In App.js

import React, { Component } from 'react'

import Project from './component/Project'

import './App.css';

export class App extends Component {

  constructor(props){

    super(props);

    this.canvasRef= React.createRef();

  }

  componentDid

  render() {

    return (

      <div>

        <canvas className="Webgl" ref={this.canvasRef} ></canvas>

        <nav>

        <a href='/'>Sphere</a>

          <ul>

            <li>good</li>

            <li>bad</li>

            <li>things</li>

          </ul>

        </nav>

        <h1 className="title">goood analysis</h1>

        <Project canvasRef={this.canvasRef} />

      </div>

    )

  }

}

export default App

in project.js

import React, { Component } from 'react'

import \* as THREE from 'three';

import {OrbitControls} from 'three/examples/jsm/controls/OrbitControls'

export class Project extends Component {

    //constructor(props){

      //  super(props);

       // };

    //shape

    componentDidMount(){

        //canvas ref from App.js

        this.canvasRef=this.props.canvasRef;

        //create scene

        this.scene=new THREE.Scene();

        //create the shape by providing geometry aand material

        const geometry=new THREE.SphereGeometry(3,64,64);

        const material =new THREE.MeshStandardMaterial({

            color:"#00ff83",

        });

        const mesh =new THREE.Mesh(geometry,material);

        this.scene.add(mesh)

        //Sizes of the view port

        this.size={

            width:window.innerWidth,

            height:window.innerHeight

        }

        //Light

        this.light= new THREE.PointLight(0xffffff,1,100);

        this.light.position.set(0,10,10)//[1(-num)=left],[1(+num)=right],[2(-num)=goes to down],[2(+num)=top],[3(-num)=back],[3(+num)=front]

        this.scene.add(this.light);

        //camera

        this.camera=new THREE.PerspectiveCamera(45,this.size.width/this.size.height,0.1,100)

        //camera.position

        this.camera.position.z=20

        this.scene.add(this.camera)

        //Renderer

        this.renderer=new THREE.WebGLRenderer({

            canvas:this.canvasRef.current

        });

        this.renderer.setSize(this.size.width,this.size.height);

        this.renderer.setPixelRatio(2)

        this.renderer.render(this.scene,this.camera);

        //controles

        const controls =new OrbitControls(this.camera,this.canvasRef.current)

        controls.enableDamping=true

        controls.enablePan=false

        controls.enableZoom=false

        controls.autoRotate=true

        controls.autoRotateSpeed=5

        //Resize

        window.addEventListener('resize',()=>{

            //update Sizes

            console.log(window.innerWidth);

            this.size.width=window.innerWidth;

            this.size.height=window.innerHeight;

            //update the width of camera

            this.camera.aspect=this.size.width/this.size.height;

            this.camera.updateProjectionMatrix();

            this.renderer.setSize(this.size.width,this.size.height)

        })

        const loop=()=>{

            this.renderer.render(this.scene,this.camera)

            window.requestAnimationFrame(loop);

            controls.update()

        }

        loop()

    }

    render() {

        return (

      <div>

      </div>

    )

  }

}

export default Project

In App.css

\*{

  margin:0;

  padding:0;

  box-sizing: border-box;

}

body,html{

  overflow-x: hidden;

}

.Webgl{

  position: absolute;

  top:0;

  left:0;

  z-index:1;

}

nav{

  color:white;

  z-index:2;

  position:relative;

  padding:4rem 8rem;

  display: flex;

  justify-content: space-between;

}

nav a{

  text-decoration: none;

  color:white;

  font-weight: bold ;

}

nav ul{

  list-style: none;

  display: flex;

  gap:4rem;

}

.title{

  color:white;

  z-index:2;

  position: absolute;

  font-size: 3rem;

  left:50%;

  top:75%;

  transform: translate(-50%,-75%);

}

Cubes rotating scaling and transforming

import React, { Component } from 'react'

import \* as THREE from 'three';

// import {OrbitControls} from 'three/examples/jsm/controls/OrbitControls';

import gsap, { Expo } from 'gsap';

export class NewProject extends Component {

    shape=()=>{

        const geometry=new THREE.BoxGeometry(1,1,1);

        const material=new THREE.MeshLambertMaterial({

            color:"0xf7f8f9",//rgba(0,123,0,0.9);

        });

        this.meshX=-10;

        for(let i=0;i<15;i++){

            this.mesh =new THREE.Mesh(geometry,material);

            this.mesh.position.x=(Math.random()-0.5)\*10;

            this.mesh.position.y=(Math.random()-0.5)\*10;

            this.mesh.position.z=(Math.random()-0.5)\*10;

            this.scene.add(this.mesh);

            this.meshX+=1;

        }

        // this.scene.add(this.mesh);

    }

    componentDidMount(props){

        this.canvasRef=this.props.canvasRef.current;

        this.sizes={

            'width':window.innerWidth,

            'height':window.innerHeight

        }

        this.scene=new THREE.Scene();

        //to start the animation on the particular object when click on the particular objects

        //by based on the raycaster

        var raycaster =new THREE.Raycaster();

        var mouse=new THREE.Vector2();

        this.shape();

        this.light=new THREE.PointLight(0xffffff,1,1000);

        this.light.position.set(0,0,0);

        this.scene.add(this.light);

        this.light=new THREE.PointLight(0xffffff,2,1000);

        this.light.position.set(0,0,25);

        this.scene.add(this.light);

        this.camera=new THREE.PerspectiveCamera(45,this.sizes.width/this.sizes.height,0.1,100);

        this.camera.position.z=10;

        this.scene.add(this.camera);

        this.renderer=new THREE.WebGLRenderer({

            canvas:this.canvasRef,

        });

        this.renderer.setClearColor("#e7e7e7")

        this.renderer.setSize(this.sizes.width,this.sizes.height);

        this.renderer.setPixelRatio(2);

        this.renderer.render(this.scene,this.camera);

        //comtrolers

        // const controls=new OrbitControls(this.camera,this.canvasRef);

        // controls.enableDamping=true

        // controls.enablePan=false

        // controls.enableZoom=false

        // controls.autoRotate=false;

        // controls.autoRotateSpeed=0;

        //resize

        window.addEventListener('resize',()=>{

            //update sizes

            this.sizes.width=window.innerWidth;

            this.sizes.height=window.innerHeight;

            //update the width of camera

            this.camera.aspect=this.sizes.width/this.sizes.height;

            this.camera.updateProjectionMatrix();

            this.renderer.setSize(this.sizes.width,this.sizes.height);

        })

        const loop=()=>{

            this.renderer.render(this.scene,this.camera);

            window.requestAnimationFrame(loop);

            // controls.update();

        }

        loop();

        //to pause the animation

        //this.t1=new gsap.timeline({paused:true});

        //to play animation

        //this.t1.play();

        //event resuming the animation

        // document.body.addEventListener("click",()=>{

        //     this.t1.play();

        // })

        const onMouseMove =(e)=>{

            e.preventDefault();

            mouse.x=(e.clientX/window.innerWidth)\*2-1;

            mouse.y= -(e.clientY/window.innerHeight)\*2+1;

            raycaster.setFromCamera(mouse,this.camera);

            var intersects=raycaster.intersectObjects(this.scene.children,true);

            for(var i=0;i<intersects.length;i++){

                // intersects[i].object.material.color.set(0xff0000);

                //this.t1= new gsap.timeline({paused:true}).delay(0.3);

                this.t1= new gsap.timeline().delay(0.3);

                this.t1.to(intersects[i].object.scale,1,{x:2,ease:Expo.easeOut});

                this.t1.to(intersects[i].object.scale,0.5,{x:0.5,ease:Expo.easeOut});

                this.t1.to(intersects[i].object.position,0.5,{x:2,ease:Expo.easeOut});

                this.t1.to(intersects[i].object.rotation,0.5,{y:Math.PI\*0.5,ease:Expo.easeOut},"=-1.5");//"=-1.5 will it going to happen in conjucntion 1.5sec efore it normaly would do"

            }

            //when we hover aor reload object turns red

        }

        window.addEventListener("mousemove",onMouseMove);

    }

    render() {

        return (

          <div>

        </div>

        )

    }

}

export default NewProject