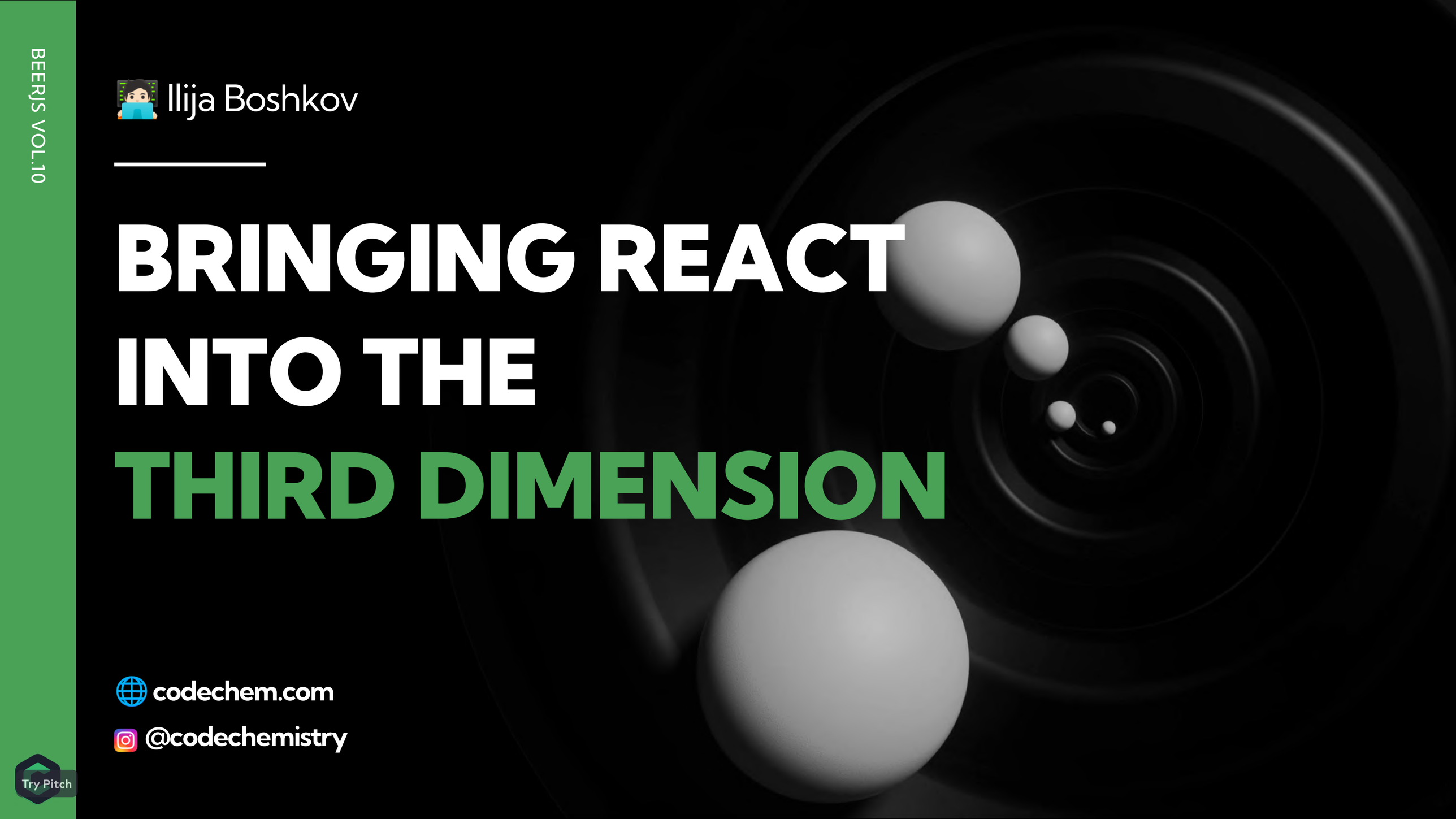
BEERJS VOL.10

👨‍💻 Ilija Boshkov

BRINGING REACT INTO THE THIRD DIMENSION



🌐 codechem.com

@codechemistry

Try Pitch

WHO AM I?

Full-stack developer @ CodeChem since 2017.
Started programming with intent on game development, found myself building engines more than games.
Have worked on every part of the stack, from Mobile to the "Cloud" and everything in between, used JS quite often.

3D and game development was what got me into programming, so here we are again.

I love **DRY** so a lot of this is copy-pasted from my last talk. :)

HEADS UP

I think I bit off more than I can chew...

There's way too much background you need to know before you get into 3D, so consider this 3D rendering 101, and a bit of web + React tie-in.

SO, ANYWAY...

BEERJS VOL.10

👨‍💻 Ilija Boshkov

3D RENDERING 101

...AND MAYBE, BRINGING REACT INTO THE THIRD DIMENSION

🌐 codechem.com

@codechemistry

3D

For when 2 dimensions just don't cut it.

Real-time 3D rendering got popularized in 90s games like DOOM and Wolfenstein.

Our computers are way more powerful now, we can do sci-fi computing compared to the 90s.

Something we built for sharing text documents (HTML) grew into a media-rich experience (JS, Audio, Video, Animations, 3D rendering and more)

HOW IS 3D DONE?

A "tad" bit more complex than just web elements...
I'll try to run you through the basics in a very, very rough, 5-minute crash course.

3D rendering requires deep understanding of proper 3D rendering pipelines, mathematics and how 3D graphics work.

It's definitely harder to represent than 2D graphics.

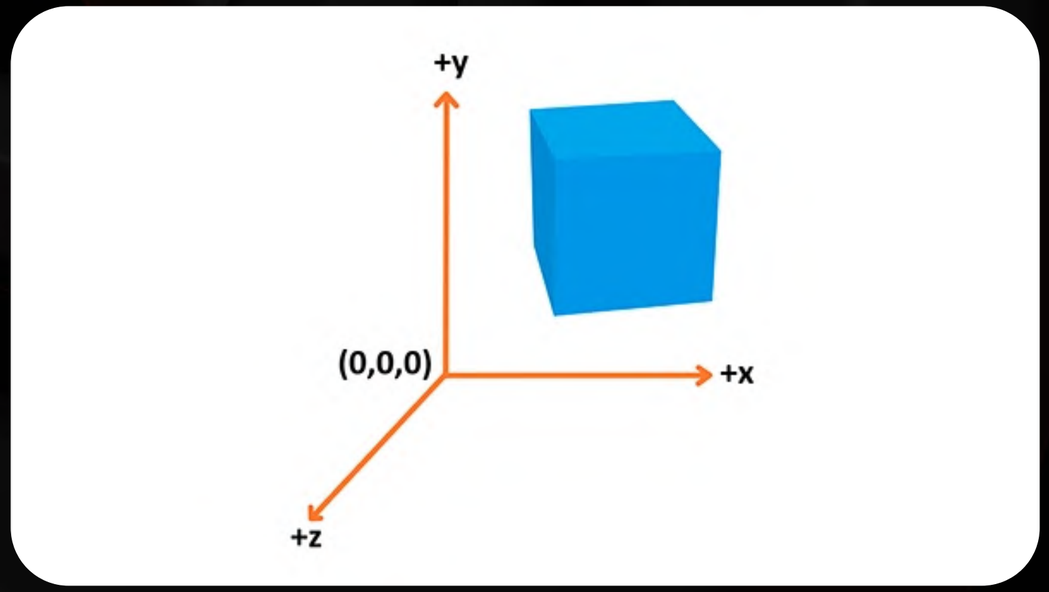
HOW IS 3D DONE - COORDINATE SYSTEM

3D essentially is all about representations of shapes in a 3D space, with a coordinate system used to calculate their position.

WebGL uses the right-hand coordinate system:

- The x axis points to the right
- The y axis points up
- The z axis points out of the screen,

as seen in the diagram



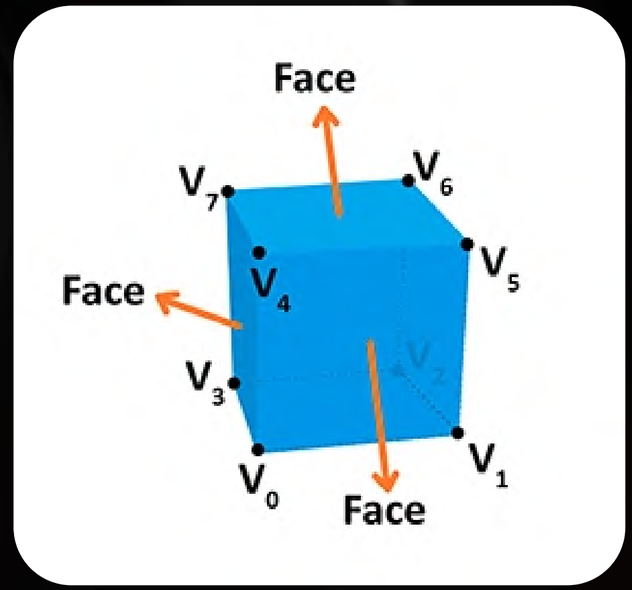
* Reference: MDN - Explaining basic 3D theory

HOW IS 3D DONE - OBJECTS

Different types of objects are built using vertices. A Vertex is a point in space having its own 3D position in the coordinate system and usually some additional information that defines it.

- **Position**: Identifies it in a 3D space (x, y, z).
- **Color**: Holds an RGBA value (R, G and B for the red, green, and blue channels, alpha for transparency — all values range from 0.0 to 1.0).
- **Normal:** A way to describe the direction the vertex is facing.
- **Texture**: A 2D image that the vertex can use to decorate the surface it is part of instead of a simple color.

We use this info tobuild geometry see the example of a cube.

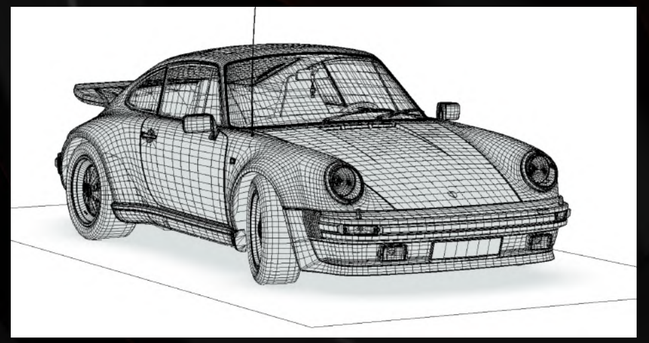


HOW IS 3D DONE - OBJECTS

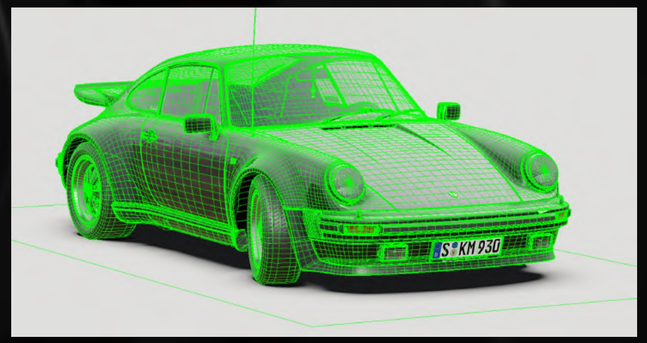
A 3D model through beer-goggles:

Base Mesh

Wireframe



Full Render + Wireframe

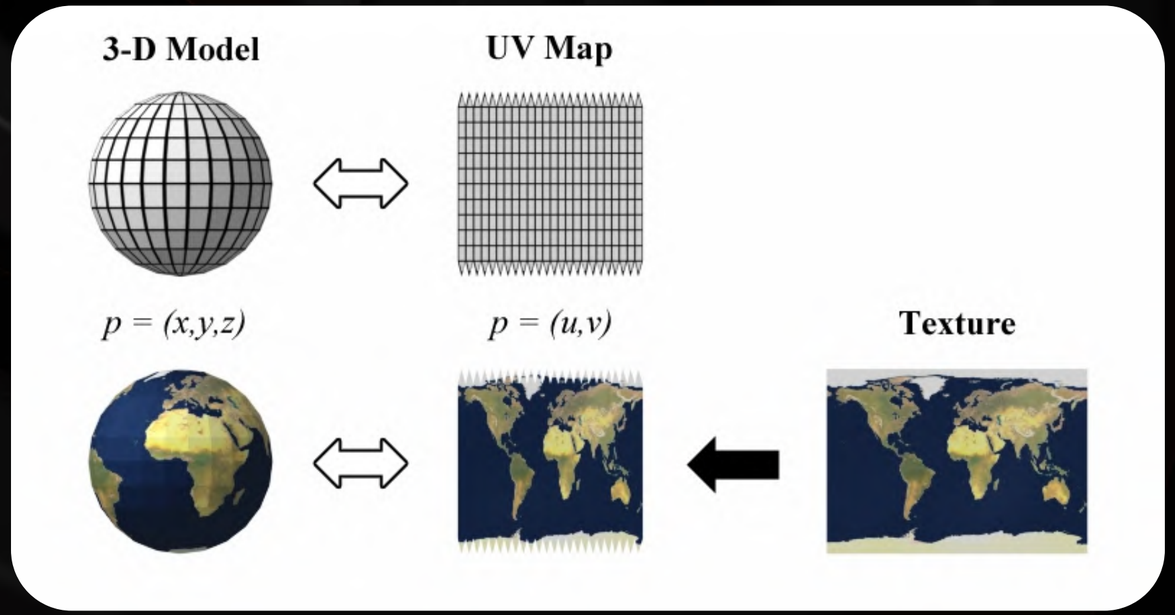


Final Render

* Reference model: SketchFab

HOW IS 3D DONE - TEXTURES

3D Models by themselves are not particularly interesting, we can use 2D textures to add additional information like color, height, normals which help with rendering the object in a more realistic way.



* Reference: MDN - Explaining basic 3D theory

HOW IS 3D DONE – SHADERS

2 Common Types:

- Vertex shaders – determine the position and other properties of each vertex
- Pixel/Fragment shaders – determine the color of each pixel (fragment)

Really deep topic, but in general, like you have code that runs on your CPU, shaders are programs that run on your GPU and calculate either the vertex properties or pixel/fragment properties at the time of rendering.

* Something fun: ShaderToy
* Reference: Khronos Group Shader Wiki
* Reference: MDN GLSL Shaders

HOW IS 3D DONE - APIS

Traditionally, there are different graphics APIs:

- DirectX/Direct3D (Windows, Xbox)
- Metal (MacOS)
- OpenGL (Cross-platform)
- Vulkan (Cross-platform, next-gen successor to OpenGL)
- **WebGL (Web subset of OpenGL, based on OpenGL ES, available in web browsers)**

* Reference: MDN - Explaining basic 3D theory

HOW IS 3D DONE - WEBGL

WebGL is a JS <canvas> API available in almost all major browsers (both desktop and mobile)
You can use it directly, but it's a pain and there a lot of great libraries, rendering engines, game engines and all sorts of tooling around it.

(Most mainstream game engines can target WebGL as a platform - Unity, Unreal Engine, Godot)

* Reference: Khronos Group Shader Wiki

* Reference: MDN GLSL Shaders

HOW IS 3D DONE – MATERIALS

Materials are configurations used by rendering engines to determine how a 3D model (or part of a 3D model) will be rendered.

They determine:

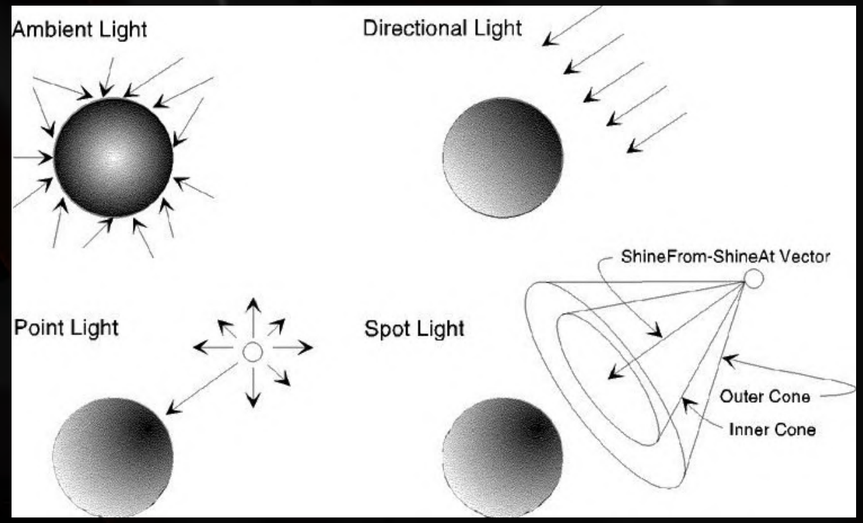
- Which shader to use
- Which properties to pass to that shader
  - Color, Reflectivity, etc. (depends on the shader)
- Which textures to pass to that shader
  - Color maps, height maps, normal maps

HOW IS 3D DONE - LIGHTING

Lights are usually logical objects (invisible) in 3D scenes and they pretty much contain data to describe the type of light and its configuration.

Types of lights in 3D space:

- Point lights
- Directional lights
- Spotlights
- Ambient
- Others

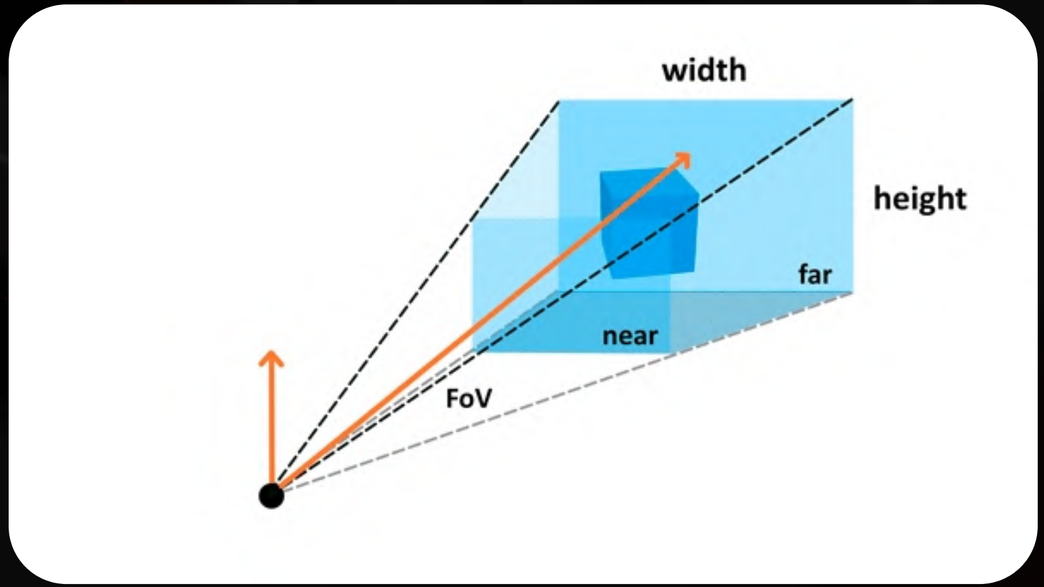


This information is passed to the pixel shader and it calculates how the light will affect the color of the object in that particular pixel (light, shadow, etc).

HOW IS 3D DONE - CAMERA

Cameras are also logical object which describe the point of view to be rendered.

Pixel shaders take them and their properties (position, FOV, view width/height) into account when rendering.



* Reference: MDN - Explaining basic 3D theory.

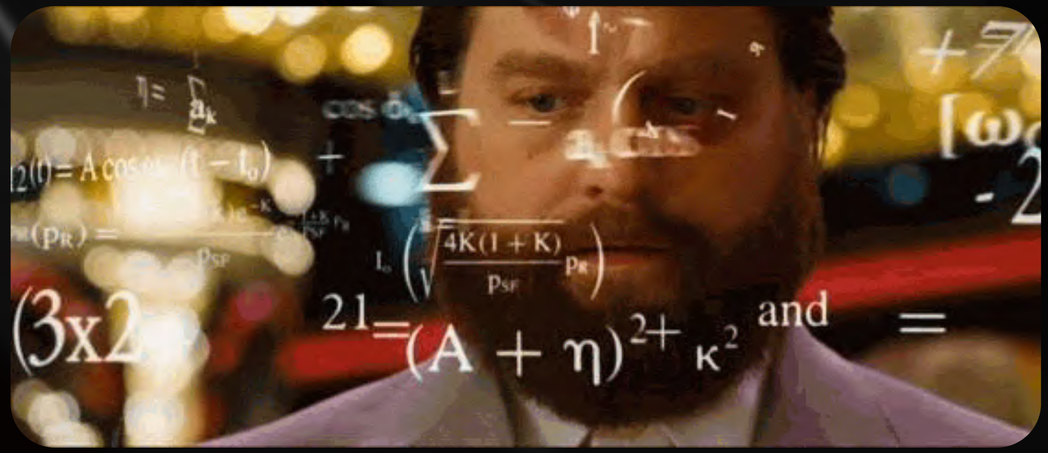
HOW IS 3D DONE - LIGHTING

Reality works with lots of rays and bounces and phyiscs and stuff (aka ray tracing)...

That's tough math to do at (least) 30 times a second.

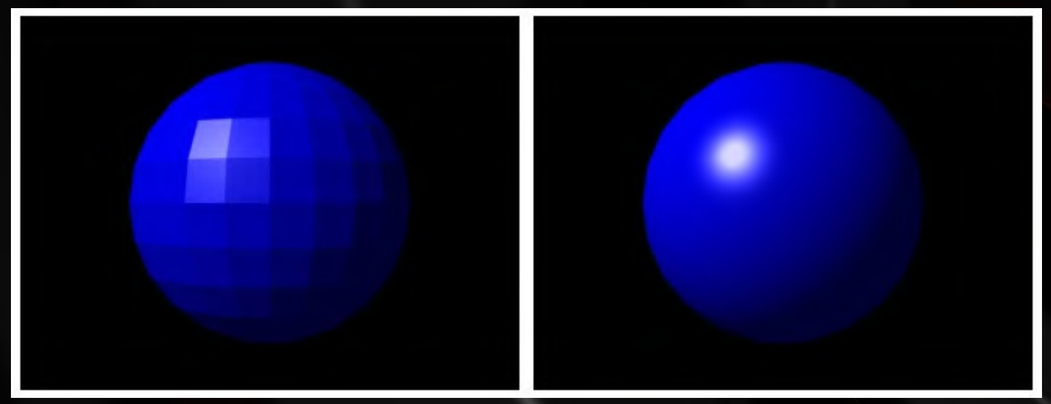
So we do shortcuts, and lots and lots of approximations.

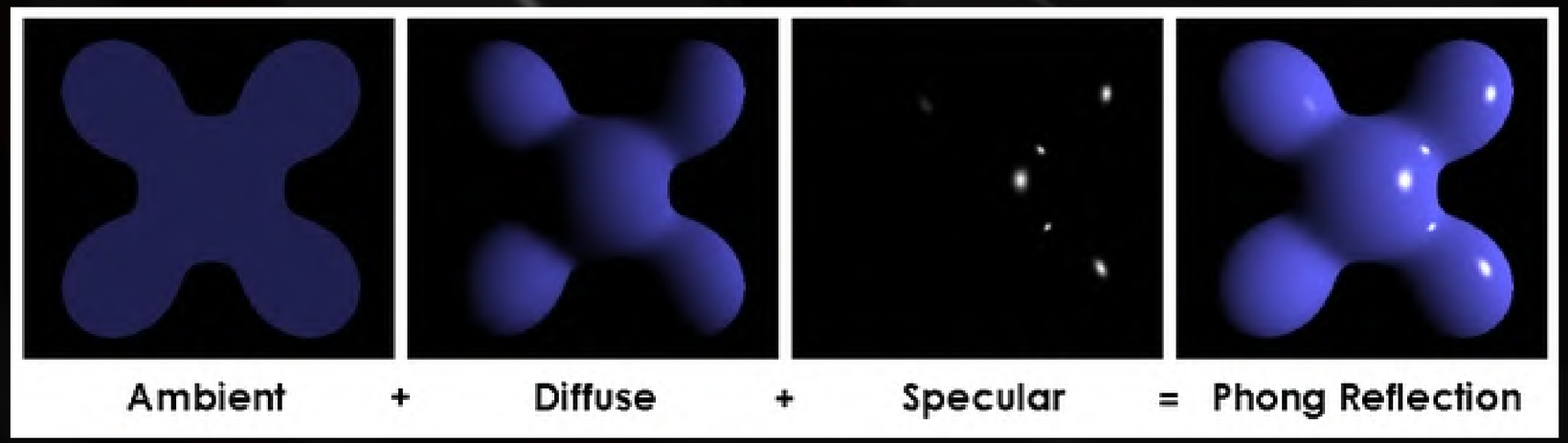
We even sort of do semi-realtime ray tracing now with modern GPUs (RTX).



HOW IS 3D DONE - LIGHTING//PHONG

The oldest trick in the book:





HOW IS 3D DONE - LIGHTING//PHONG

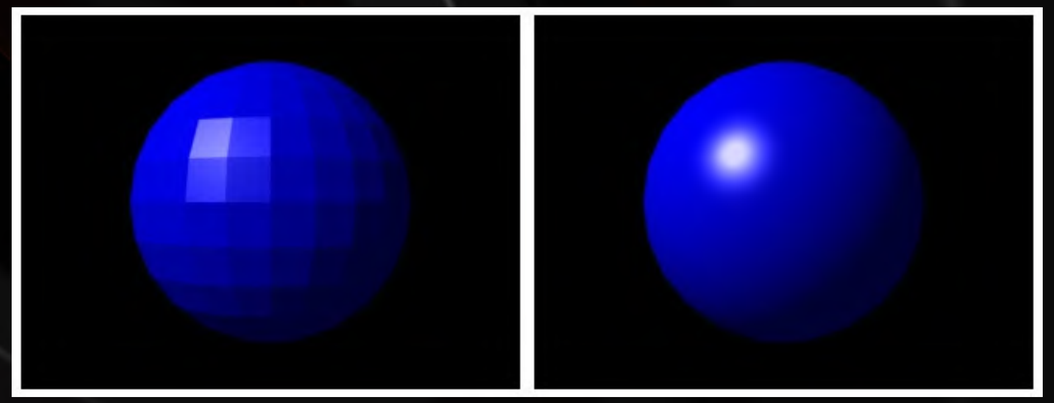
Calculates shading, specularity (highlights) based on normals

Simple code and math

Fast execution

Tends to look a bit plastic-y

Very basic approximation of reality



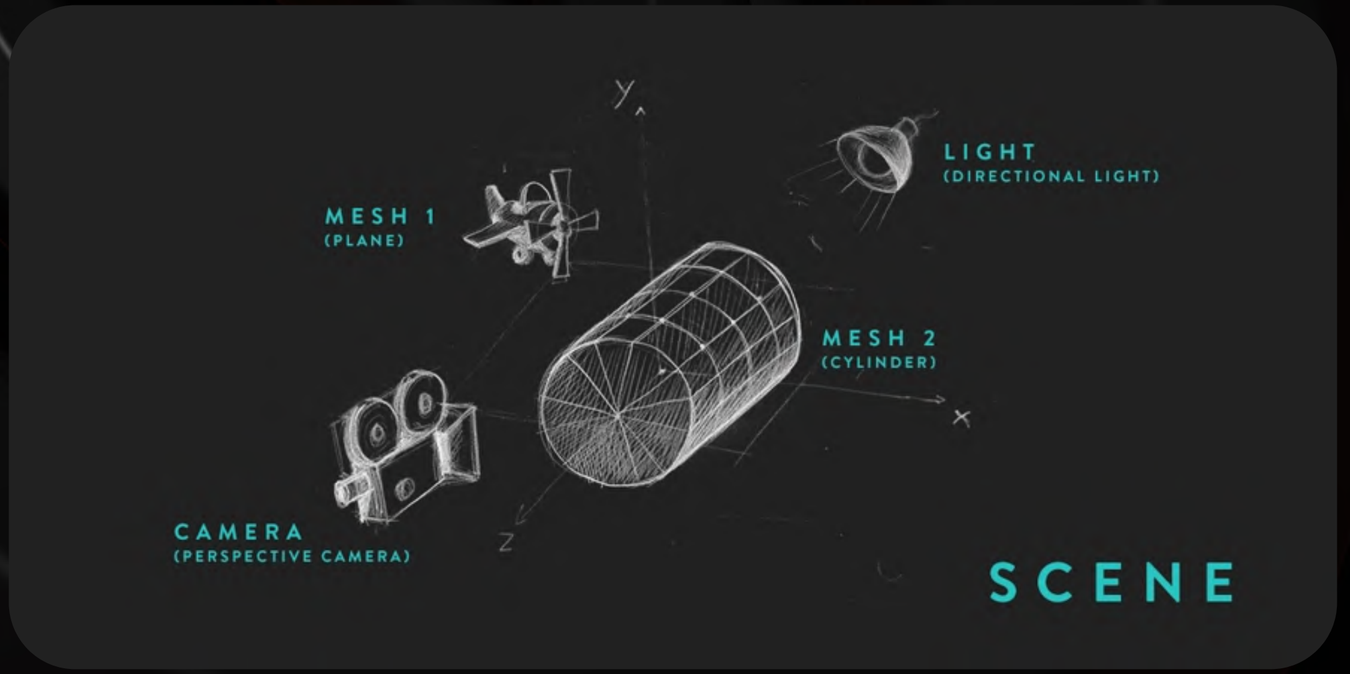
HOW IS 3D DONE – SCENE

Scenes are representations of 3D spaces that combine everything we just went through.

Combines everything we want to show, Lights, Camera, Objects.

Different types of projection

- Perspective
- Orthogonal



ARE WE THERE YET?

THREE.JS

https://threejs.org/
https://threejs.org/docs/

Describe and render 3D scenes using <canvas> and JS.

Since 2010, very stable, very reliable.

Lots of resources.

Alternatives:

- Babylon JS (more modern, better architected)
- Any game engine
- WebGL directly if you're 1337

THREE.JS – LET'S SPIN A CUBE

Make an HTML file.

index.html

```html
<!DOCTYPE html>
<html>
  <head lang="en">
    <meta charset="utf-8">
    <title>My first three.js app</title>
    <style>
      body { margin: 0; }
    </style>
  </head>
  <body>
    <script type="module" src="/main.js"></script>
  </body>
</html>
```

THREE.JS – LET'S SPIN A CUBE

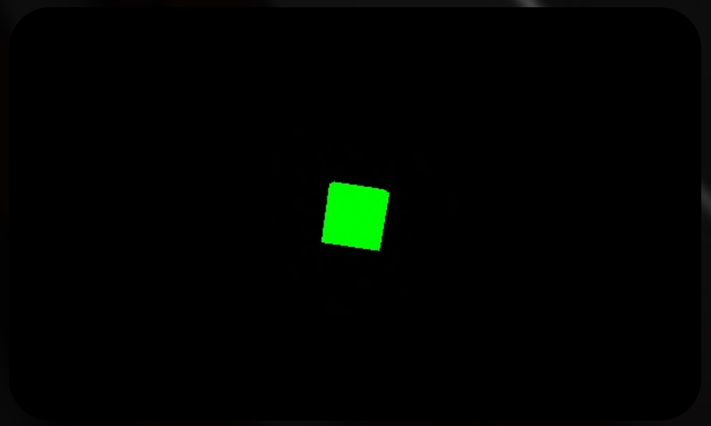
Add the JS (magic powder)

Live Demo

BOOM!

Your very own 3D world.

Population: 1 cube



main.js

```
import * as THREE from 'https://unpkg.com/three/build/three.module.js';

const scene = new THREE.Scene();
const camera = new THREE.PerspectiveCamera( 75, window.innerWidth/window.innerHeight, 0.1,
1000 );

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

const geometry = new THREE.BoxGeometry( 1, 1, 1 );
const material = new THREE.MeshBasicMaterial( { color: 0x00ff00 } );
const cube = new THREE.Mesh( geometry, material );
scene.add( cube );

camera.position.z = 5;

function animate() {
  requestAnimationFrame( animate );

  cube.rotation.x += 0.01;
  cube.rotation.y += 0.01;

  renderer.render( scene, camera );
}

animate();
```

WAIT... WASN'T THIS TALK ABOUT REACT?

It's been a while since I started talking. Now that we mastered Three.js let's switch to talking about React for a second.

React has this cool thing called JSX, it's pseudo syntax for JS, and the compiler replaces calls to <YourComponent /> with React.createElement().

This scene thing seems like it would be nice to express via JSX, no?

@REACT-THREE/FIBER

Someone else thought so too, so we have Fiber, a set of wrapper components and hooks that build and manage three.js scenes "behind-the-scenes" to enable you to use the React ecosystem to manage state, when to update, what to update, etc.

BabylonJS also has a JSX implementation (react-babylonjs).

@REACT-THREE/FIBER – LET'S SPIN SOME CUBES

Seems a lot nicer than our previous code, right?

- Easier to use events
- No overhead over pure three.js
- Very popular ecosystem

```
function Box(props) {
  // This reference gives us direct access to the THREE.Mesh object
  const ref = useRef()
  // Hold state for hovered and clicked events
  const [hovered, hover] = useState(false)
  const [clicked, click] = useState(false)
  // Subscribe this component to the render-loop, rotate the mesh every frame
  useFrame((state, delta) => (ref.current.rotation.x += delta))

  return (
    <mesh
      {...props}
      ref={ref}
      scale={clicked ? 1.5 : 1}
      onClick={(event) => click(!clicked)}
      onPointerOver={(event) => hover(true)}
      onPointerOut={(event) => hover(false)}>
      <boxGeometry args={[1, 1, 1]} />
      <meshStandardMaterial color={hovered ? 'hotpink' : 'orange'} />
    </mesh>
  )
}
```

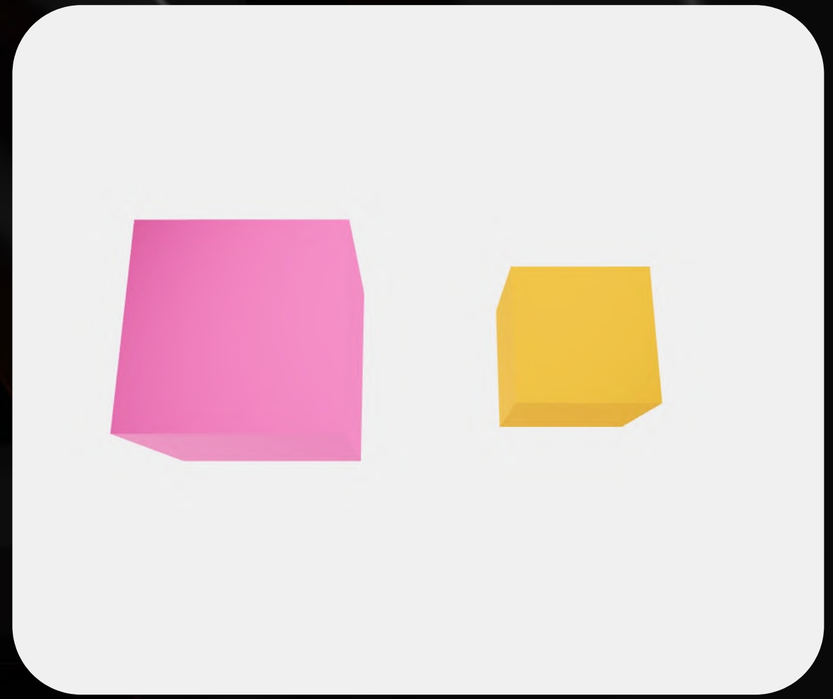
@REACT-THREE/FIBER – LET'S SPIN SOME CUBES

LIVE DEMO

BOOM!

Your very own reactive 3D world.

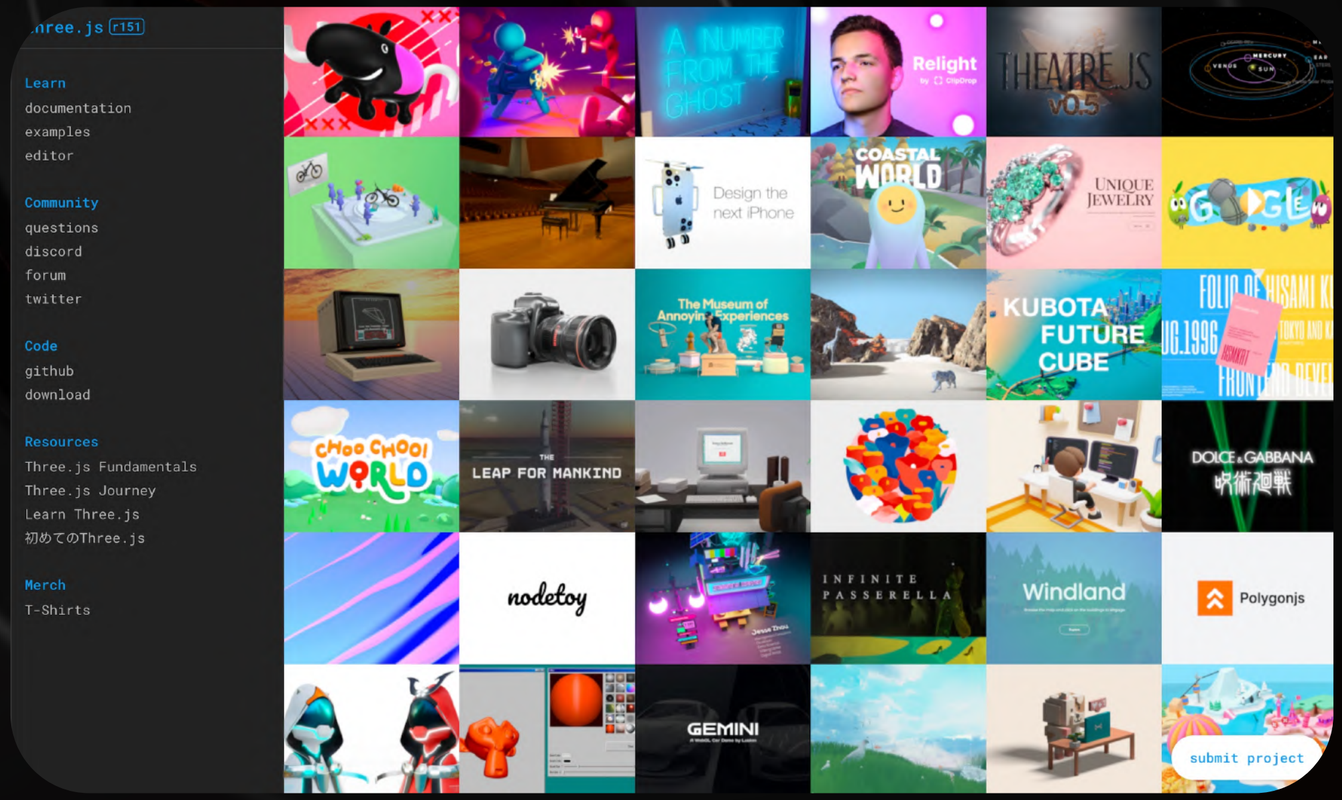
Population: **2 cubes**



COOL, SPINNING CUBES... NOW WHAT?

Check out the three.js home page for lots of cool things people have done.

Lots of inspiration there.



ONE LAST THING BEFORE I SHUT UP

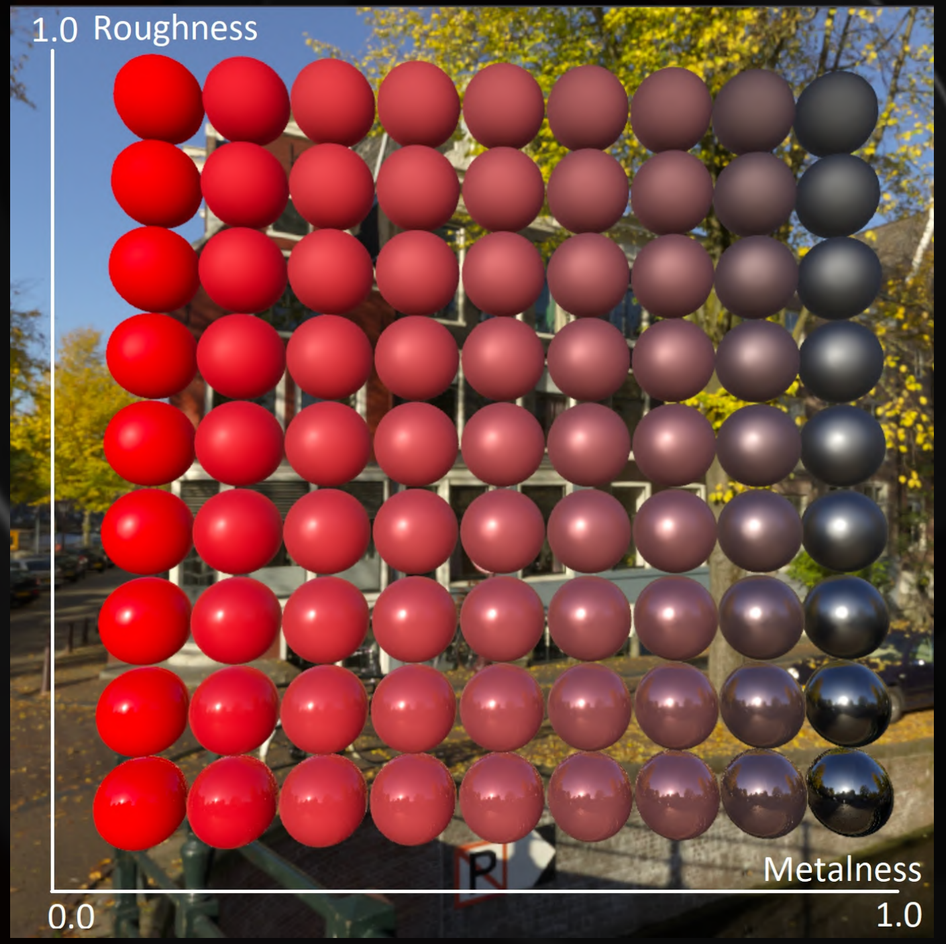
Let's explore some more modern rendering/lighting techniques supported by three.js.

REALISTIC SHADING AND RENDERING OF MATERIALS

PBR (Physically-Based Rendering)

- Modern technique of simulating realistic material properties in a standard 3D pipeline
  - Color
  - Metallic
  - Roughness
  - Reflectance
- We can combine textures and map them to specific properties of materials

REALISTIC SHADING AND RENDERING OF MATERIALS



REALISTIC SHADING AND RENDERING OF MATERIALS

DEMO TIME

Something I built just for you ❤️!

Fully open source, feel free to extend it, learn from it.

src:

https://github.com/codechem/beerjs-3d

bin:

https://beerjs10.codechem.com/

Scan & see

THANK YOU
FOR YOUR ATTENTION

Any Questions ? 💬