Textbuffergeometry

Using typeface font

Facetype.js

Or use fonts provided by three.js

Put them in the static folder

We grab our fonts from node modules/three/examples/fonts etc. grab with license and pull into static folder.

Then we want to try and pull them in with a fontLoader. Here is how setup looks:

const fontLoader = new FontLoader();

fontLoader.load("/fonts/helvetiker\_regular.typeface.json", (font) => {

  const textGeometry = new TextGeometry("Hello Three.js", {

    font: font,

    size: 0.5,

    height: 0.2,

    curveSegments: 12,

    bevelEnabled: true,

    bevelThickness: 0.03,

    bevelSize: 0.02,

    bevelOffset: 0,

    bevelSegments: 5,

  });

});

const fontLoader = new FontLoader();

fontLoader.load("/fonts/helvetiker\_regular.typeface.json", (font) => {

  const textGeometry = new TextGeometry("Hello Three.js", {

    font: font,

    size: 0.5,

    height: 0.2,

    curveSegments: 12,

    bevelEnabled: true,

    bevelThickness: 0.03,

    bevelSize: 0.02,

    bevelOffset: 0,

    bevelSegments: 5,

  });

});

We are going to try and center the text with bounding

The bounding is an information associated with the geometry that tells what space is taken by that geometry. It can be a box or sphere.

Basically it is the box or circle around a character or object that the world interacts with.

It helps Three.js calculate if the object is on the screen with (frustum culling)

We can uses the measures to recenter items.

Frustum culling is about rendering or not rendering an object.

Basically you don’t want to render things that are behind the camera. Can lead to performance issues.

By default three.js uses sphere bounding.

We can calculate the box bounding with computeBoundingBox()

  textGeometry.computeBoundingBox();

  console.log(textGeometry.boundingBox);

we geta box3 which is an invisible box that surrounds our item.

Now to center it instead of moving the mesh we are going to translate the object.

  textGeometry.computeBoundingBox();

  textGeometry.translate(

    -textGeometry.boundingBox.max.x \* 0.5,

    -textGeometry.boundingBox.max.y \* 0.5,

    -textGeometry.boundingBox.max.z \* 0.5

  );

It is not fully centered because of the bevel thickness and bevel size.

There is a more simple way of centering by using the center() method.

Now we are going to use a MeshMatcapMaterial instead of the basic material or wireframe

Now lets add donuts to the screen for background stuff. (torus)

  for (let i = 0; i < 100; i++) {

    let donutMaterial = new THREE.MeshNormalMaterial();

    let donutGeometry = new THREE.TorusBufferGeometry(0.3, 0.2, 20, 45);

    let donut = new THREE.Mesh(donutGeometry, donutMaterial);

    donut.position.set(

      Math.random() \* 20 - 10,

      Math.random() \* 20 - 10,

      Math.random() \* 20 - 10

    );

    donut.rotation.set(

      Math.random() \* 20 - 10,

      Math.random() \* 20 - 10,

      Math.random() \* 20 - 10

    );

    scene.add(donut);

  }

To Bruno – this is the wrong way haha. But it works!

Bruno says to do (Math.random() - 0.5)

Instead which is much smarter… it normalizes the value output by math.random.

Now add opactity to the material

    donutMaterial.transparent = true;

    donutMaterial.opacity = Math.random();

now add scale to donuts

    let donutSize = Math.random() \* 5;

    donut.scale.set(donutSize, donutSize, donutSize);

have to pass same scaling to each x y and z or will get distorted donuts.

Change rotation to

donut.rotation.set(

      Math.random() \* Math.PI,

      Math.random() \* Math.PI,

      donut.rotation.z

    );

Only need to rotate two axes to get all movement.

The main problem is that this is not optimized. Add this before and after the loop to test how long it took.

  console.time("donuts");

<code>

  console.timeEnd("donuts");

Then how do we optimize?

Well, we can use the same material and geometry for multiple meshes. So, take the instantiation out side of the for loop and reuse the material and geometry. So final code looks like:

console.time("donuts");

  let donutMaterial = new THREE.MeshNormalMaterial();

  donutMaterial.transparent = true;

  donutMaterial.opacity = Math.random();

  let donutGeometry = new THREE.TorusBufferGeometry(0.3, 0.2, 20, 45);

  for (let i = 0; i < 100; i++) {

    let donut = new THREE.Mesh(donutGeometry, donutMaterial);

    donut.position.set(

      (Math.random() - 0.5) \* 20,

      (Math.random() - 0.5) \* 20,

      (Math.random() - 0.5) \* 20

    );

    donut.rotation.set(

      Math.random() \* Math.PI,

      Math.random() \* Math.PI,

      donut.rotation.z

    );

    let donutSize = Math.random() \* 3;

    donut.scale.set(donutSize, donutSize, donutSize);

    scene.add(donut);

  }

  console.timeEnd("donuts");