Creating a Reflective Glass Cube in Three.js

Matt Park Follow 3 min read · Mar 5, 2024

I've been thinking that I should start adding more assets other than the 3d component itself to each of the components I'm making.

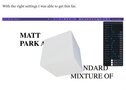
Started with a bold text on the background, the plan is to create a glass cube on top of the text. Note that the tests here are pure react component, not STM. Component from @react-three/drei, source/docs/lib/components/Text/index.tsx

Loading Canvas with React Components

One of the key steps in loading react components with Canvas (and other three.js components) is setting the size position of your react component. Based on research by react component's positioning must be set to absolute in order to have the canvas to load along with the react component.

Nice. Getting there.

With the right settings I was able to get this far.



But something seemed off. Transmission material wasn't looking anything close to being reflective.

I had a planeGeometry installed below the material just to check that our material was still working as expected.

So it turns out that since our react component is actually not on top of our canvas, meshPhysicalMaterial can't see through it. Obvious, but still wanted to give it a try and see if it would work. (Hint, this doesn't work with text. Form this as well)

Using Text Geometry

To be able to get a reflection from our text, we need our text to be also a mesh as well.

So we can achieve this via https://github.com/pmndrs/drei/tree/master/docs/geometries/TextGeometry.md wrapped around in drei.

Setting a text behind our cube,

```
<Text
    position={[0, 0, -10]}
    scale={[10,10,1]}
    >
    Hello world
    </Text>
```

we can get something like this:

I've also adjusted my MeshPhysicalMaterial settings to the following:

```
<MeshPhysicalMaterial
    transmission={1}
    roughness={0}
    thickness={0.5}
    />
```

(Note that the same effect is achievable by using MeshPhysicalMaterial)

Setting Text

Let's finish setting text on the correct position.

```
<Text position={[0.5, 0, 0]} color={blueColor}
    font={myFont} scale={0.2}>MOMENT</Text>
<Text position={[0.5, -0.3, 0]} color={blueColor}
    font={myFont} scale={0.2}>OF</Text>
<Text position={[0.5, -0.6, 0]} color={blueColor}
    font={myFont} scale={0.2}>EXPRESSION</Text>
```

Background

Get Matt Park's stories in your inbox.

Join Medium for free to get updated articles from Matt Park.

Enter your email Subscribe

We've set `Background` to get the correct lighting for our mesh, but notice how we're also getting the background from the environ.

We can overcrite the background while still having the environment with the right setting by simply using color component with background attached.

```
<color args={['white']} attach="background" />
```

Controls

Now, we want the background to be fixed, having mouse movement only trigger the movement / control of our glass cube.

Presentation Controls https://github.com/pmndrs/drei/tree/master/docs/lights-and-shadows/PresentationControls.md is what we want to achieve this effect.

One of the upsides for using presentation controls is that we can have control for each object.

```
<PresentationControls>
    <mesh ref={cube} position={[0, 0,0]} rotation={[Math.random() * Math.PI,
        material-envMapIntensity={0.8} />
    <mesh ref={cube} position={[0, 0,0]} rotation={[Math.random() * Math.PI,
        material-envMapIntensity={0.8} />
    </PresentationControls>
```

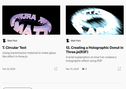
Notice how we're creating around 0.5x0.5

While implementing this, I realized that in PresentationControls, any mesh that are not centered would have offset and would always be rotated around the center axis (x, 0,0)

Applying Design & Overlay

This section on about applying react components to on top of our canvas. There are no hard restrictions to merging our canvas with plain react component as long as the react component is an overlay and defined after the canvas.

Applied some design, here's what I have so far



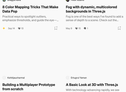
Rotation on Mouse Movement

I wanted the cubes to rotate based on my mouse movement. This needs to happen on each frame (so called `useFrame` and we can use state pointer to get the location of our pointer.

```
useFrame(state, delta) => {
    const x = state.pointer.x
    const y = state.pointer.y * Math.PI; // Rotate around x axis based on
    mesh.current.rotation.x = y * Math.PI;
}
```

Future work

- Rotation movement depending on mouse position is kind of lame, I think we can do better
- Design. Had some extra component that I wanted to make which I will leave for my future work for now.



Written by Matt Park
0 Followers · 1 Following
Live Blending Mountains for events. Contact me via instagram @mattpparkmixtures

Follow

No responses yet

Write a response

What are your thoughts?

More from Matt Park

Matt Park
How I made my 3d Portfolio Website.
As I left school in December, what good
opportunity to do a project that would prepare
Nov 5 · 1

Matt Park
6. (Unfinished) Inflated Text
In progressive
Feb 25, 2024

Matt Park
7. Circular Text
Using transformation matrix to make glass
like effect in Blender
Feb 18, 2024

Matt Park
12. (Creating a Holographic Donut in Three.js)(EP.1)
Using Blender to create a donut that contains
holographic effect using GLSL
Apr 9, 2024 · 1

See all from Matt Park

Recommended from Medium

Filippo Ranza
6 Color Mapping Tricks That Make Data Pop
Color mapping can highlight patterns,
emphasize thresholds, and guide the eye —

Vitor Mesaro
Pop with dynamic, multicolored backgrounds in Three.js
Learn how to use Three.js shaders to add a
sense of depth to scenes. Check out this...

Intelligencehub

Building a Multiplayer Prototype from scratch
Hey everyone!

Original Tetrisel
A Basic Look at 3D with Three.js
Hello and welcome to my tutorial series on 3D
Graphics with Three.js! In this tutorial, we will
cover...

Sep 19 · 1 Sep 20

Modexa

5 Reasons SolidJS Became Devs' Unexpected Darling

A hands-on look at how fine-grained reactivity, zero VDOM, and a pragmatic DX...

Sep 14 150 1

QuarkAndCode

Visualizing Data in Jupyter Matplotlib, Seaborn, Plotly &...

How to visualize data in Jupyter with Matplotlib, Seaborn, Plotly, and bqplot—...

Oct 14 11

See more recommendations