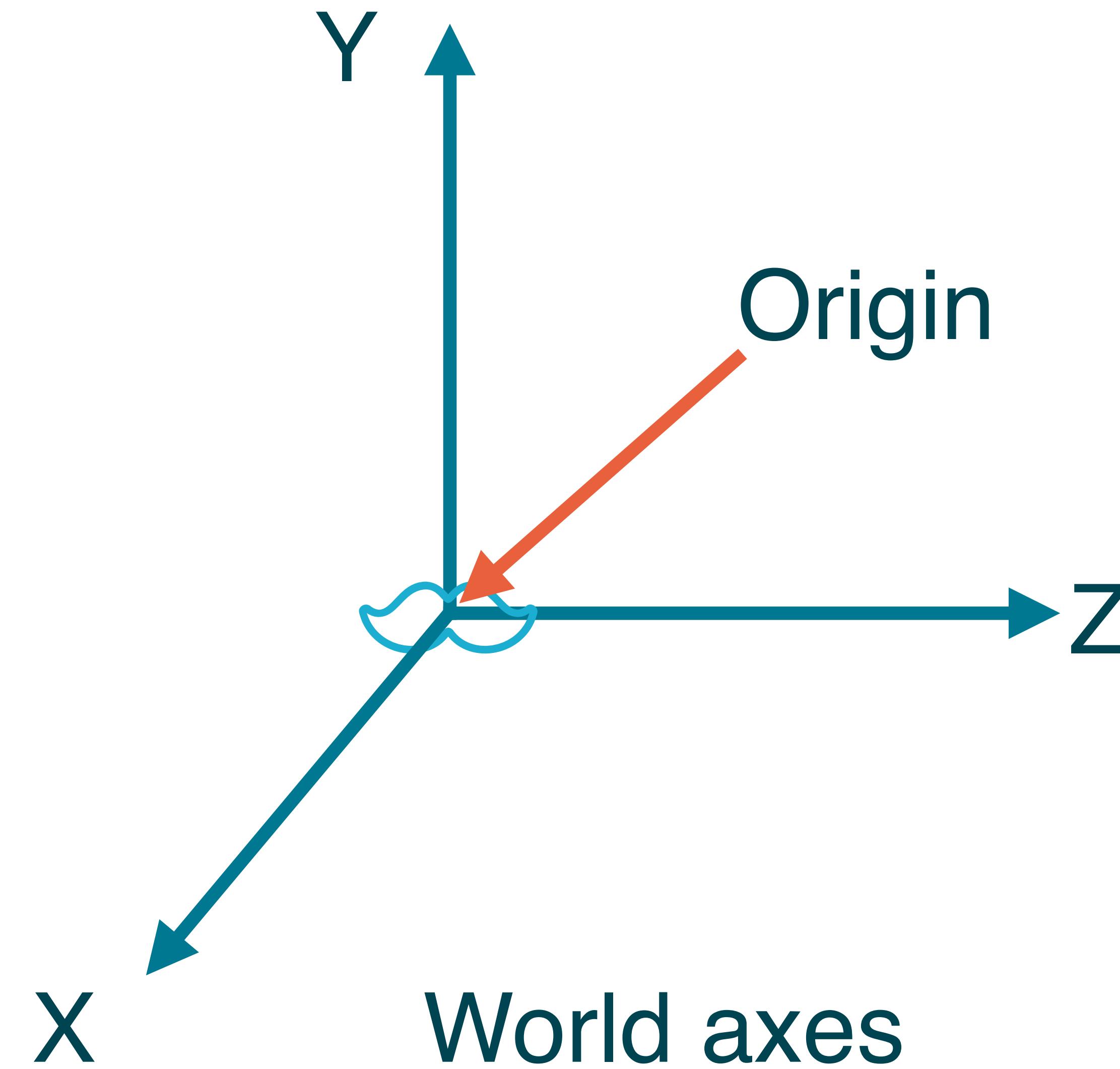
I want to see
something
on the canvas

Parameters

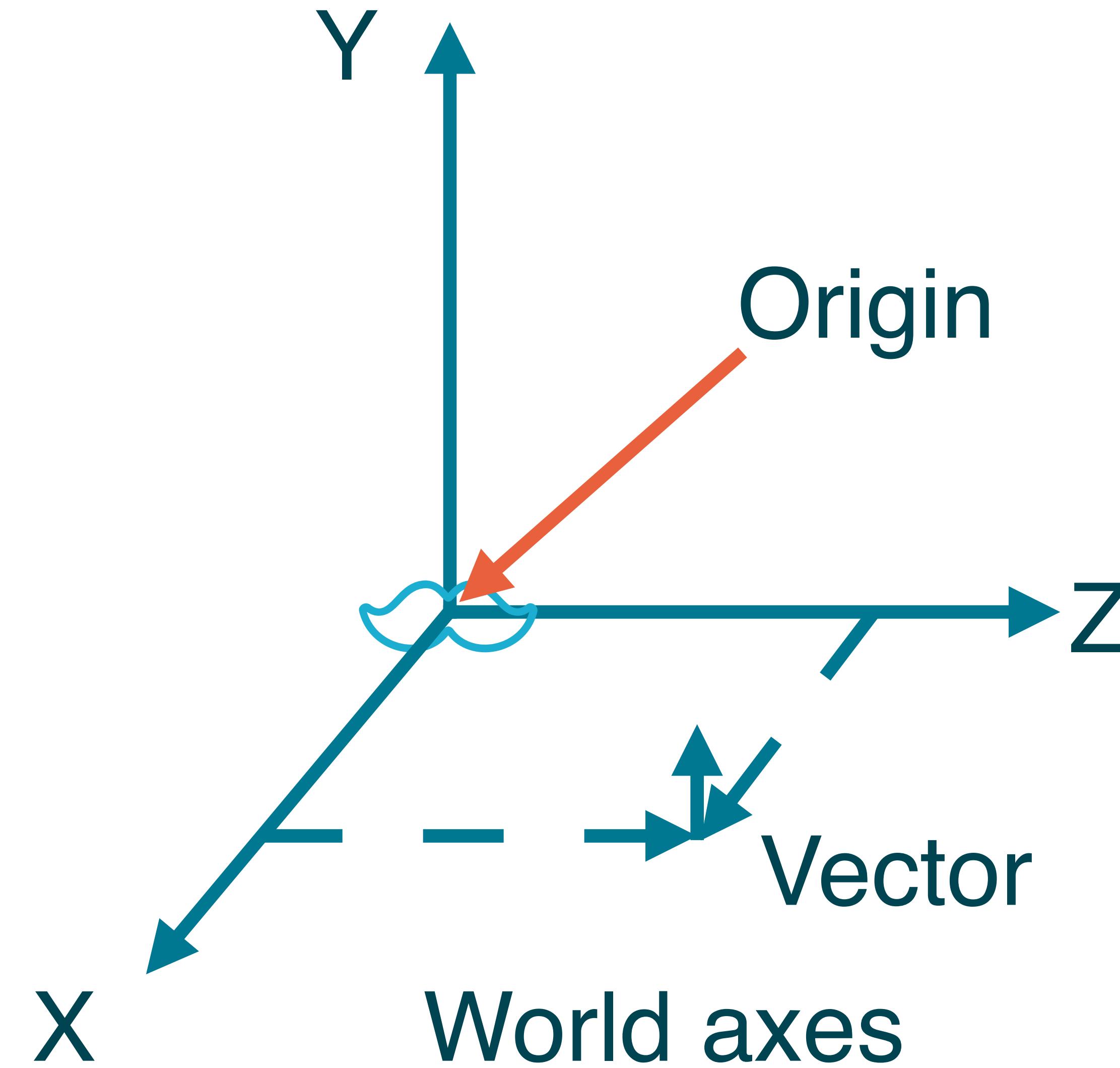
```
const thing = new Thing("name", {}, scene);
```

class name options scene

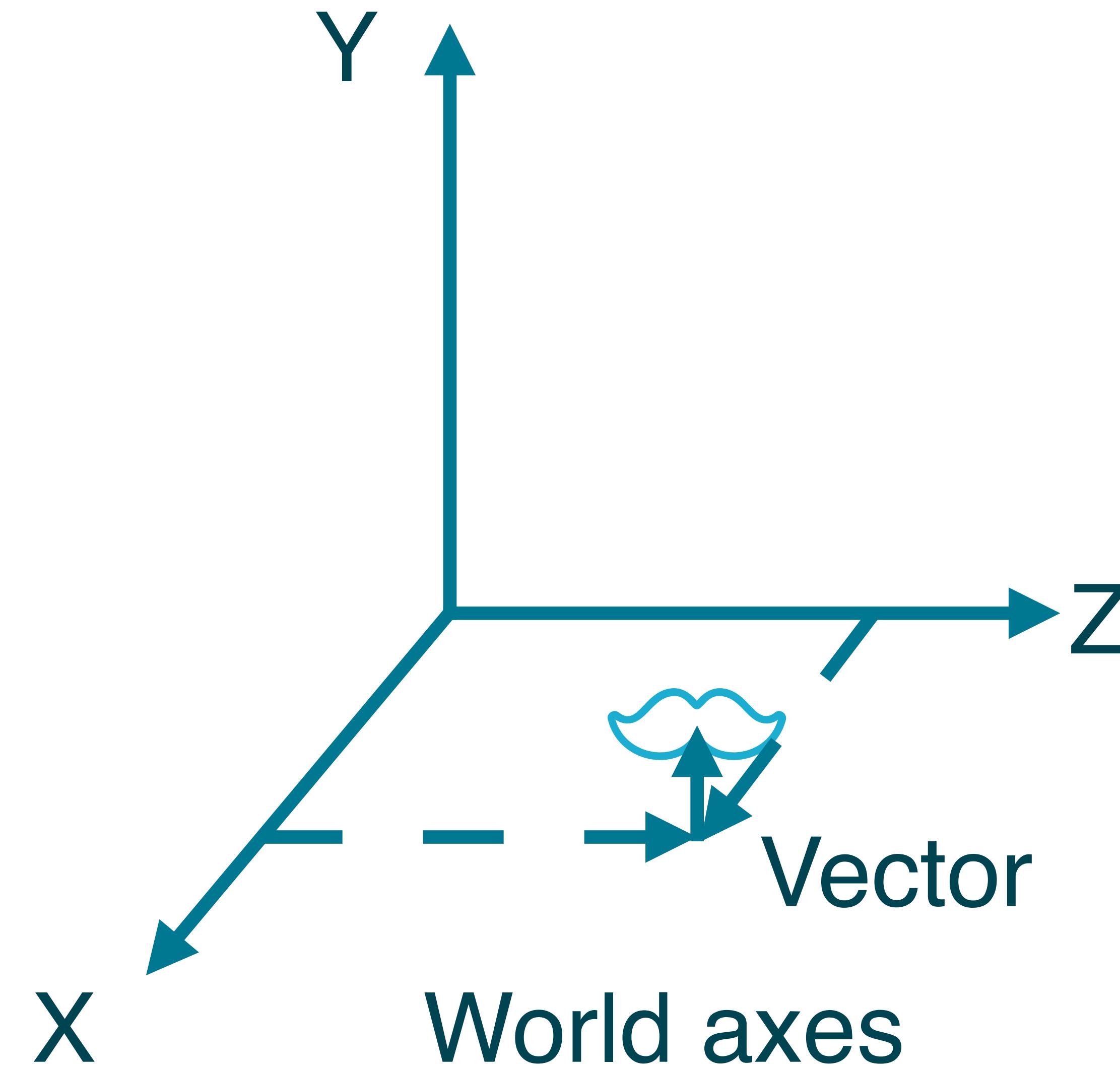
Positioning



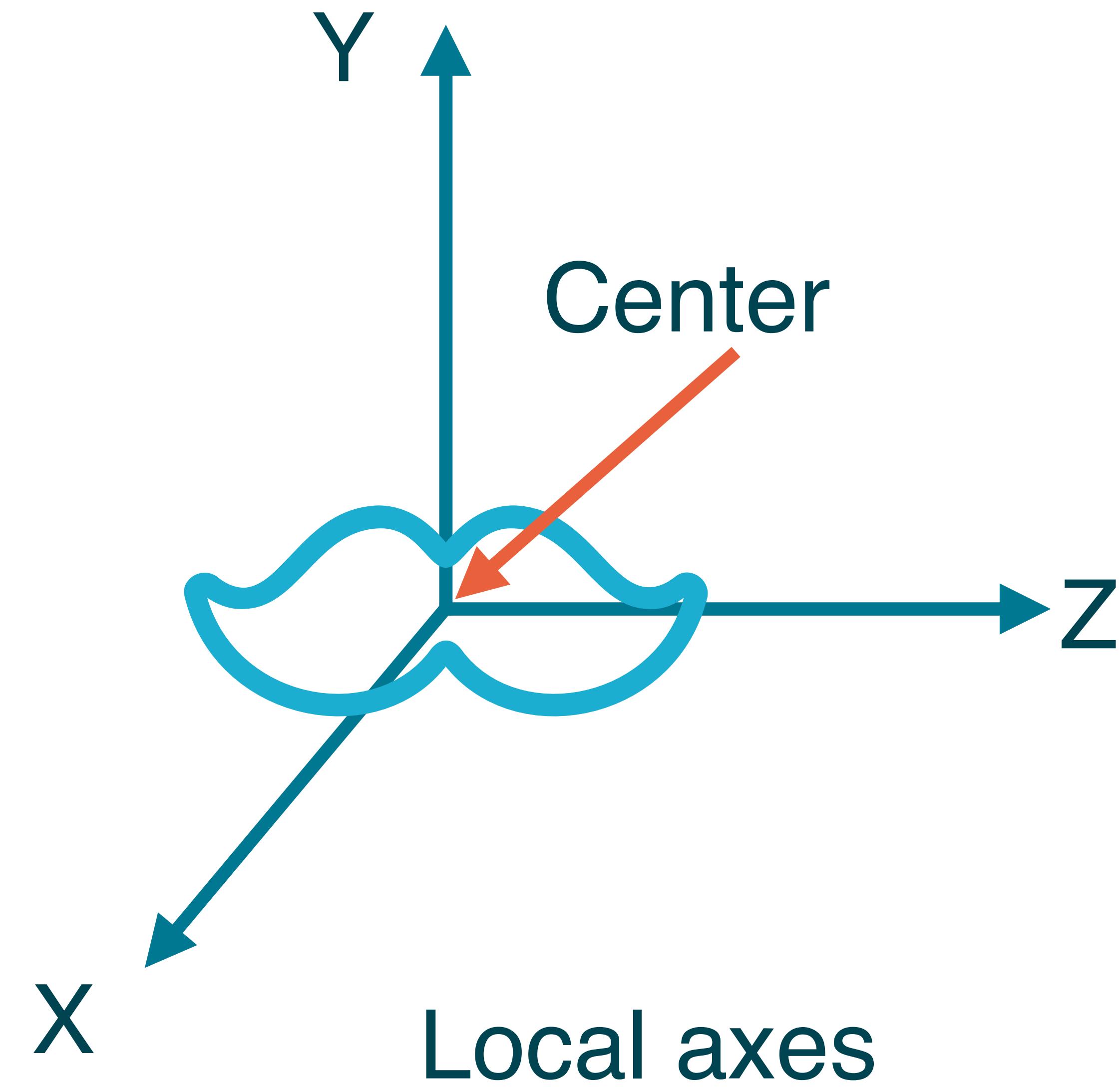
Positioning



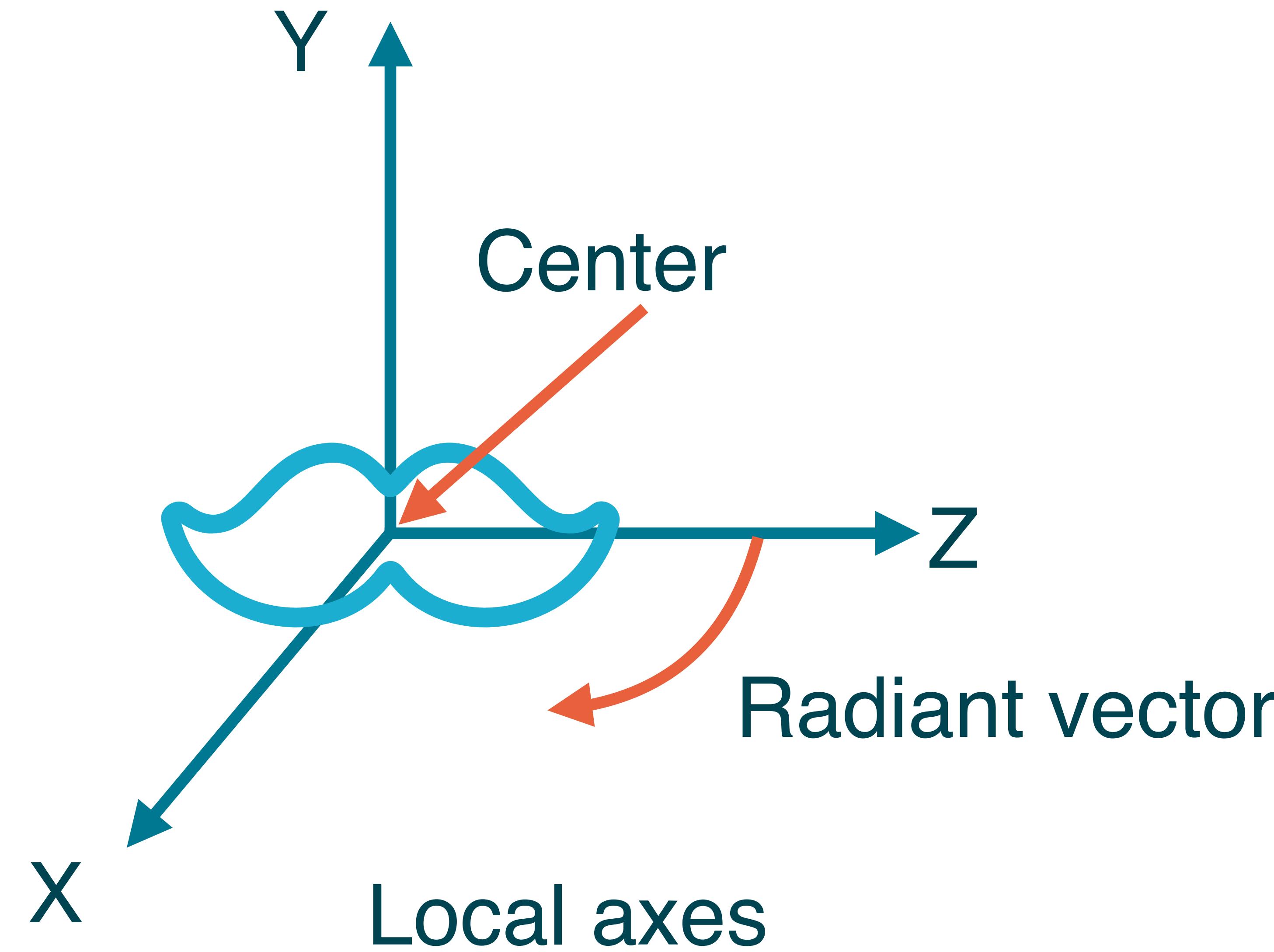
Positioning



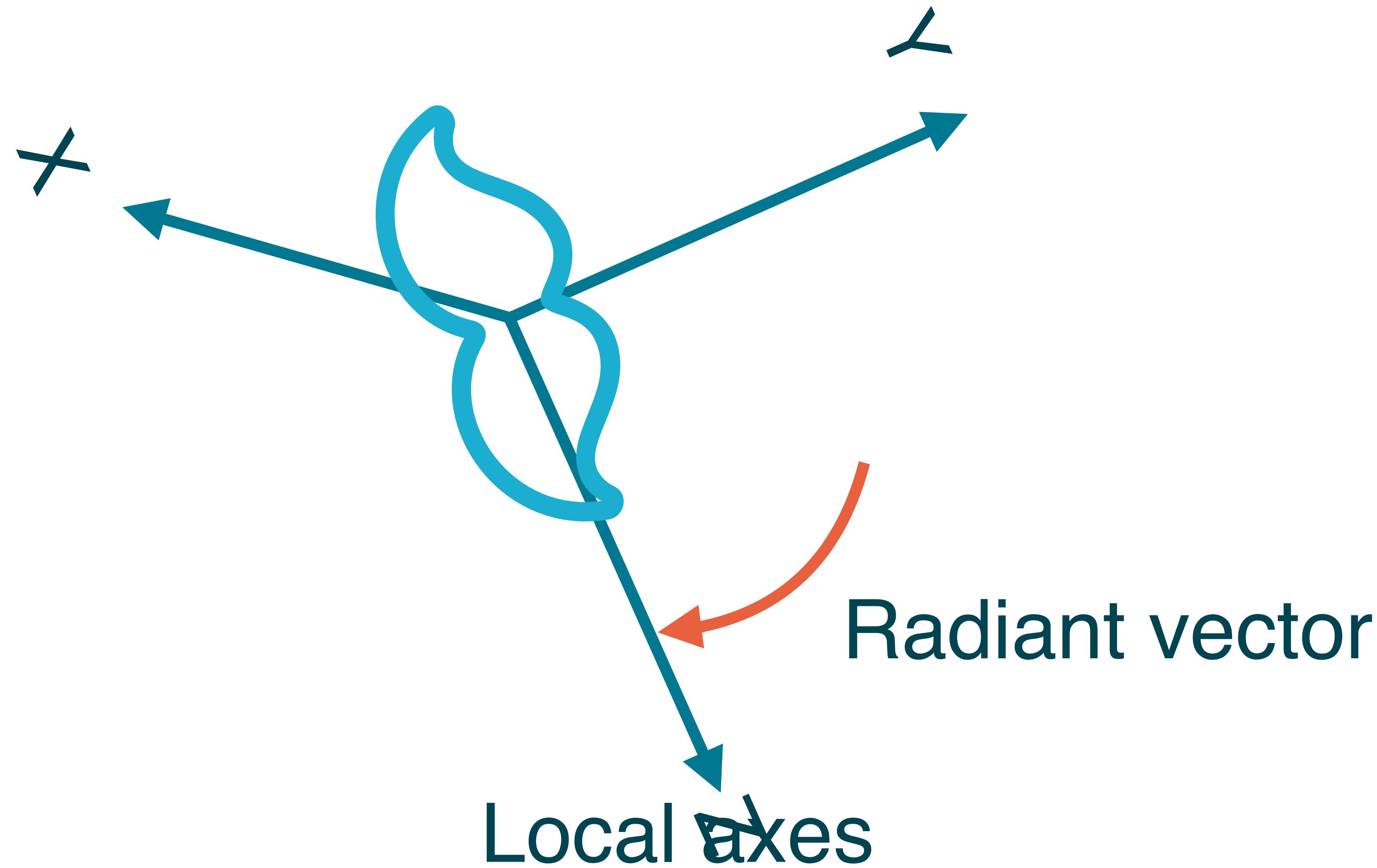
Rotation



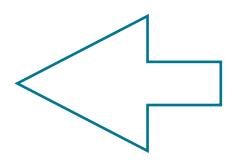
Rotation



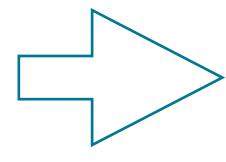
Rotation



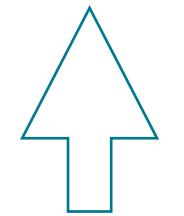
Universal camera



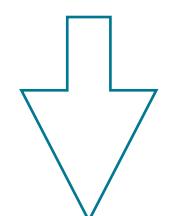
Left



Right



Forward



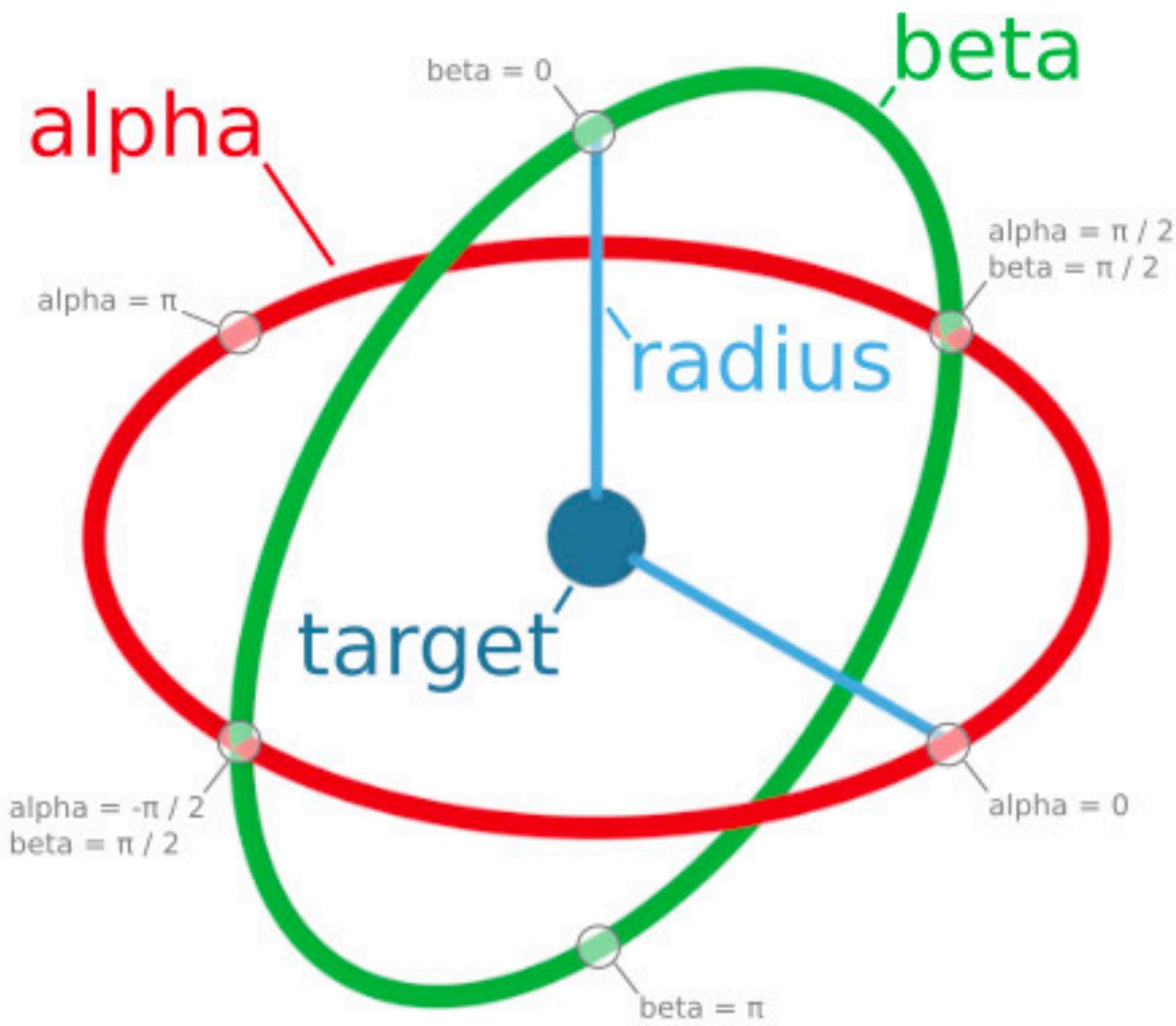
Backward

```
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



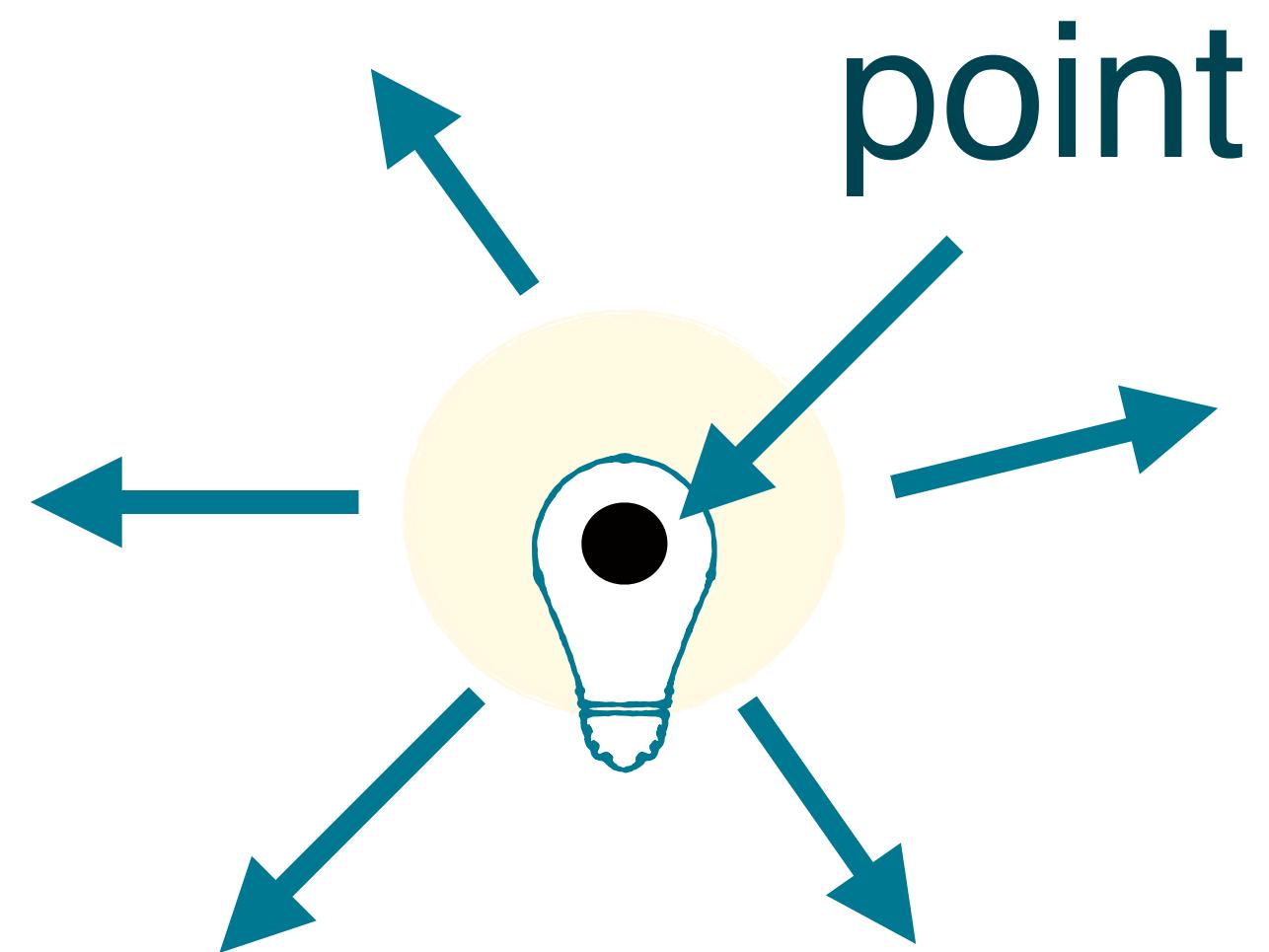
```
const camera = new ArcRotateCamera("camera", 0,  
0, 20, new Vector3(0, 0, 0), scene);  
  
// This attaches the camera to the canvas  
camera.attachControl(canvas, true);
```

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



Without light
everything
is black

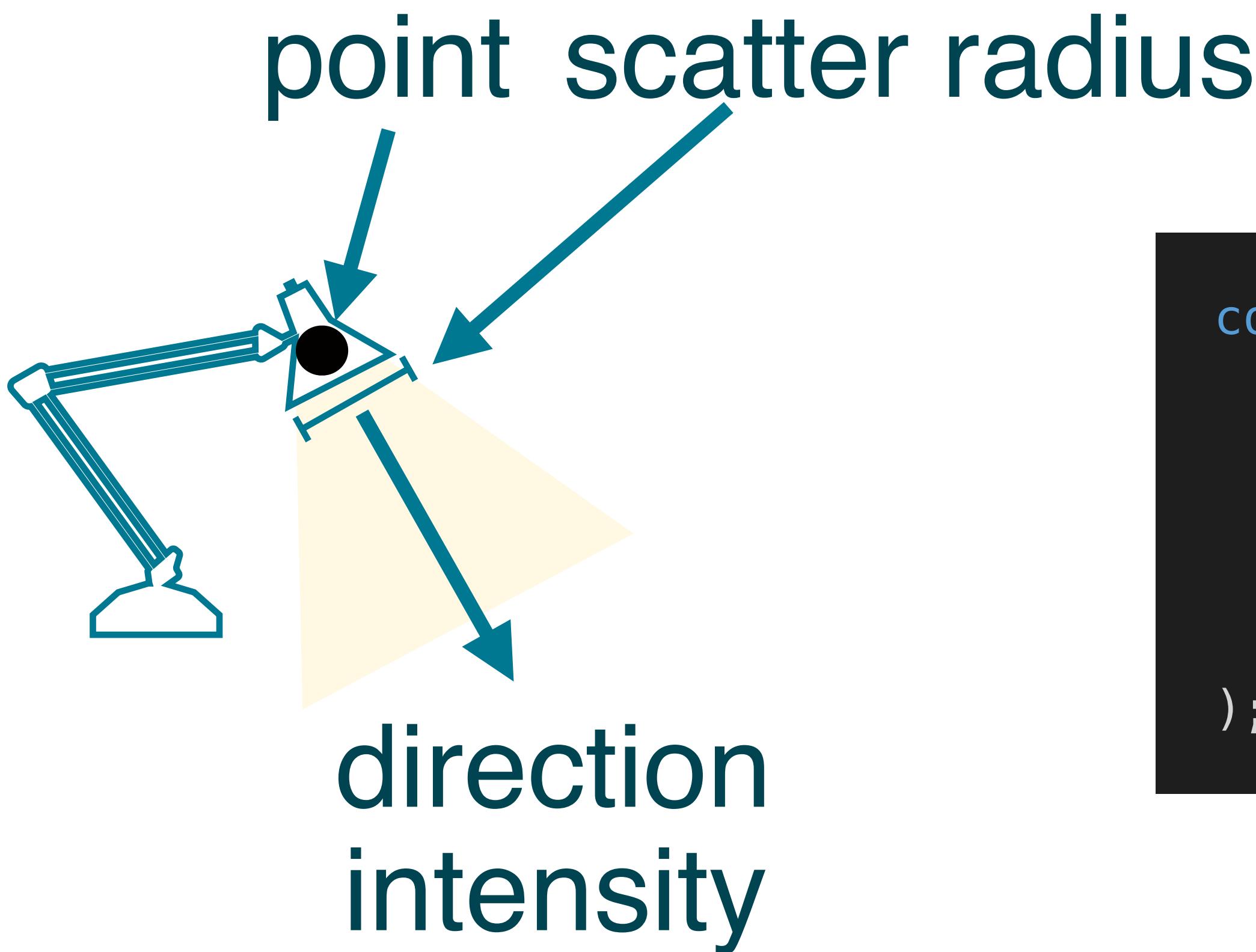
Point light



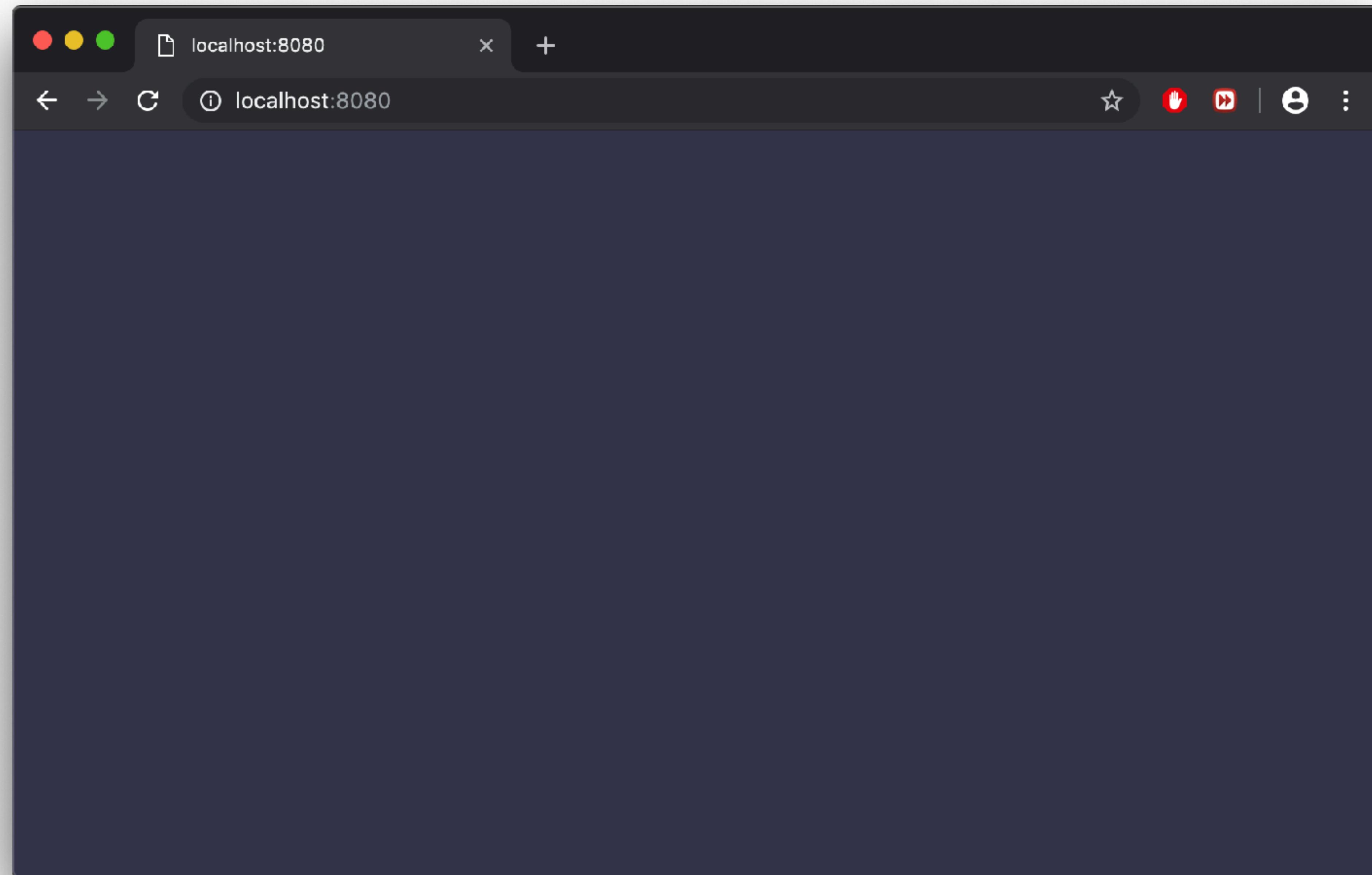
point

```
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  
)
```

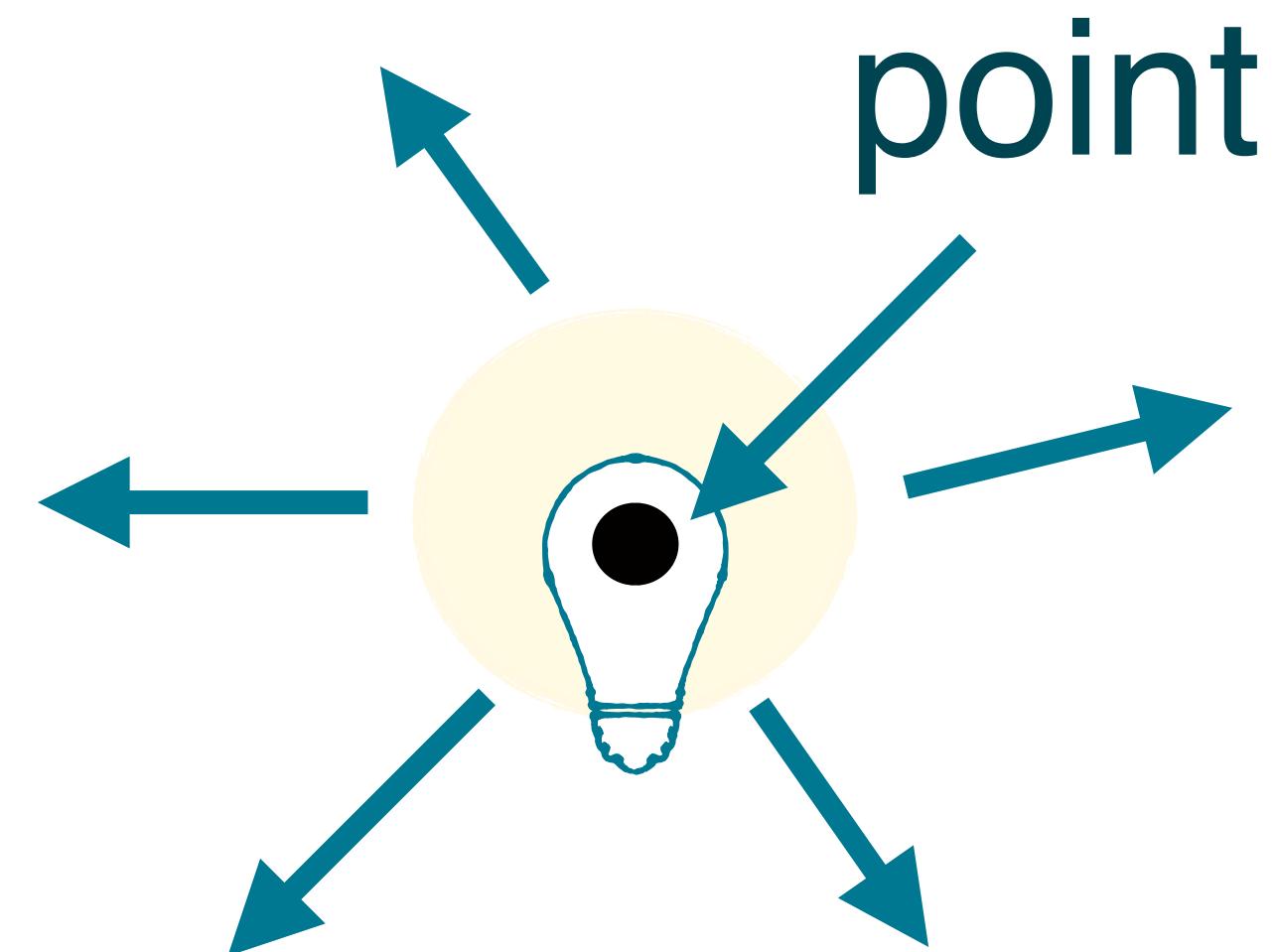
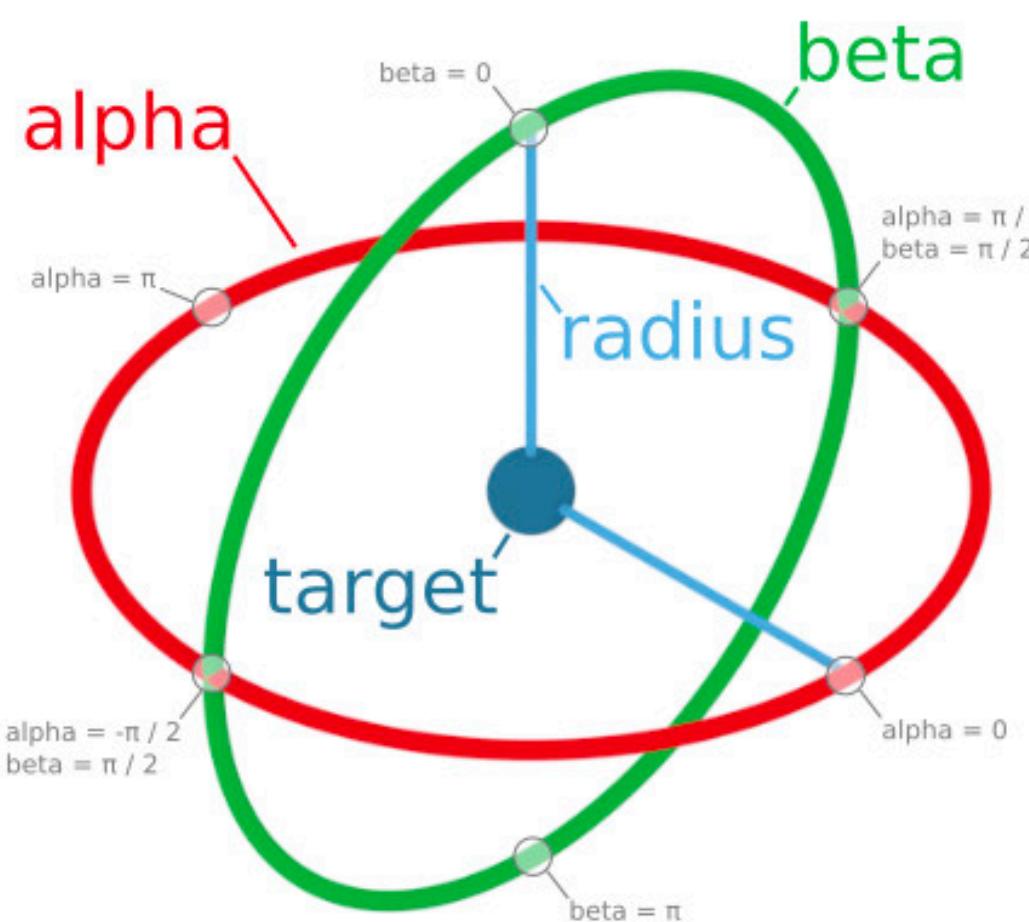


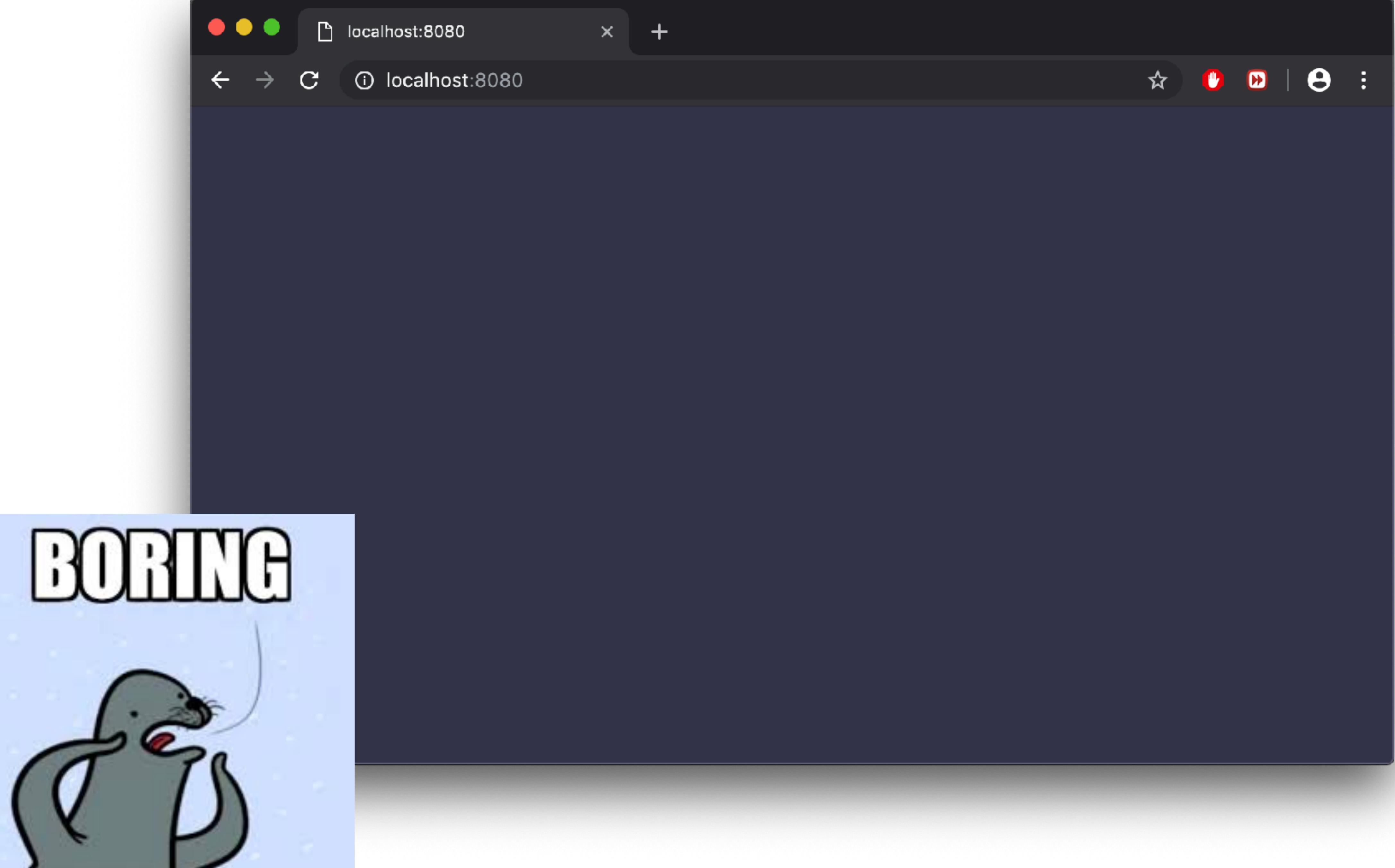
Task 1: Let there be light

Add an **ArcRotateCamera**.

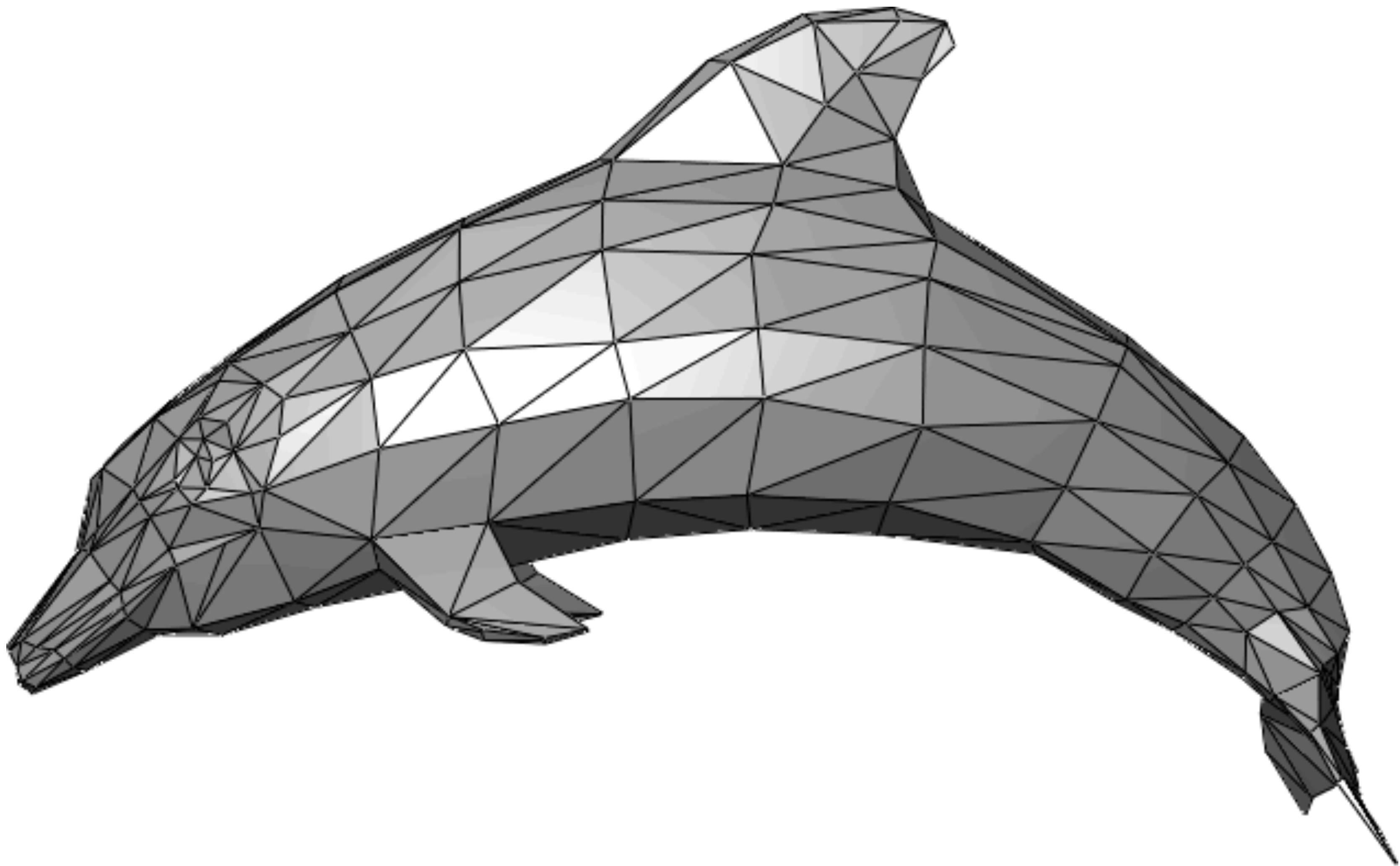
Point the camera to the Origin.

Add a **PointLight**.

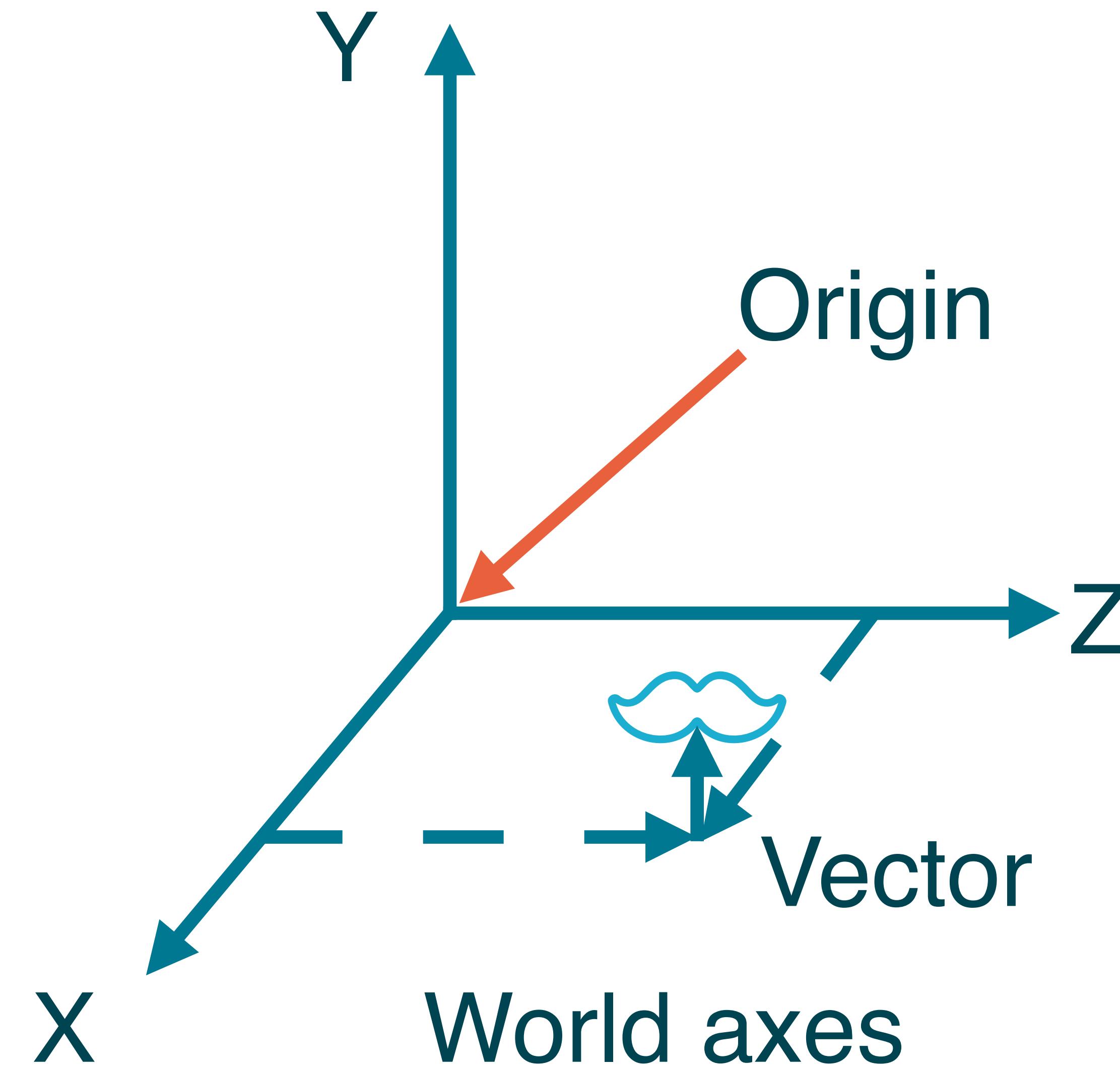




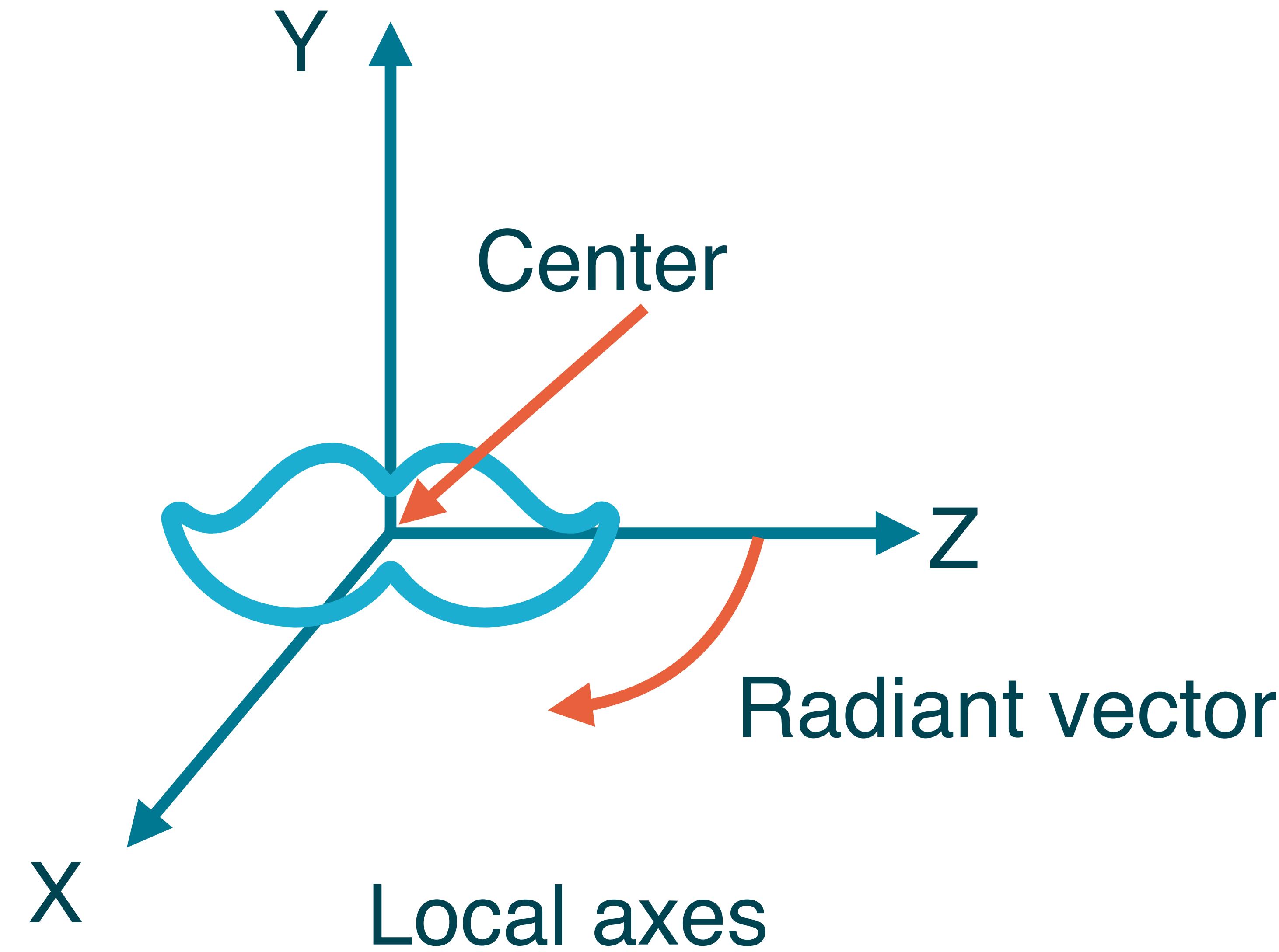
Mesh



Positioning



Rotation



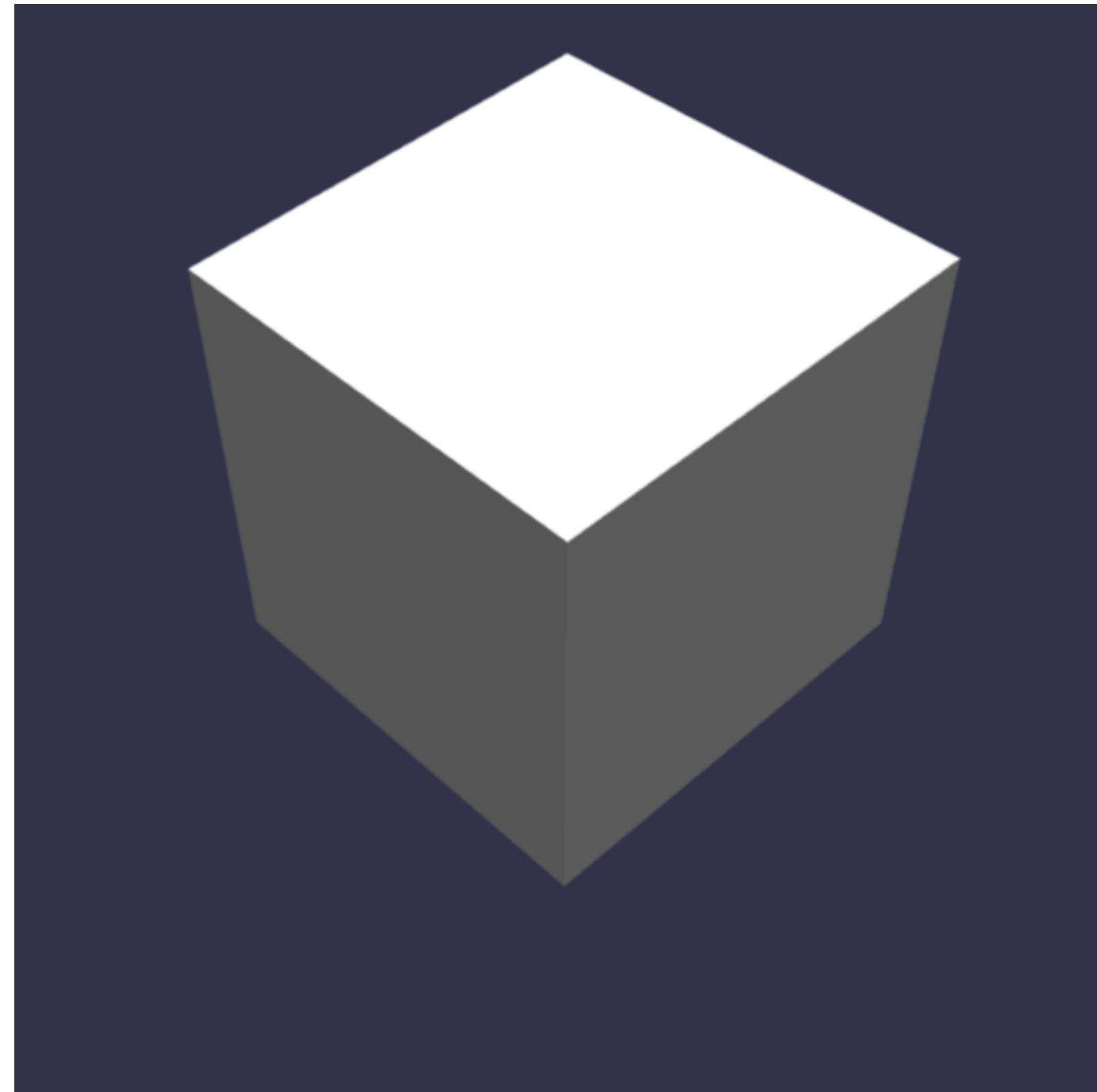
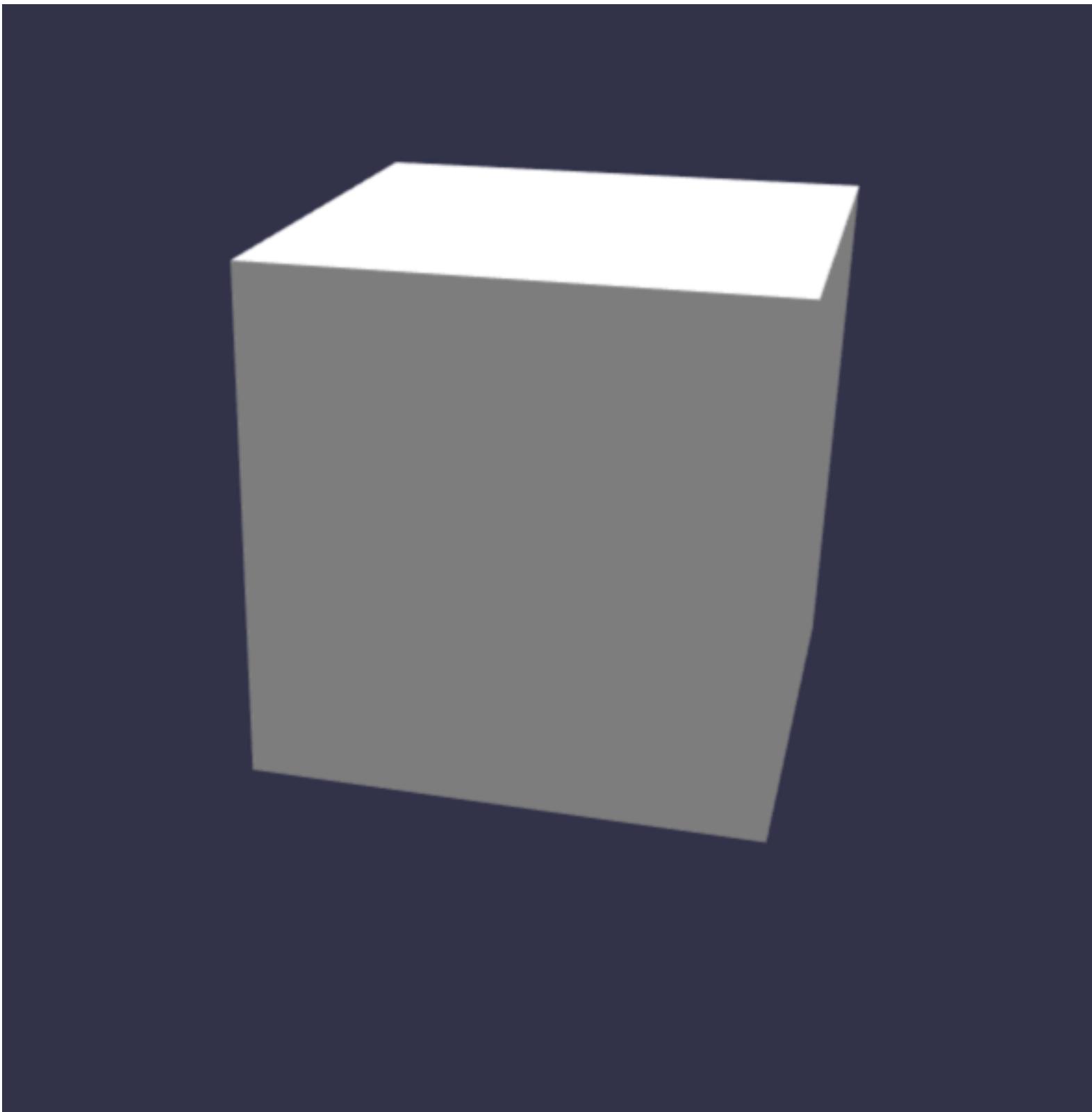
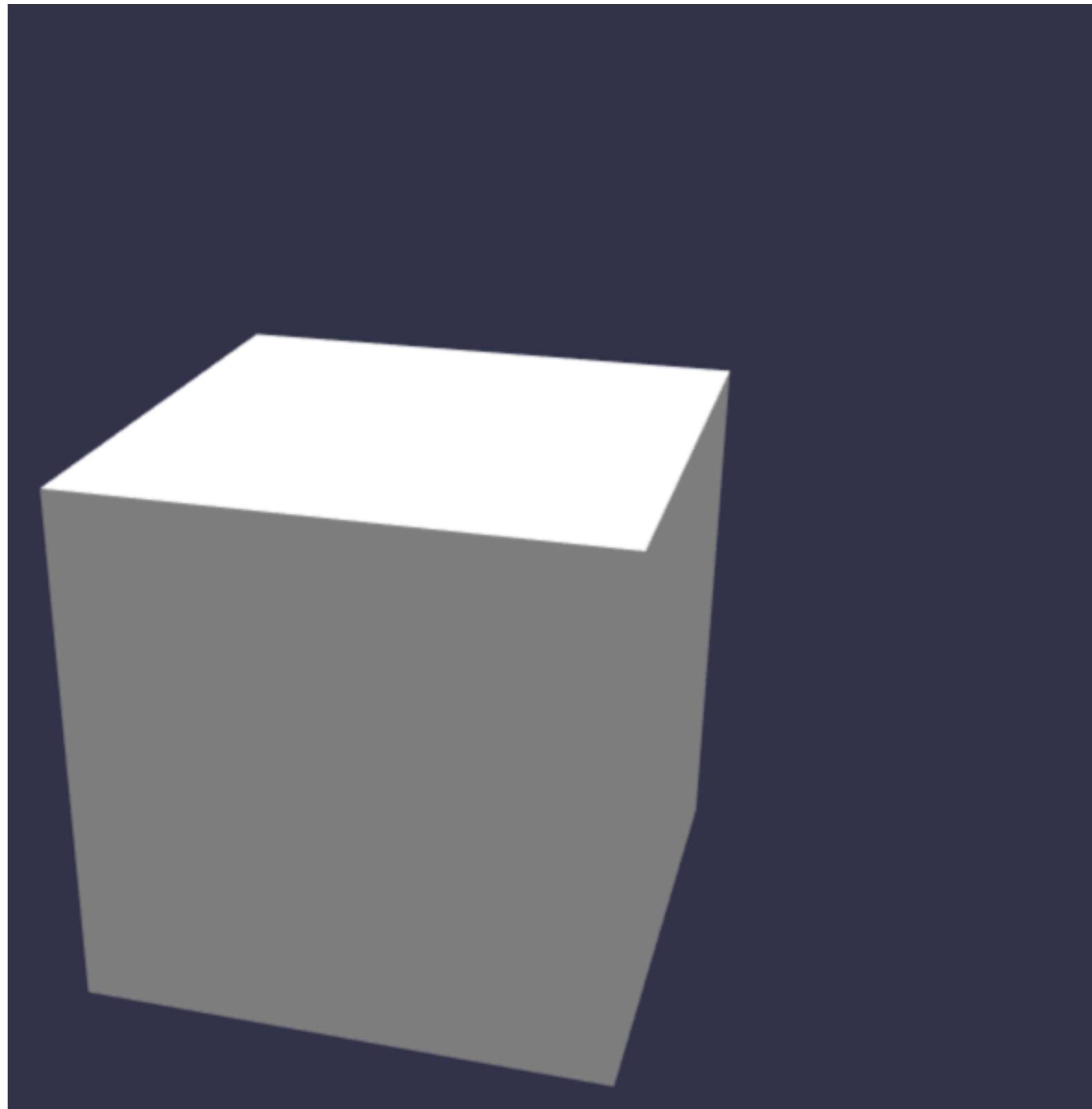
Example

```
//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

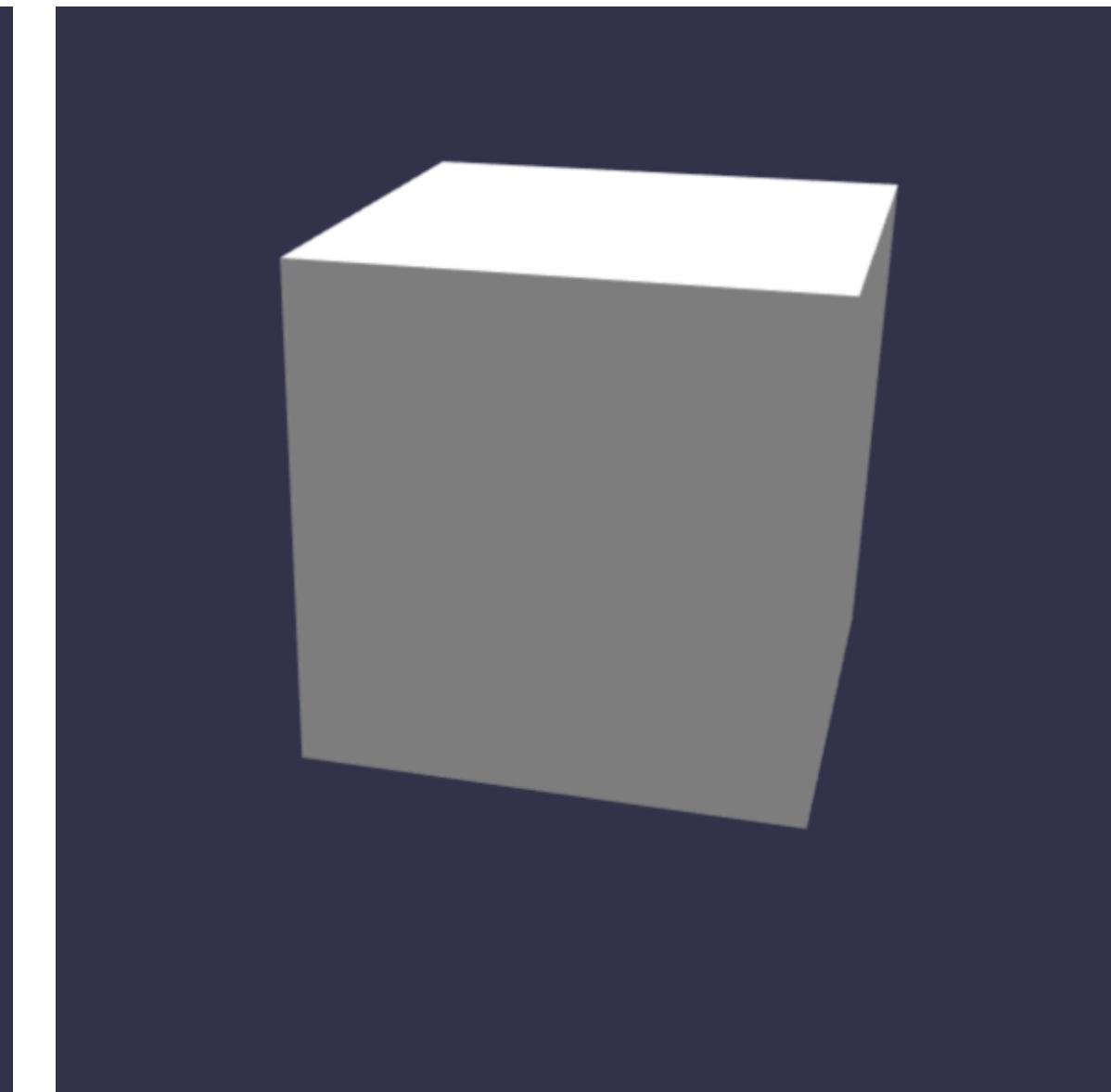
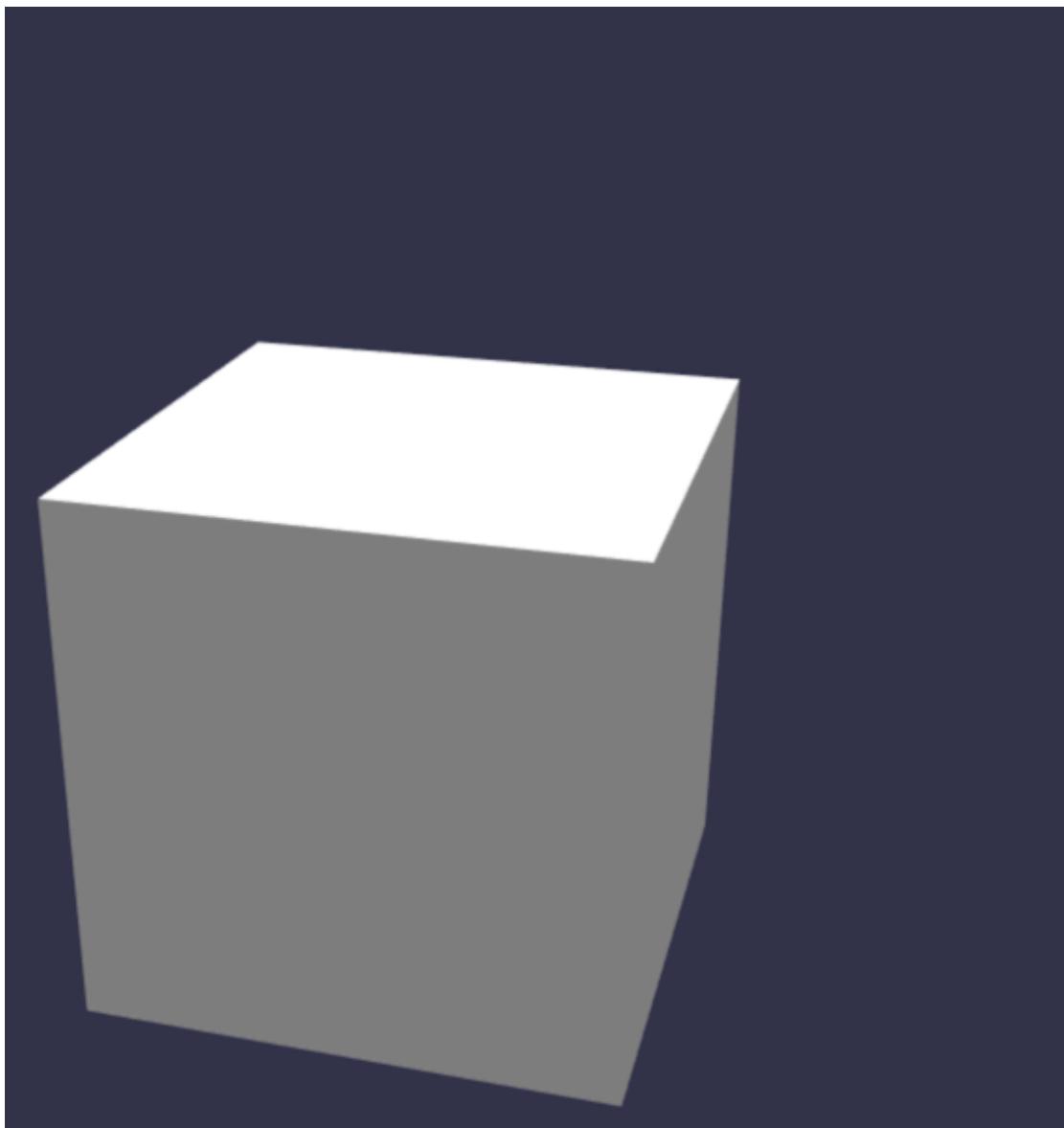


Task 2: A box

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

Position the box.

Rotate the box.

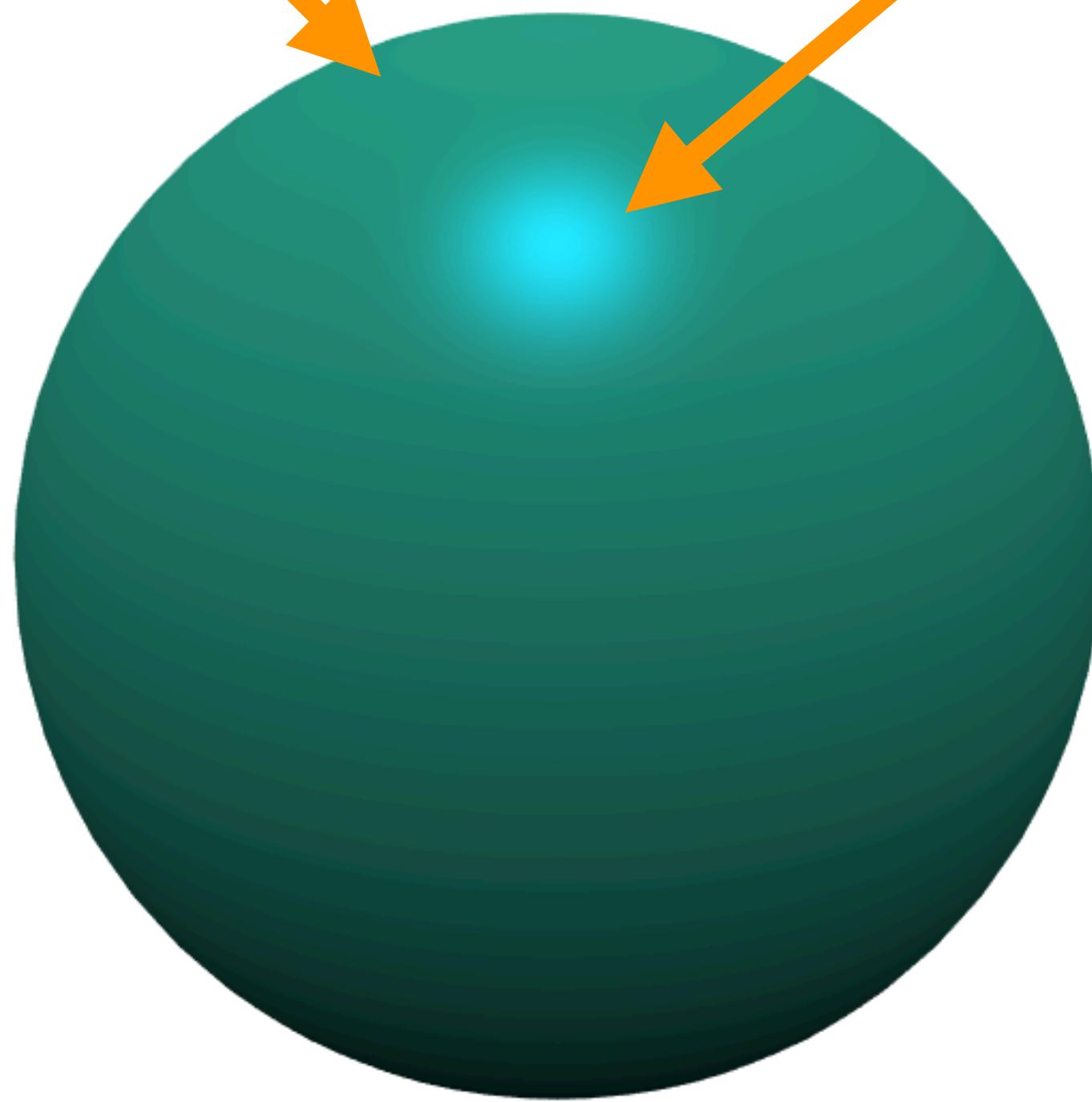


BORING

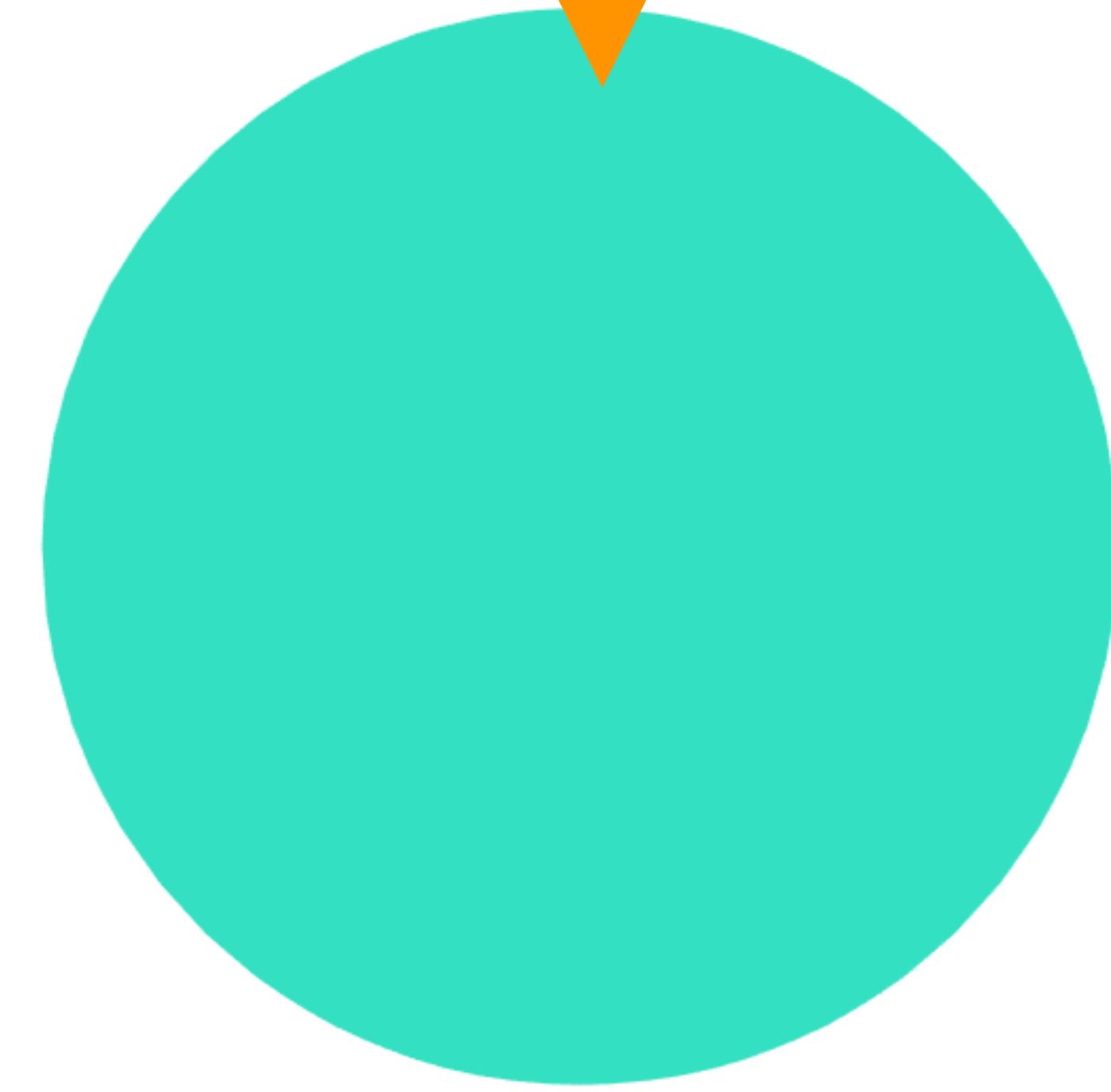


Materials

diffuse

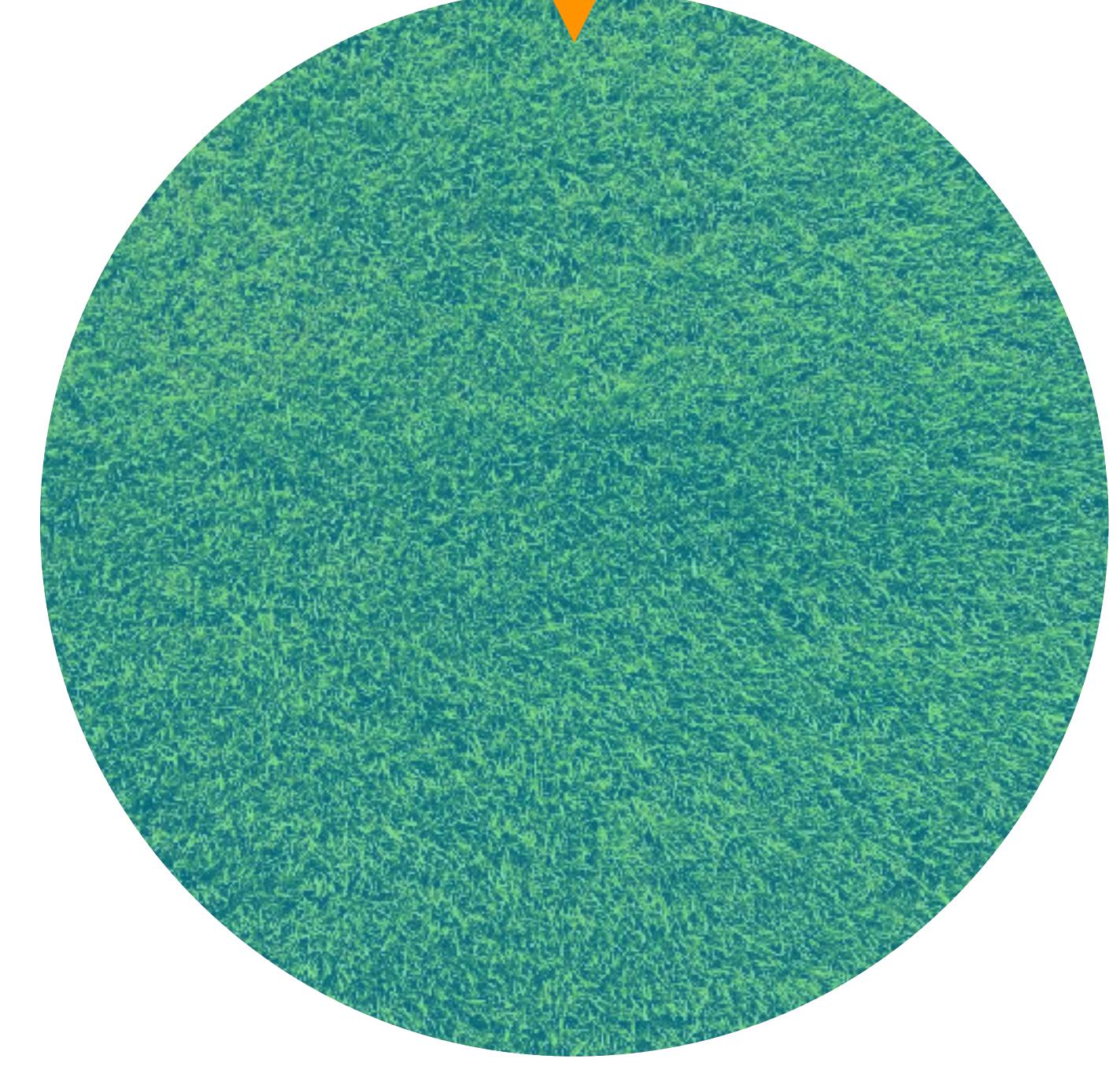


specular



emissive

texture

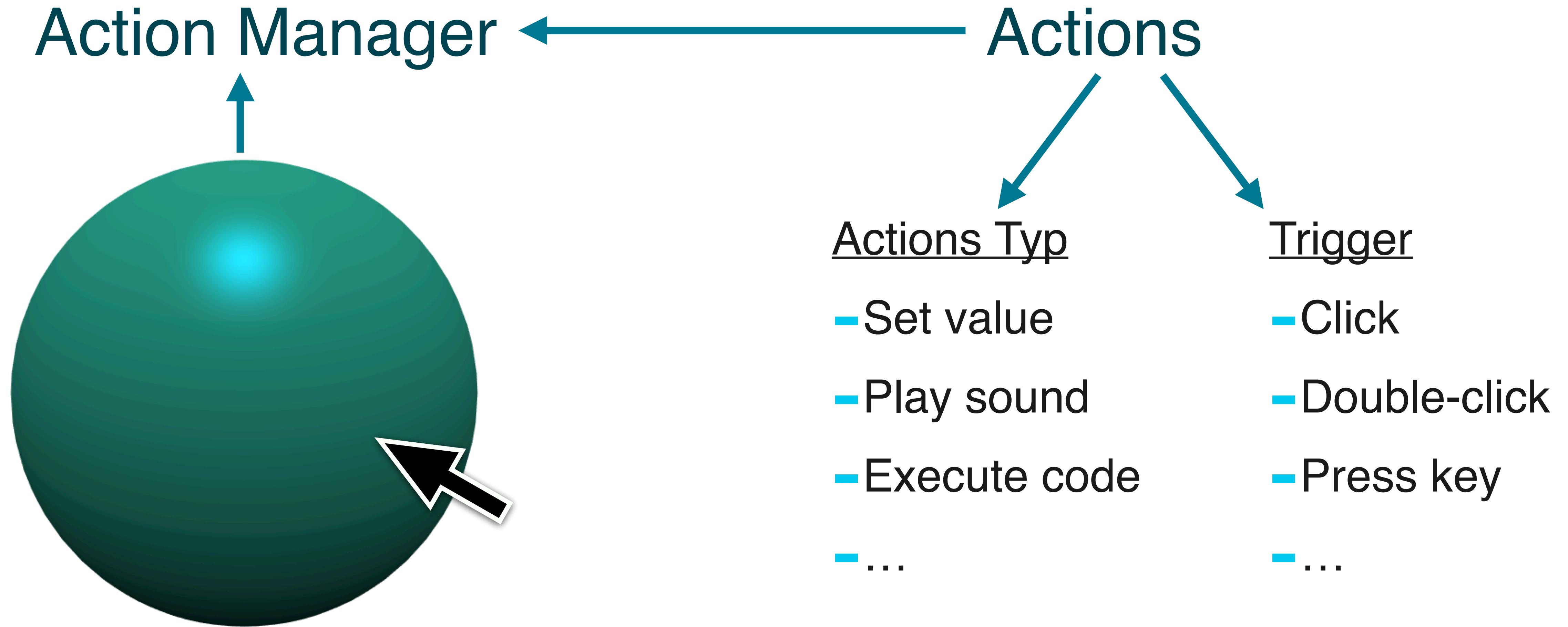


Materials

```
// 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;
```

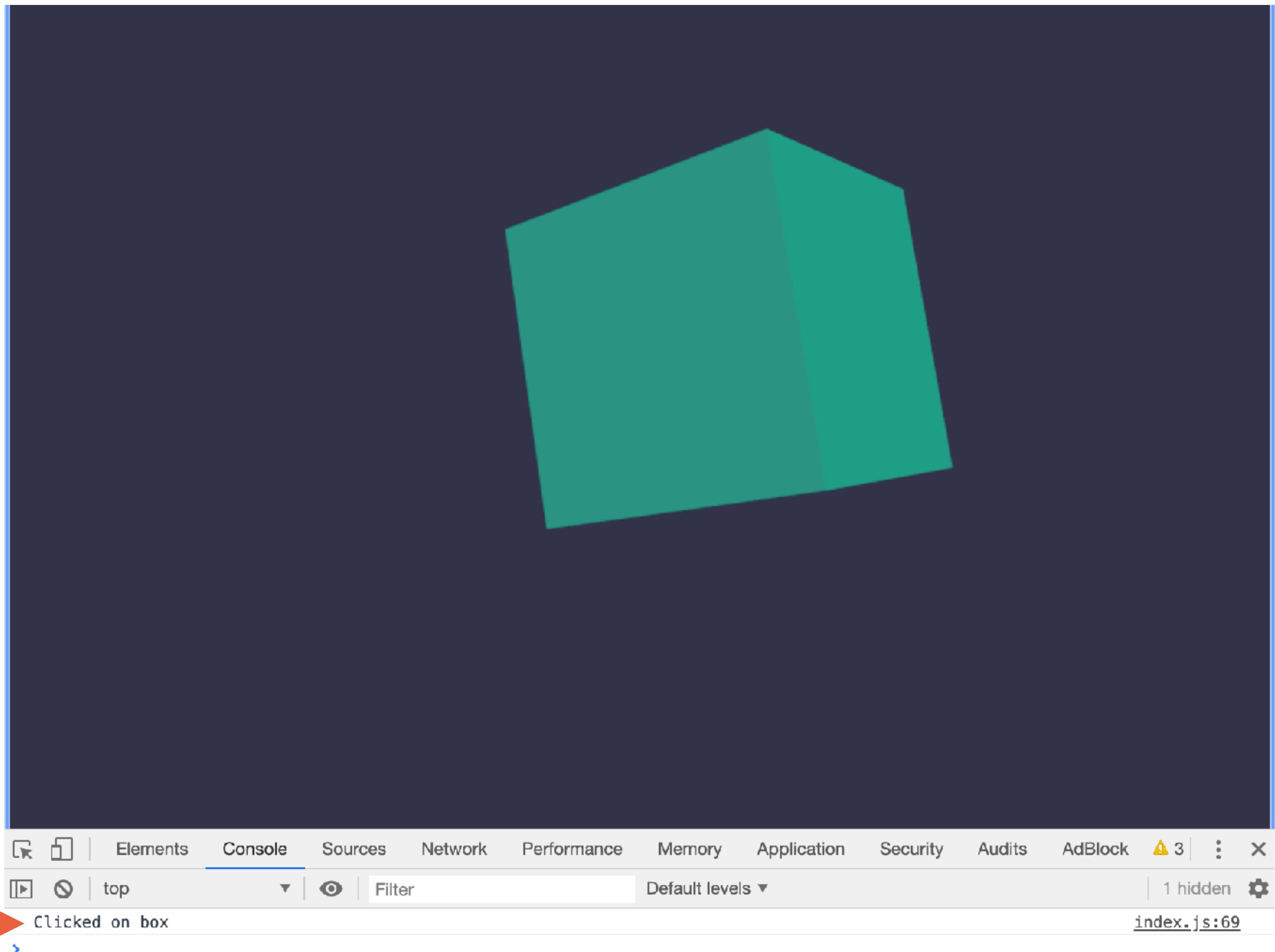
User interaction



Example

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

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

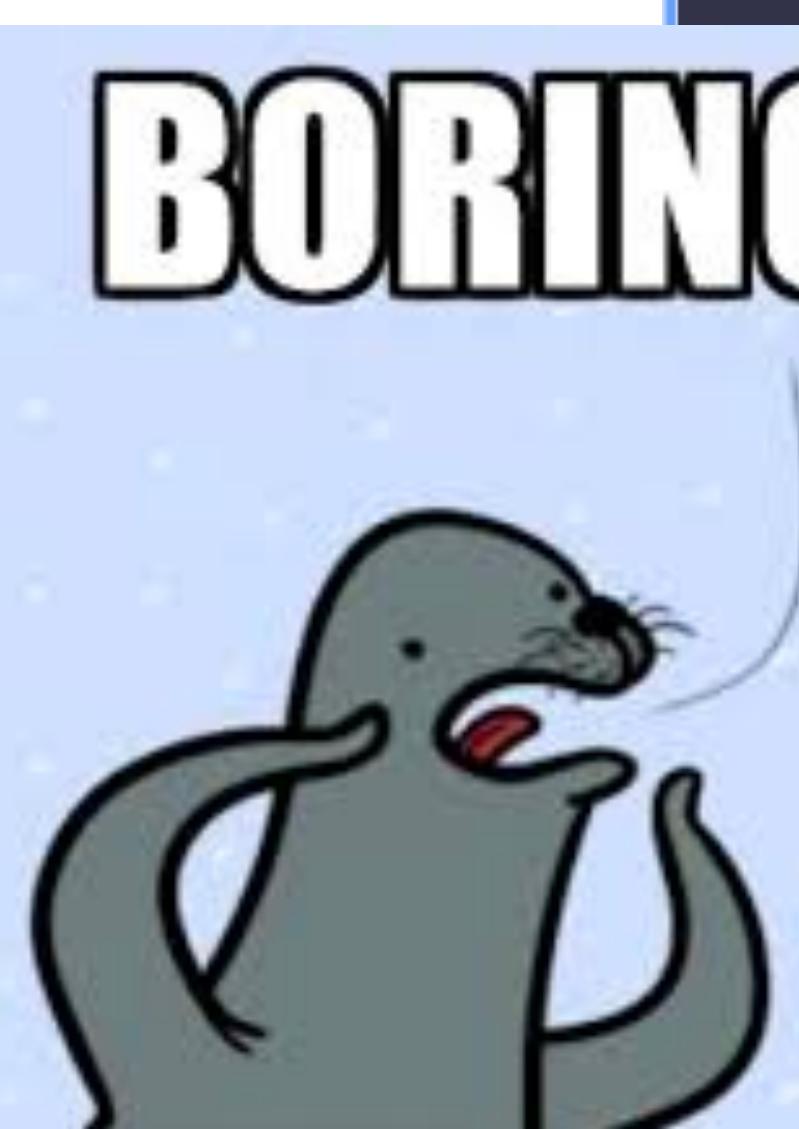


Task 3: Colors :D

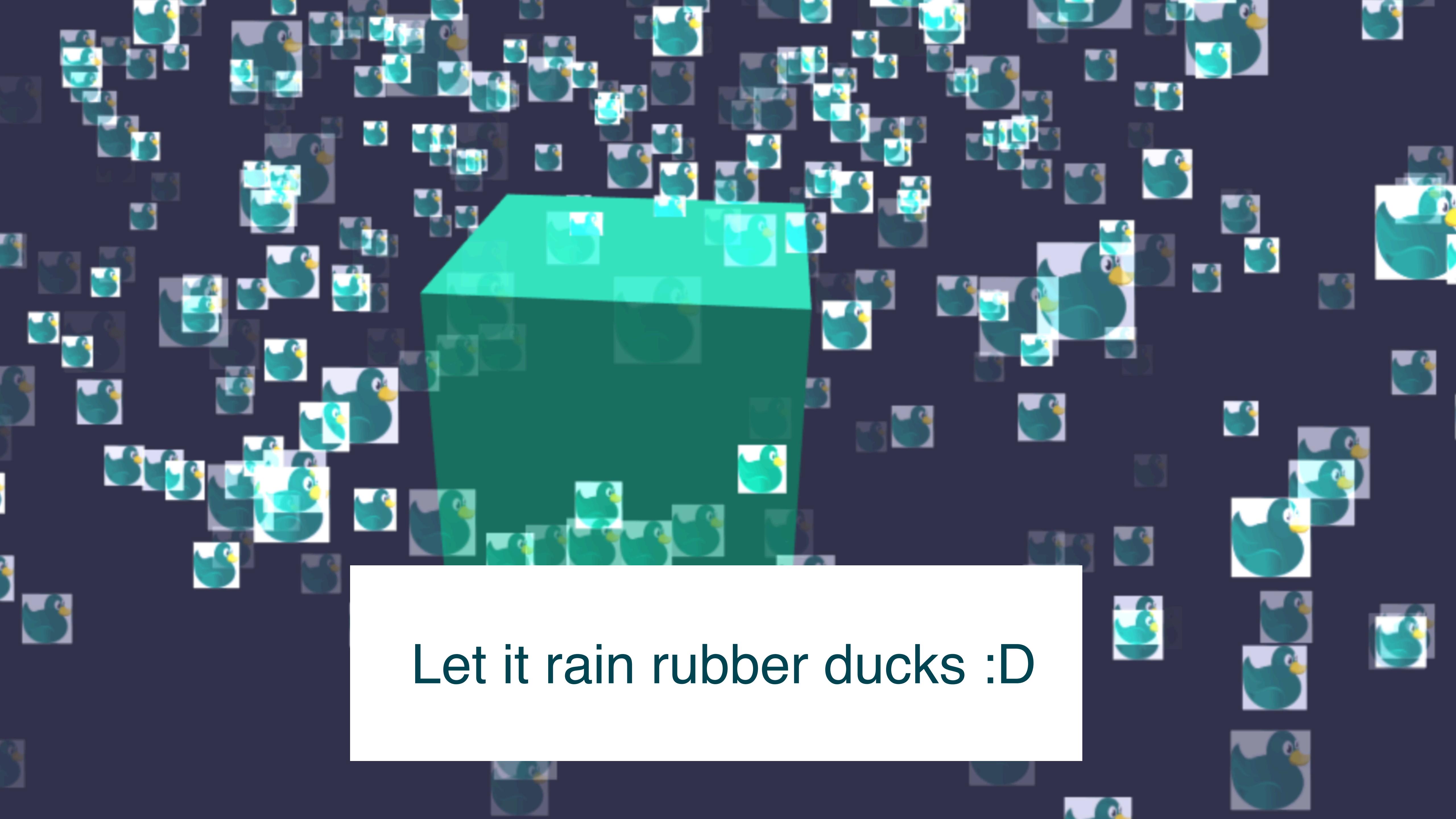
Use a **StandardMaterial** to make the box colorful.

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



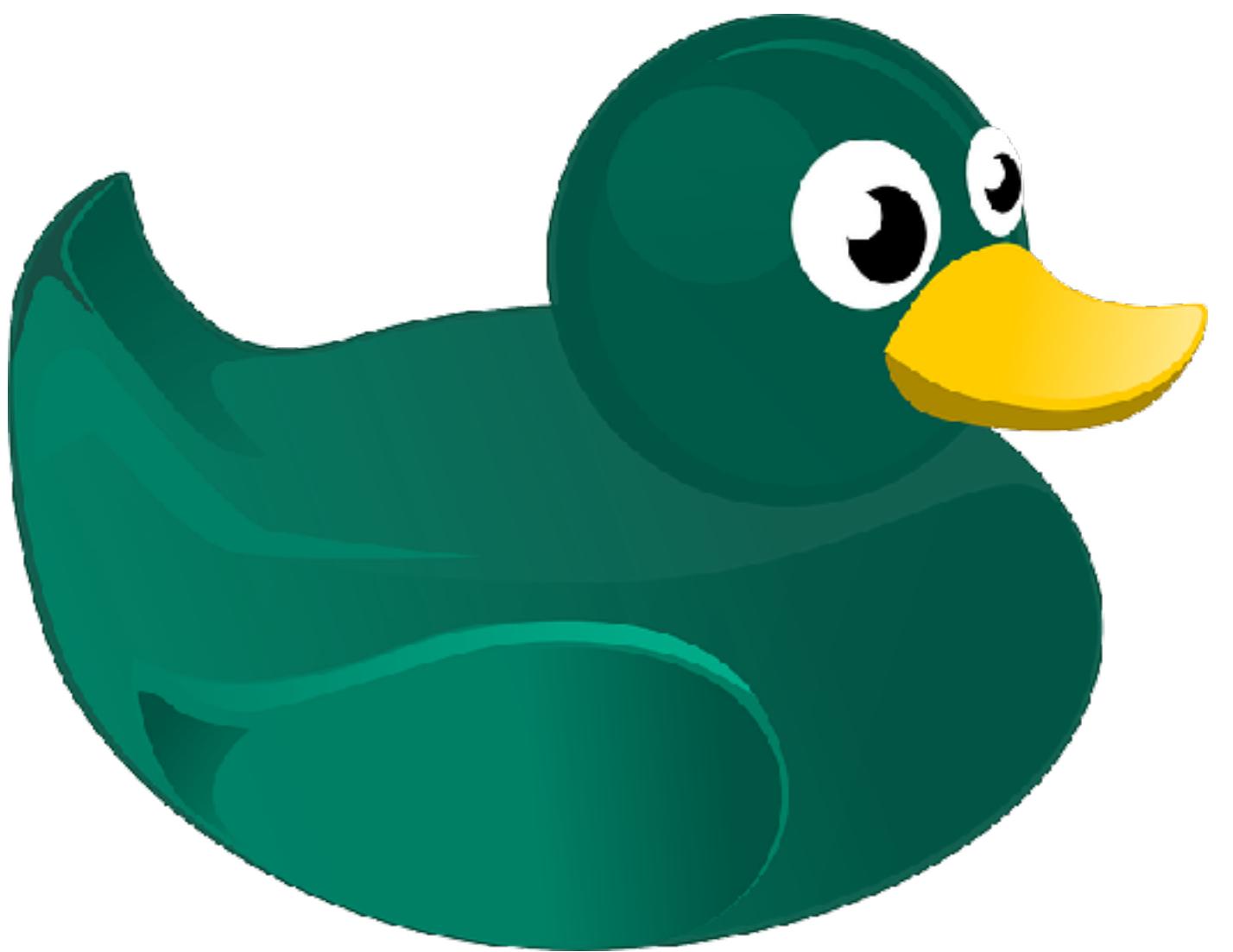


	index.js:69
x	



Let it rain rubber ducks :D

Particle System

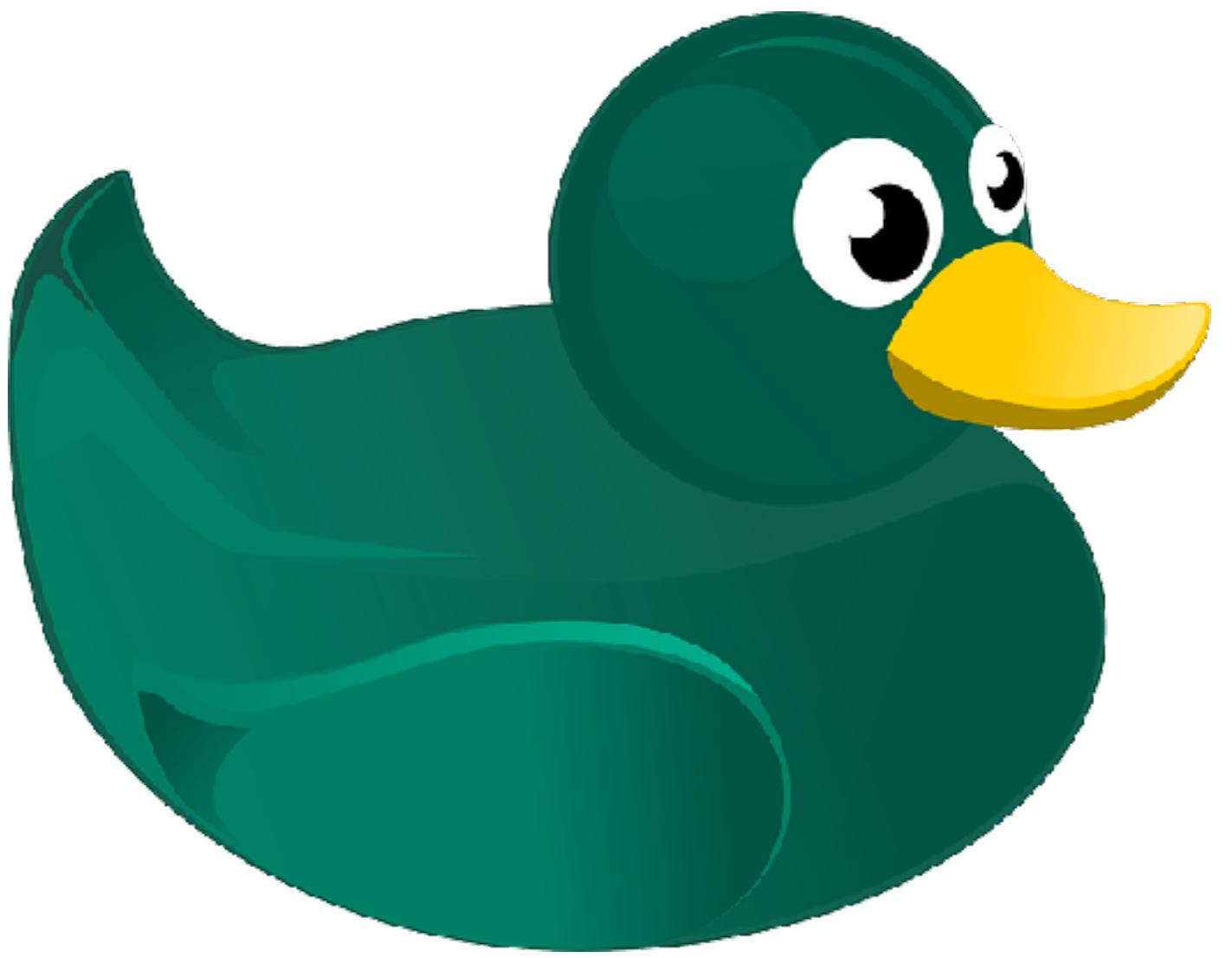


2D Sprite



Emitter

Particle System

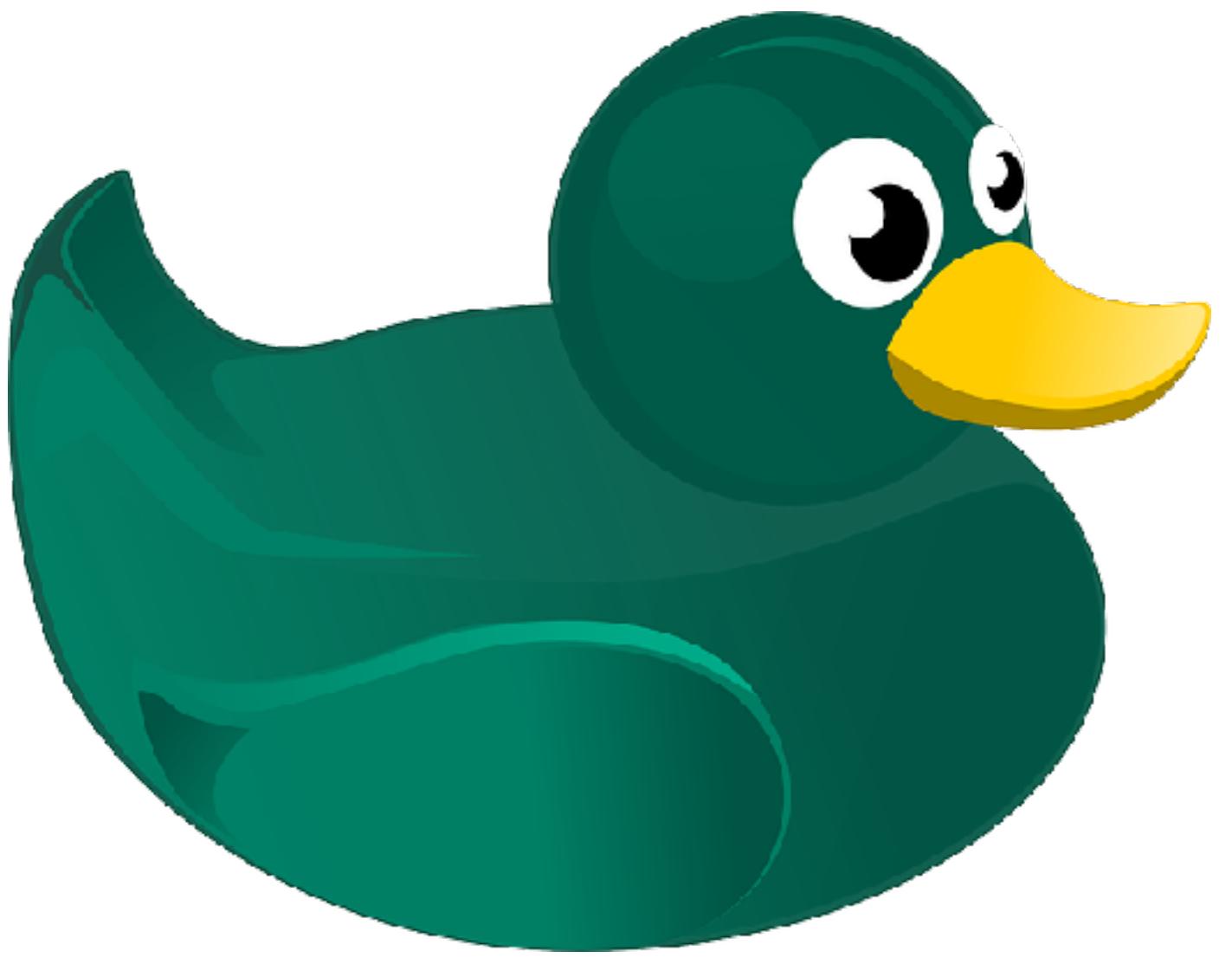


2D Sprite

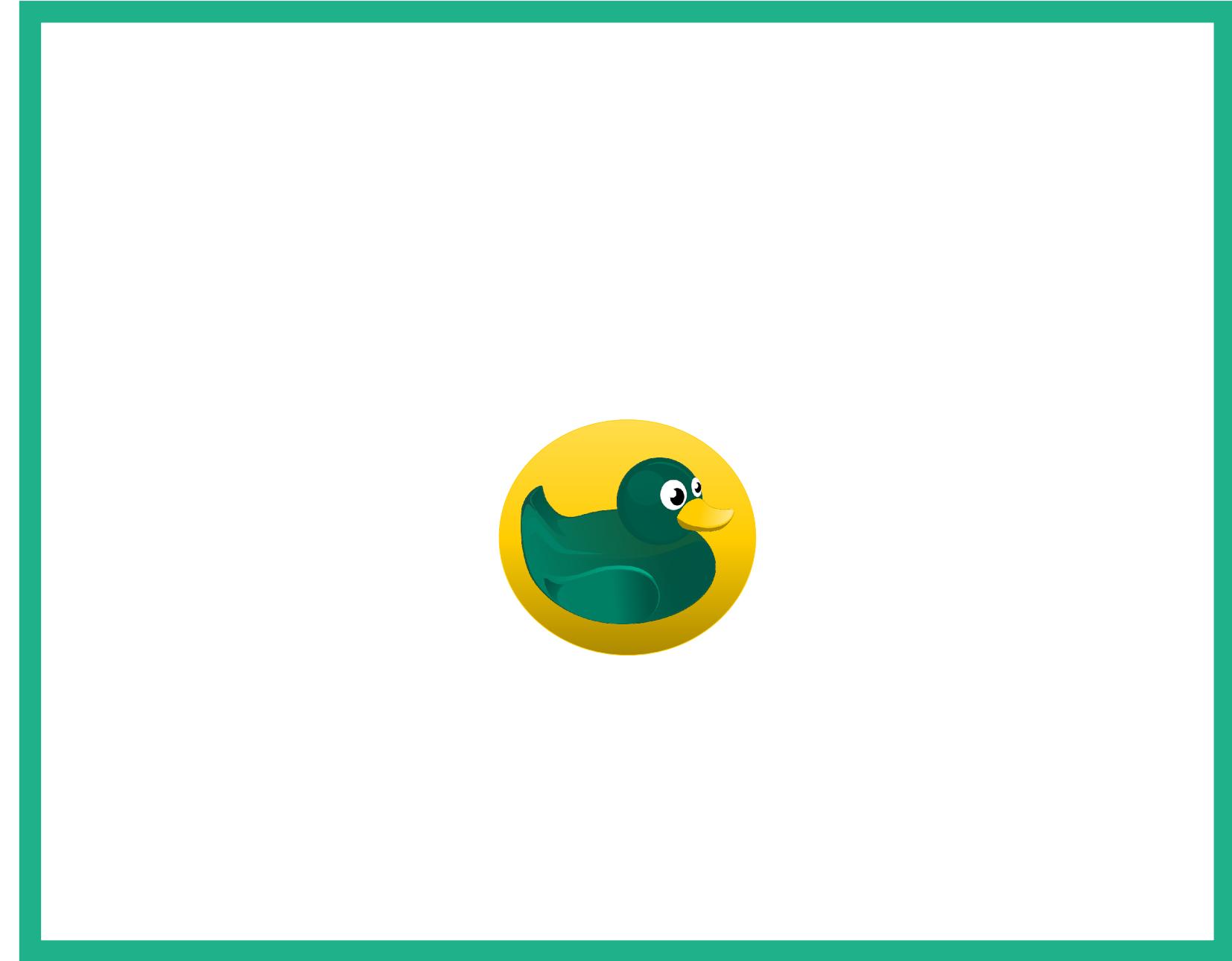


Emitter

Particle System

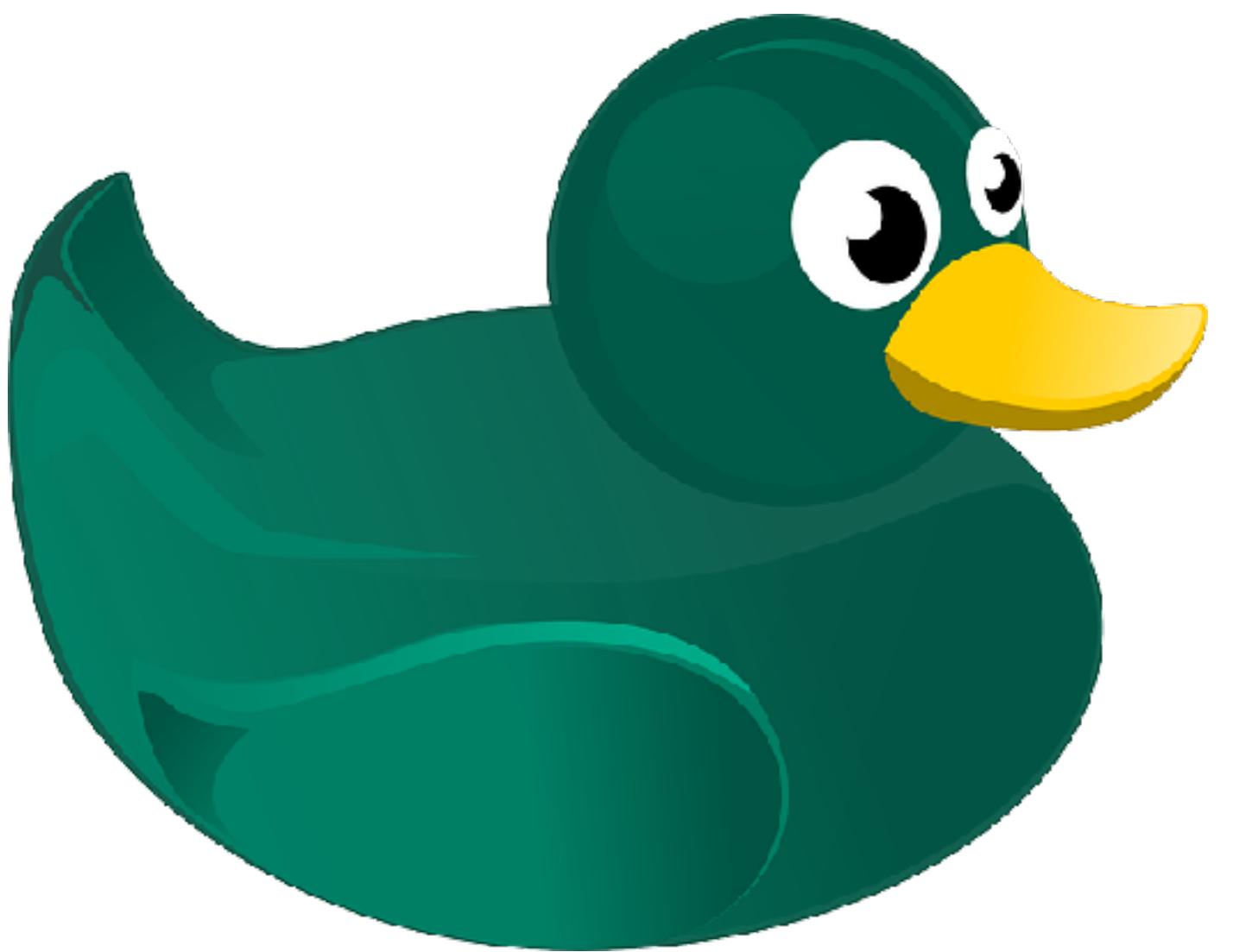


2D Sprite

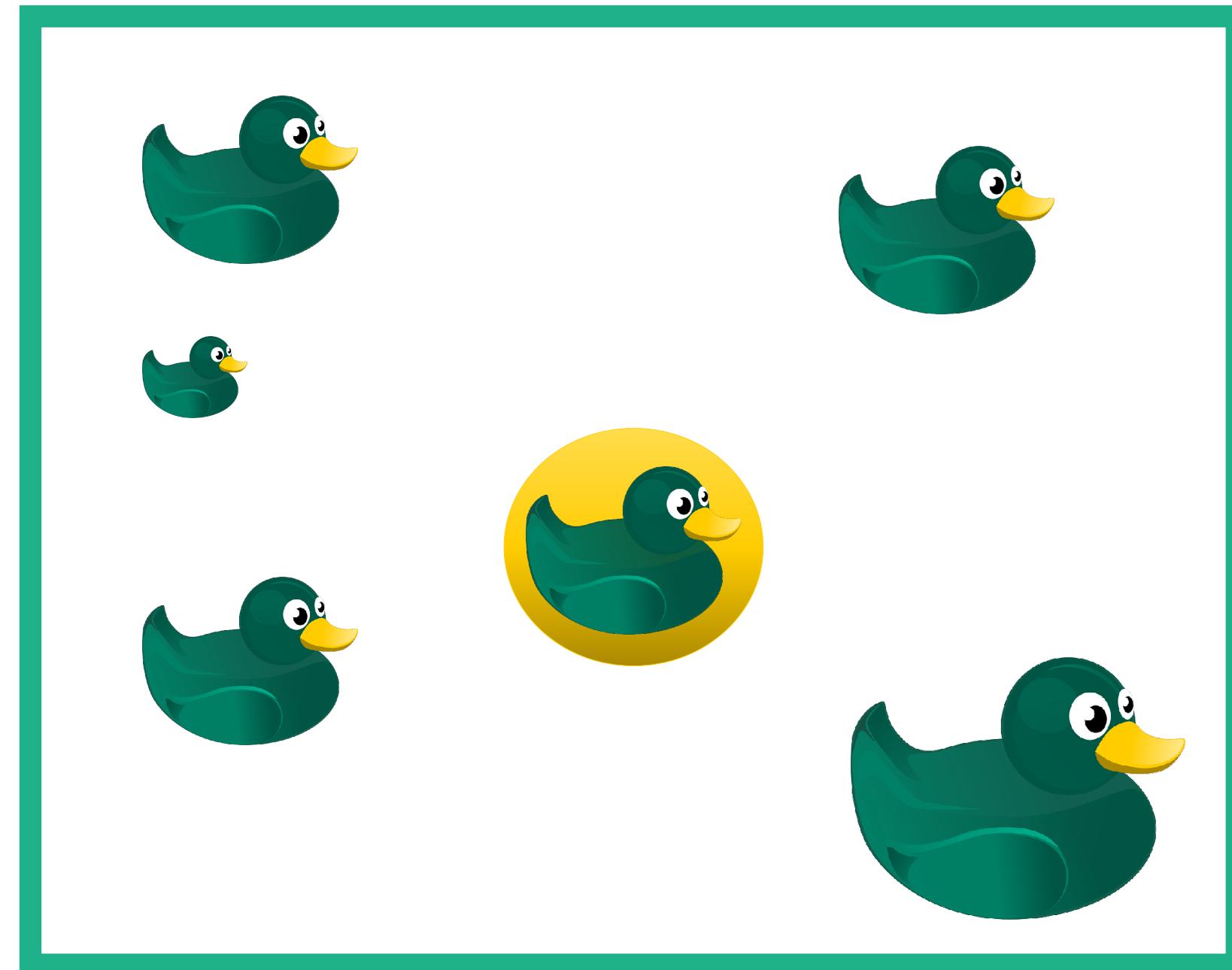


Emitter

Particle System



2D Sprite



Emitter

Particle Systems

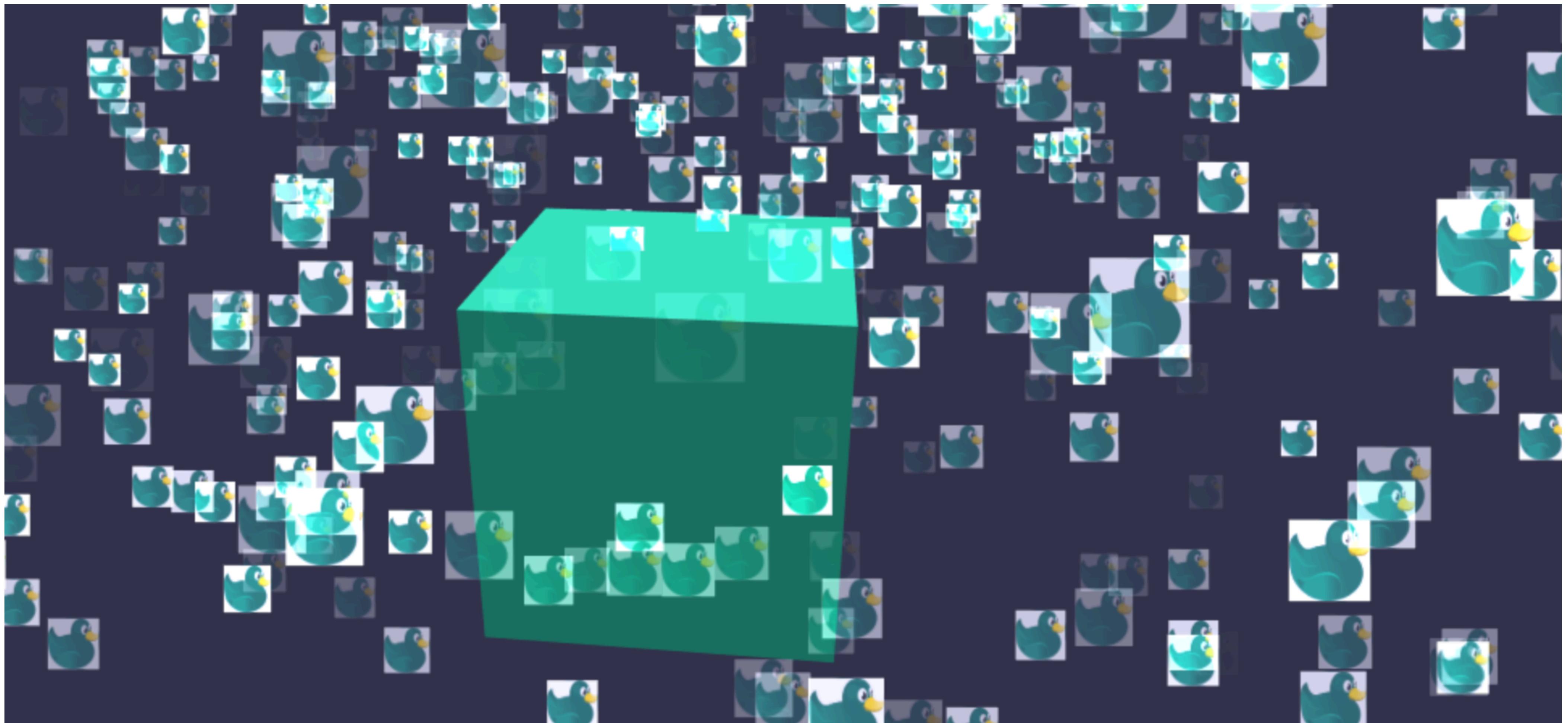
```
const duckParticleSystem = new ParticleSystem("ducks", 2000, scene);

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

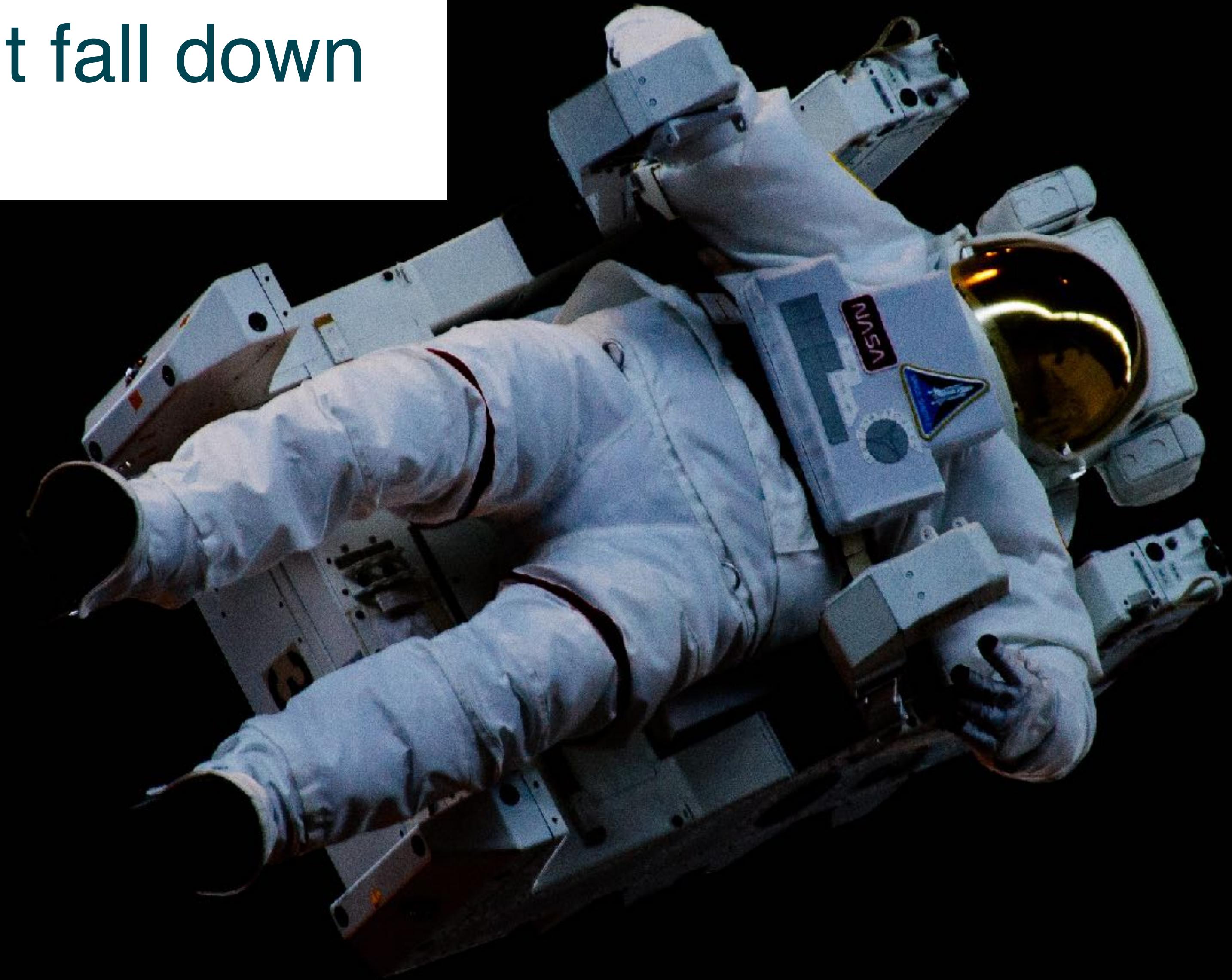
Use the **ParticleSystem** with a **Texture** of your choice.



BORING



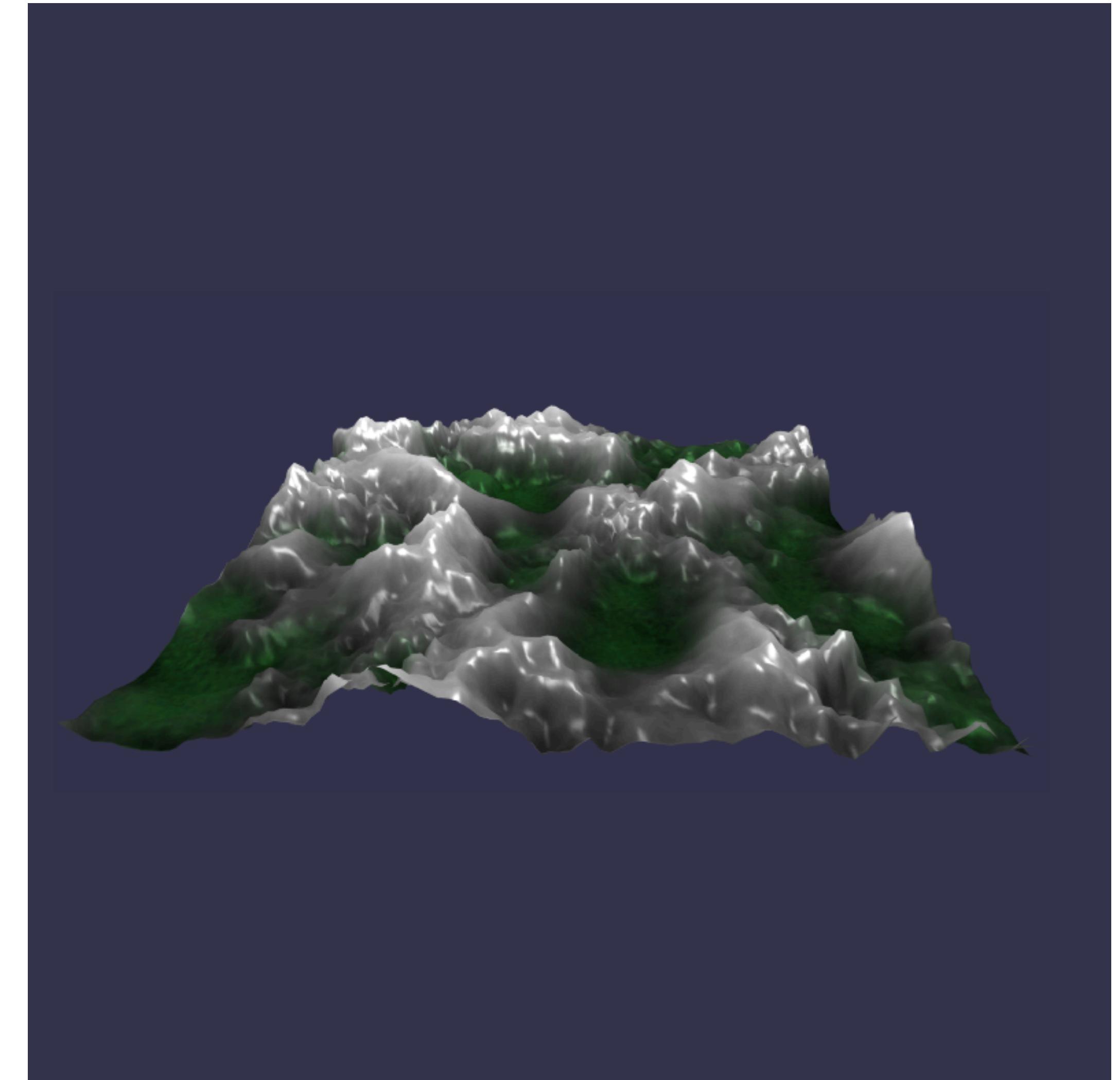
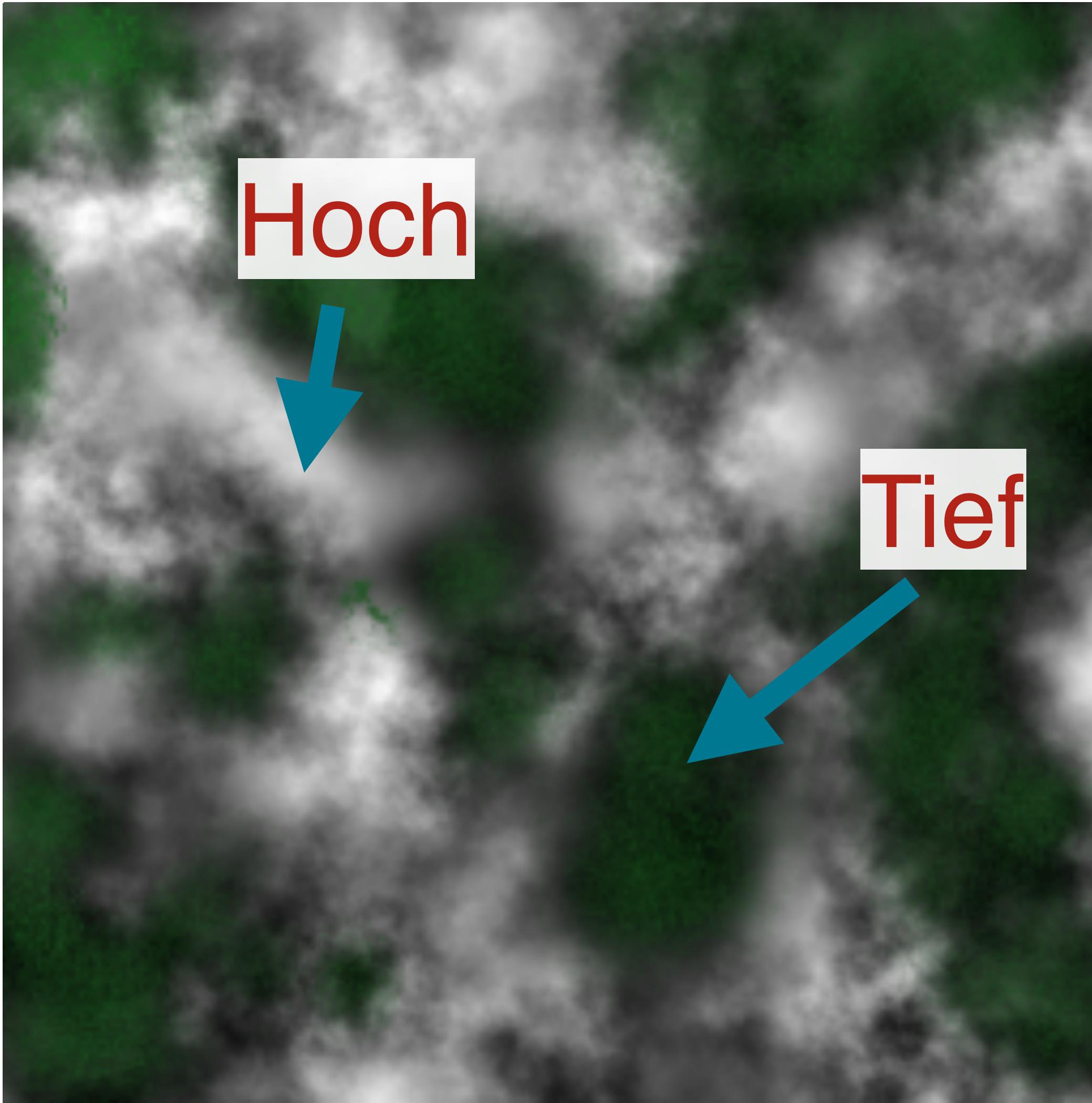
Let it fall down



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

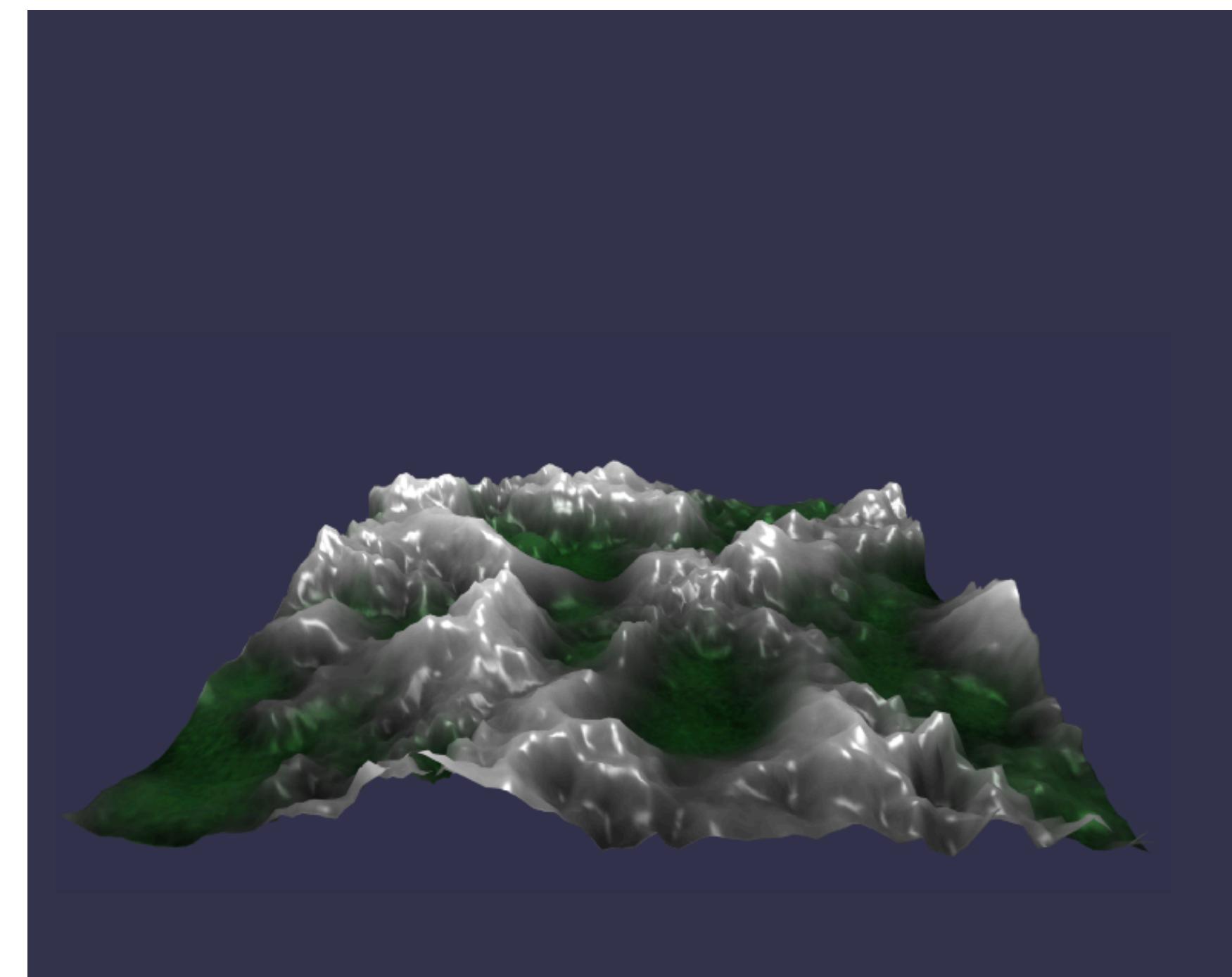
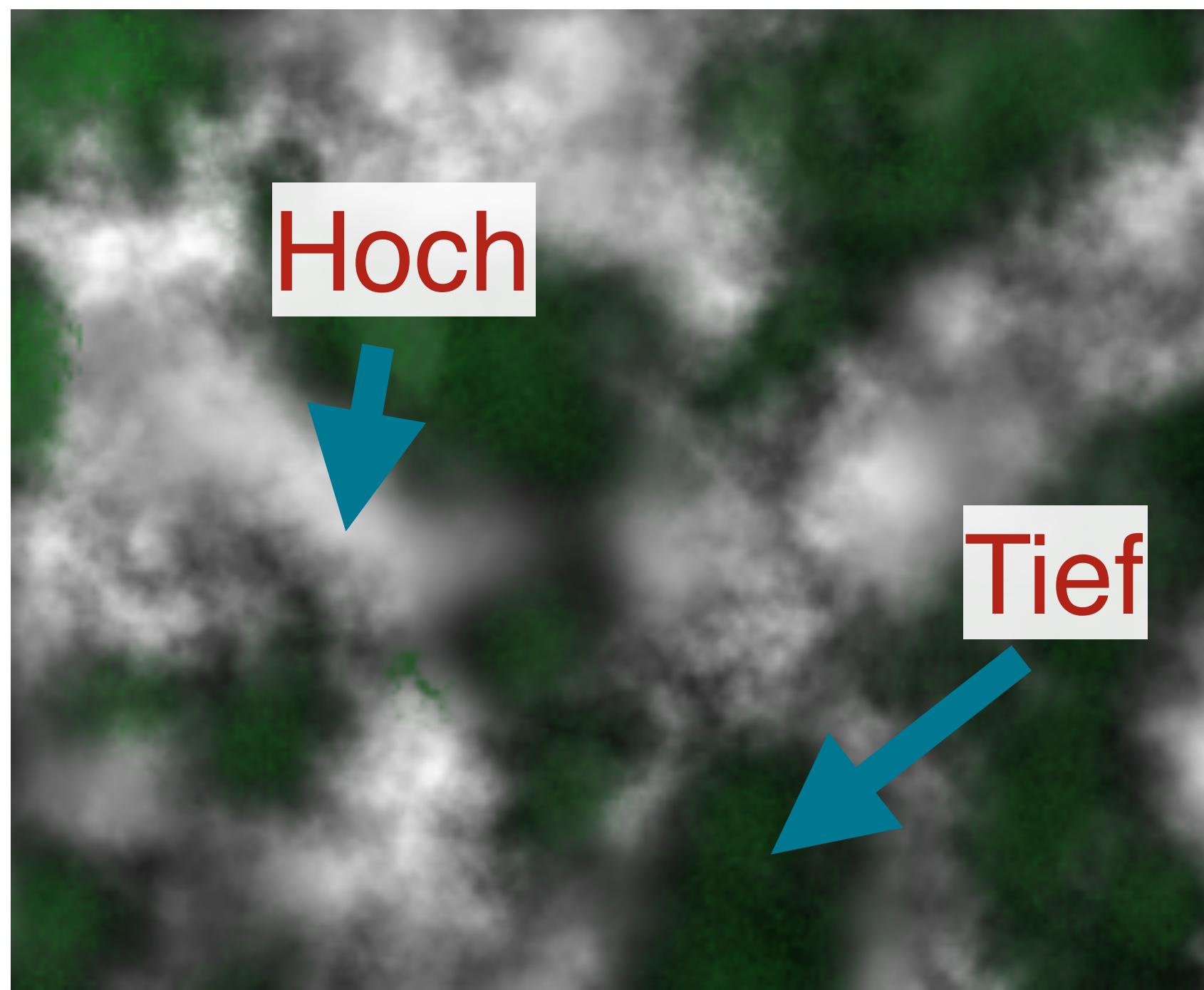
```
const ground = new Mesh.CreateGroundFromHeightMap(  
    "ground",  
    "assets/Heightmap.png",  
    400, //width  
    400, //height  
    100, //subdivide  
    -5, //min height  
    50, // max height  
    scene,  
    false, //updatable  
    () => {  
        //function on success  
    }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

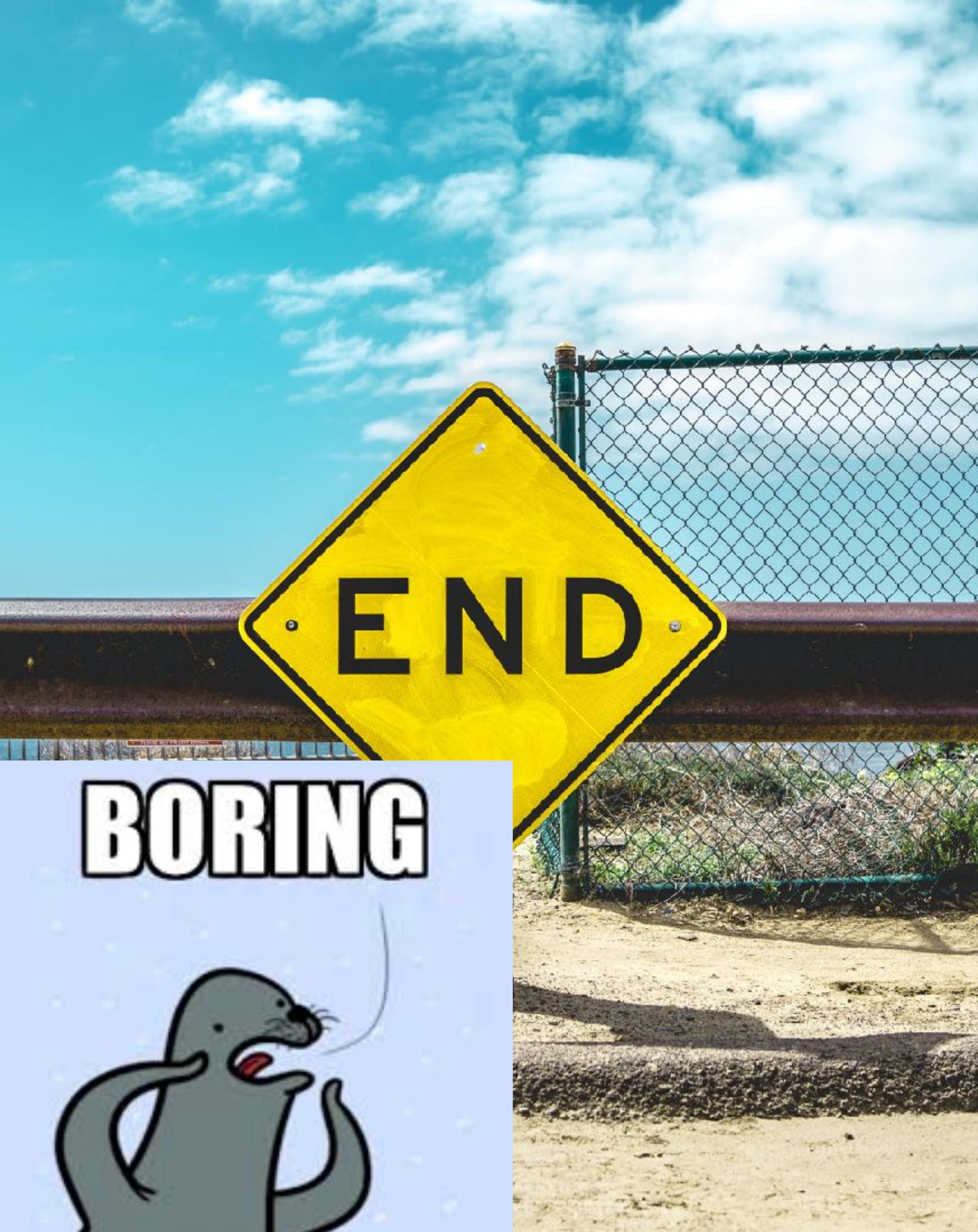
Task 5: A whole new world

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...



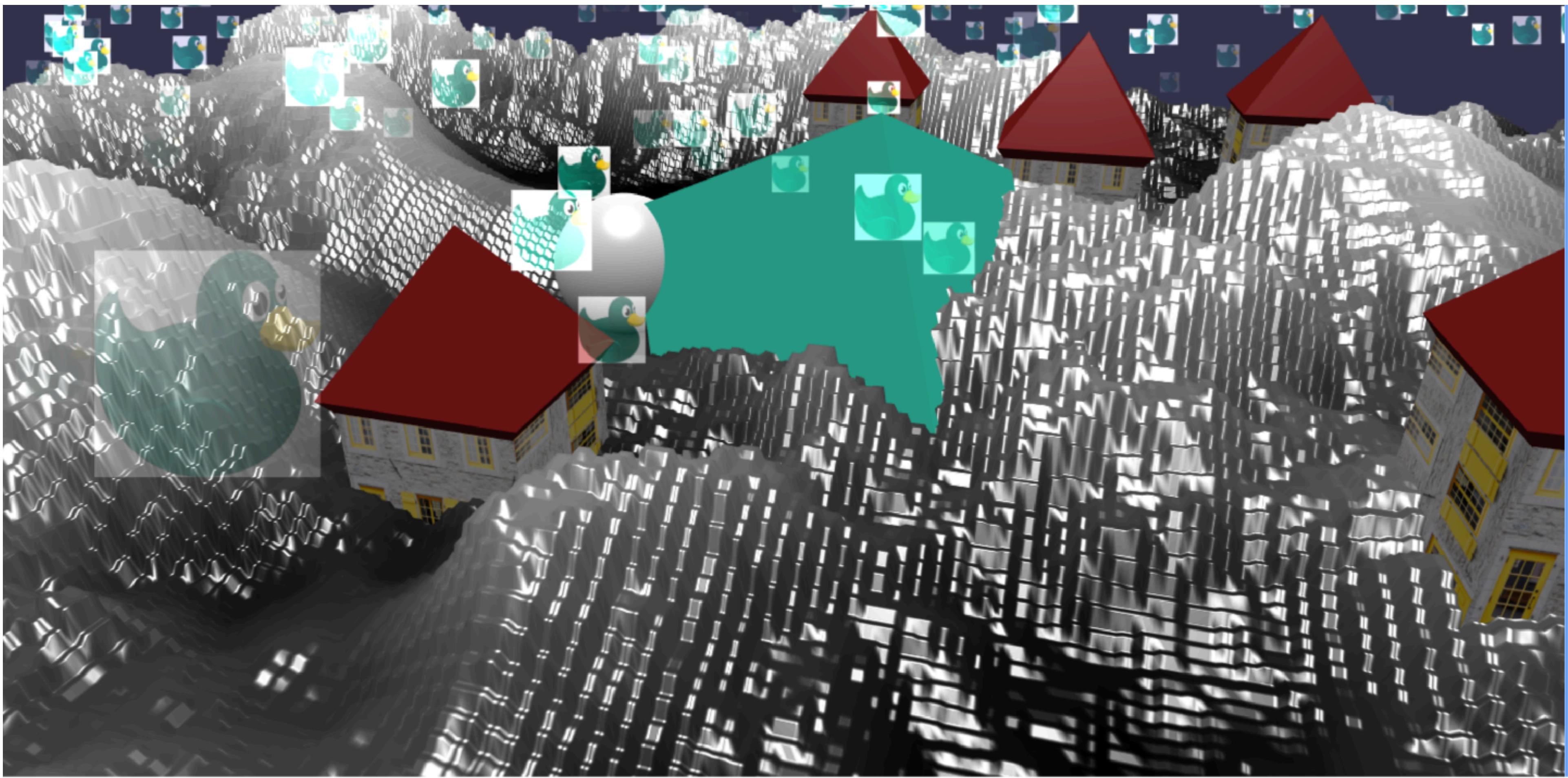
...and start of
the fun part :D



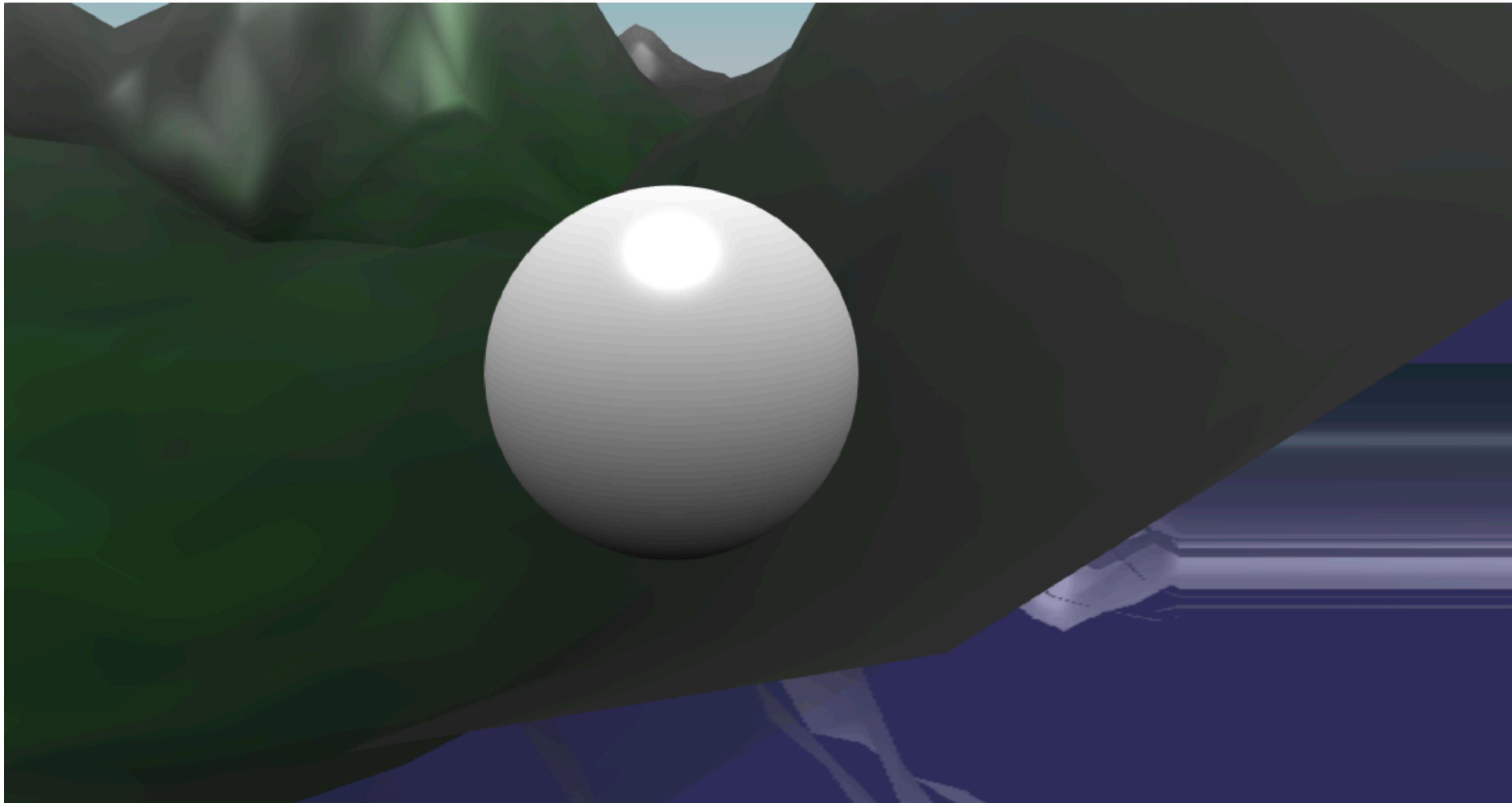
Your idea!

We help you
build it.

Inspiration



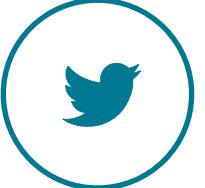
Inspiration (aka Live Demo)



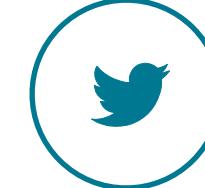
QUESTIONS?

COME TALK TO US! :)



CHRISTINA
 merelyChristina



ANNA
 merelyAnna