3D-SaaS Website

M. Fachry Dwi Handoko 5025201159
Faraihan Rafi Adityawarman 5025211074
Pascal Roger Junior Tauran 5025211072

Website Features

- Intro page with 3D elements that rotate as you scroll down.
- Home page with 3D elements that you can interact with.

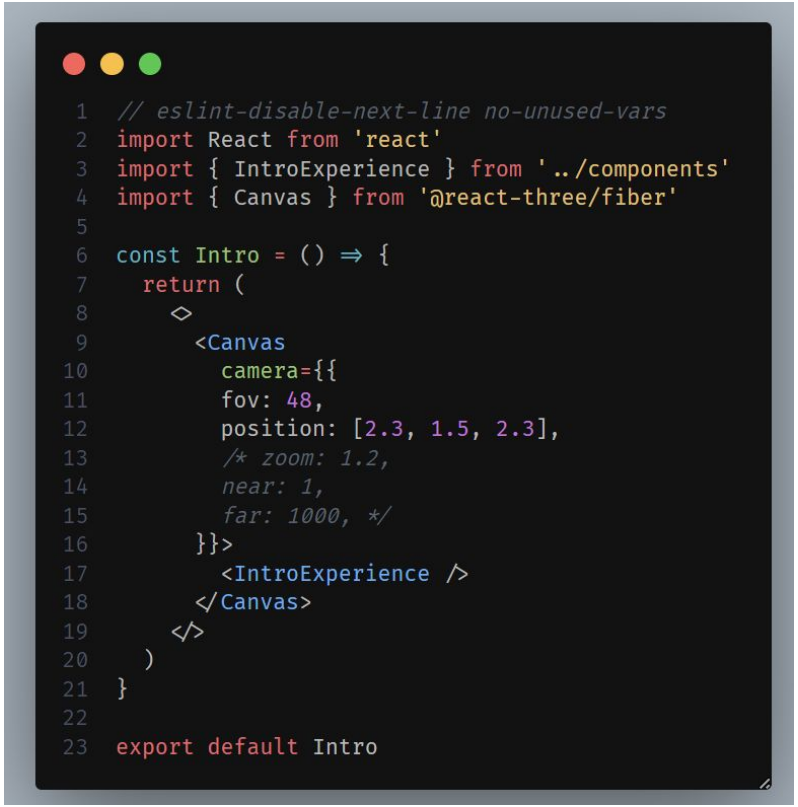
Intro Scene


Halo,
TEAM FEDEX
Welcome to Our Final Project
We design custom, personal landing pages with a creative twist!


MEET OUR TEAM
Mohammed Fachry Dwi Handoko
5025201159
Farraihan Rafi Adityawarman
5025211074
Pascal Roger Junior Tauran
5025211072


SEE OUR PAGE
Start your journey here and now!

Intro.jsx



```jsx
// eslint-disable-next-line no-unused-vars
import React from 'react'
import { IntroExperience } from '../components'
import { Canvas } from '@react-three/fiber'

const Intro = () => {
  return (
    <>
      <Canvas
        camera={{
        fov: 48,
        position: [2.3, 1.5, 2.3],
        /* zoom: 1.2,
        near: 1,
        far: 1000, */
      }}>
        <IntroExperience />
      </Canvas>
    </>
  )
}

export default Intro
```

Camera is to set the position of the camera itself

IntroExperience.jsx

```jsx
// eslint-disable-next-line no-unused-vars
import React from 'react'
/* eslint-disable react/no-unknown-property */
import { OrbitControls, ScrollControls } from '@react-three/drei'
import { Office } from './canvas/Office'
import { IntroOverlay } from './IntroOverlay'
// import { SaulGoodman } from './SaulGoodman'

const IntroExperience = () => {
  return (
    <>
        <ambientLight intensity={1} />
        <OrbitControls enableZoom={false} />
        <ScrollControls pages={3} damping={0.25}>
          <IntroOverlay />
          <Office />
        </ScrollControls>
    </>
  )
}

export default IntroExperience
```

To accompany the intro page itself we made the controls and lighting in this jsx file

“<Office />” is used to import the 3D model itself to the jsx file

IntroOverlay.jsx

```jsx
import React from 'react'
import { Scroll, useScroll } from "@react-three/drei"
import { useFrame } from "@react-three/fiber"
import { useState } from "react"

const Section = (props) => {
  return (
    <section
      className={`h-screen flex flex-col justify-center p-10 ${
        props.right ? "items-end" : "items-start"
      }`}
      style={{
        opacity: props.opacity,
      }}
    >
      <div className="w-1/2 flex items-center justify-center">
        <div className="max-w-sm w-full">
          <div className="bg-transparent rounded-lg px-8 py-12">
            {props.children}
          </div>
        </div>
      </div>
    </section>
  )
}
```

A section of IntroOverlay.jsx consisting of messages

```jsx
export const IntroOverlay = () => {
  const scroll = useScroll();
  const [opacityFirstSection, setOpacityFirstSection] = useState(1);
  const [opacitySecondSection, setOpacitySecondSection] = useState(1);
  const [opacityLastSection, setOpacityLastSection] = useState(1);

  useFrame(() => {
    setOpacityFirstSection(1 - scroll.range(0, 1 / 3));
    setOpacitySecondSection(scroll.curve(1 / 3, 1 / 3));
    setOpacityLastSection(scroll.range(2 / 3, 1 / 3));
  });

  return (
    <Scroll html>
      <div className="w-screen text-gray-800">
        <Section opacity={opacityFirstSection}>
          <h1 className="font-semibold font-['Mars'] pb-5 text-lg">
            <span className="text-4xl font-['Evaa']">
              Halo,
            </span>
            <br />
            Team FEDEX
          </h1>
          <p className="text-gray-500"> Welcome to Our Final Project </p>
          <p className="mt-3 font-light"> We design custom, personal landing pages with a creative twist! </p>
          <a href="#members">
            <p className="animate-bounce mt-6 text-4xl font-bold"> ↑ </p>
          </a>
        </Section>
```

Office.jsx

```jsx
// eslint-disable-next-line no-unused-vars
import React, { useLayoutEffect, useRef } from "react"
import { useGLTF, useScroll } from "@react-three/drei"
import { useFrame } from "@react-three/fiber"
import gsap from "gsap"

export const FLOOR_HEIGHT = 2.3;
export const NB_FLOORS = 3;

export function Office(props) {
  const { nodes, materials } = useGLTF("./models/WawaOffice.glb");
  const ref = useRef();
  const tl = useRef();
  const libraryRef = useRef();
  const atticRef = useRef();

  const scroll = useScroll();

  useFrame(() => {
    tl.current.seek(scroll.offset * tl.current.duration());
  });

  useLayoutEffect(() => {
    tl.current = gsap.timeline();
```

```jsx
// Office Rotation
    tl.current.to(
      ref.current.rotation,
      { duration: 1, x: 0, y: Math.PI / 6, z: 0 },
      0
    );
    tl.current.to(
      ref.current.rotation,
      { duration: 1, x: 0, y: -Math.PI / 6, z: 0 },
      1
    );

    // Office movement
    tl.current.to(
      ref.current.position,
      {
        duration: 1,
        x: -1,
        z: 2,
      },
      0
    );
    tl.current.to(
      ref.current.position,
      {
        duration: 1,
        x: 1,
        z: 2,
      },
      1
    );
```

We modify each movement of the office in the Office.jsx.
That consist of the rotation, movement, etc (using GSAP)

Library.jsx and Attic.jsx

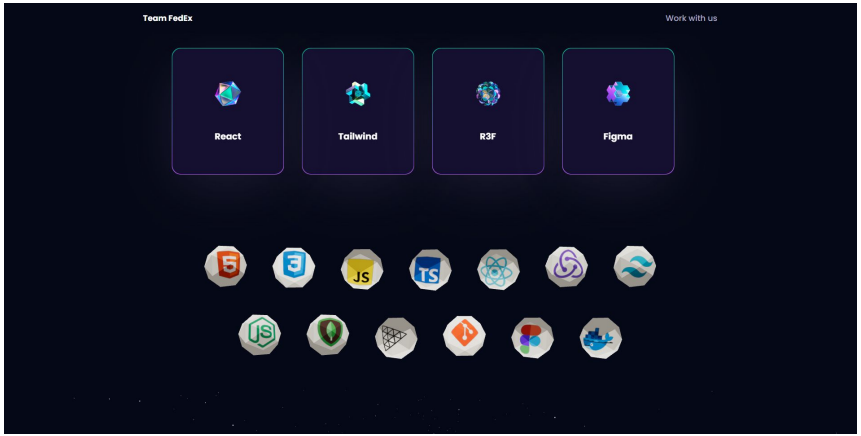
```
// LIBRARY FLOOR
    tl.current.from(
      libraryRef.current.position,
      {
        duration: 0.5,
        x: -2,
      },
      0.5
    );
    tl.current.from(
      libraryRef.current.rotation,
      {
        duration: 0.5,
        y: -Math.PI / 2,
      },
      0
    );
```

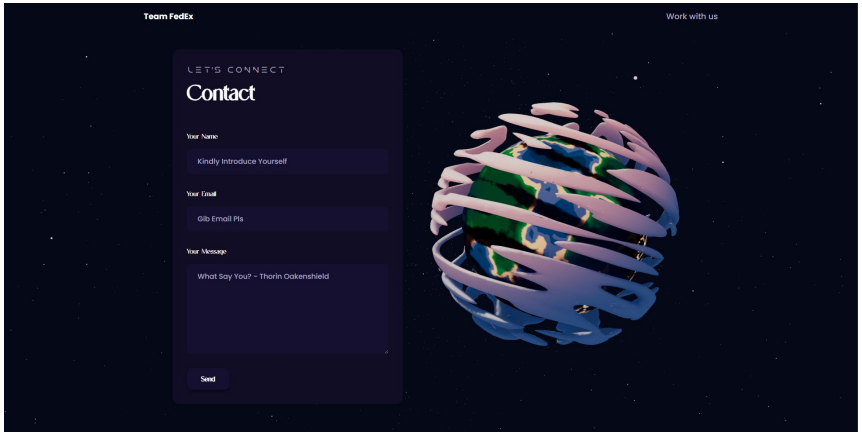
```
// ATTIC
    tl.current.from(
      atticRef.current.position,
      {
        duration: 1.5,
        y: 2,
      },
      0
    );

    tl.current.from(
      atticRef.current.rotation,
      {
        duration: 0.5,
        y: Math.PI / 2,
      },
      1
    );

    tl.current.from(
      atticRef.current.position,
      {
        duration: 0.5,

        z: -2,
      },
      1.5
    );
  }, []);
```

Position settings for each 3D models in Intro.jsx

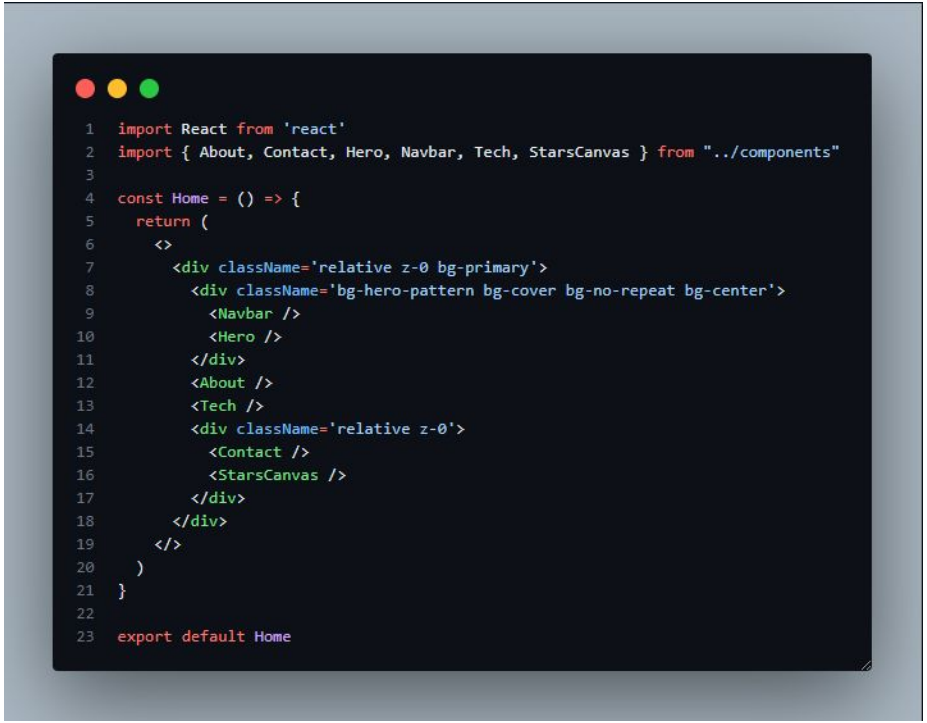
```jsx
  return (
    <group
      {...props}
      dispose={null}
      ref={ref}
      position={[0.5, -1, -1]}
      rotation={[0, -Math.PI / 3, 0]}
    >
      <mesh geometry={nodes["01_office"].geometry} material={materials["01"]} />
      <group position={[0, 2.11, -2.23]}>
        <group ref={libraryRef}>
          <mesh
            geometry={nodes["02_library"].geometry}
            material={materials["02"]}
          />
        </group>
      </group>
      <group position={[-1.97, 4.23, -2.2]}>
        <group ref={atticRef}>
          <mesh
            geometry={nodes["03_attic"].geometry}
            material={materials["03"]}
          />
        </group>
      </group>
    </group>
  );
}

useGLTF.preload("./models/WawaOffice.glb");
```

Home page Scene





Home.jsx



```jsx
import React from 'react'
import { About, Contact, Hero, Navbar, Tech, StarsCanvas } from "../components"

const Home = () => {
  return (
    <>
      <div className='relative z-0 bg-primary'>
        <div className='bg-hero-pattern bg-cover bg-no-repeat bg-center'>
          <Navbar />
          <Hero />
        </div>
        <About />
        <Tech />
        <div className='relative z-0'>
          <Contact />
          <StarsCanvas />
        </div>
      </div>
    </>
  )
}

export default Home
```

This calls the `Navbar`, `Hero`, `About`, `Tech`, `Contact` and `StarsCanvas` elements for the web page.

Hero.jsx

```jsx
import { motion } from "framer-motion"

import { styles } from "../styles"
import { ComputersCanvas } from "./canvas"

const Hero = () => {
  return (
    <section className={`relative w-full h-screen mx-auto`}>
      <div
        className={`absolute inset-0 top-[120px]  max-w-7xl mx-auto ${styles.paddingX} flex flex-row items-start gap-5`}
      >
        <div className='flex flex-col justify-center items-center mt-5'>
          <div className='w-5 h-5 rounded-full bg-[#915EFF]' />
          <div className='w-1 sm:h-80 h-40 violet-gradient' />
        </div>

        <div>
          <h1 className={`${styles.heroHeadText} text-white-100 font-['Mars']`}>
            IF <span className="text-[#915EFF] font-['Mars']"> GRAFKOM </span>
          </h1>
          <p className={`${styles.heroSubText} mt-2 text-gray-200`}>
          Teknik Informatika ITS
          </p>
        </div>
      </div>

      <ComputersCanvas />

      <div className='absolute xs:bottom-10 bottom-32 w-full flex justify-center items-center'>
        <a href='#about'>
          <div className='w-[35px] h-[64px] rounded-3xl border-4 border-secondary flex justify-center items-start p-2'>
            <motion.div
              animate={{
                y: [0, 24, 0],
              }}
              transition={{
                duration: 1.5,
                repeat: Infinity,
                repeatType: "loop",
              }}
              className='w-3 h-3 rounded-full bg-secondary mb-1'
            />
          </div>
        </a>
      </div>
    </section>
  );
};

export default Hero;

```

This calls the `ComputerCanvas` element. There is also another element which contains an anchor tag (<a>) with a href attribute set to '#about', It has a rounded rectangular shape and contains an animated dot element (<motion.div>) using the motion object from the "framer-motion" library. The dot animates vertically with a repeating motion defined by the animate and transition properties

Computers.jsx pt.1

```jsx
import React, { Suspense, useEffect, useState } from "react";
import { Canvas } from "@react-three/fiber";
import { OrbitControls, Preload, useGLTF } from "@react-three/drei";

import CanvasLoader from "../Loader";

const Computers = ({ isMobile }) => {
  const computer = useGLTF("./desktop_pc/scene.gltf");

  return (
    <mesh>
      <hemisphereLight intensity={0.15} groundColor='black' />
      <spotLight
        position={[-20, 50, 10]}
        angle={0.12}
        penumbra={1}
        intensity={1}
        castShadow
        shadow-mapSize={1024}
      />
      <pointLight intensity={1} />
      <primitive
        object={computer.scene}
        scale={isMobile ? 0.7 : 0.75}
        position={isMobile ? [0, -3, -2.2] : [0, -3.25, -1.5]}
        rotation={[-0.01, -0.2, -0.1]}
      />
    </mesh>
  );
};
```

In this computer.jsx, we use the `useGLTF` hook to load the3D model of a desktop located at ./desktop_pc /scene.gltf. The loaded model is assigned to the computer variable. The component returns a <mesh> element, which serves as a container for the 3D scene. Inside the <mesh>, there are several lighting elements defined; <hemisphereLight> which represents a hemisphere light with an intensity of 0.15 and a ground color of black. It provides ambient lighting to the scene. <spotLight> which represents a spotlight positioned at [-20, 50, 10] with specific properties angle, penumbra, intensity, and shadow casting.

It provides directional and focused lighting with shadows and <pointLight> which represents a point light with an intensity of 1. It provides additional lighting to the scene. The 3D model of the computer is rendered as a <primitive> component. The object prop is set to computer.scene, which is the loaded 3D model from the `useGLTF` hook.

Computers.jsx pt.2

```jsx

const ComputersCanvas = () => {
  const [isMobile, setIsMobile] = useState(false);

  useEffect(() => {
    // Add a listener for changes to the screen size
    const mediaQuery = window.matchMedia("(max-width: 500px)");

    // Set the initial value of the `isMobile` state variable
    setIsMobile(mediaQuery.matches);

    // Define a callback function to handle changes to the media query
    const handleMediaQueryChange = (event) => {
      setIsMobile(event.matches);
    };

    // Add the callback function as a listener for changes to the media query
    mediaQuery.addEventListener("change", handleMediaQueryChange);

    // Remove the listener when the component is unmounted
    return () => {
      mediaQuery.removeEventListener("change", handleMediaQueryChange);
    };
  }, []);

  return (
    <Canvas
      frameloop='demand'
      shadows
      dpr={[1, 2]}
      camera={{ position: [20, 3, 5], fov: 25 }}
      gl={{ preserveDrawingBuffer: true }}
    >
      <Suspense fallback={<CanvasLoader />}>
        <OrbitControls
          enableZoom={false}
          maxPolarAngle={Math.PI / 2}
          minPolarAngle={Math.PI / 2}
        />
        <Computers isMobile={isMobile} />
      </Suspense>

      <Preload all />
    </Canvas>
  );
};

export default ComputersCanvas;

```

The <Canvas> element is configured with several props. `frameloop` is set to 'demand', which means the rendering loop will only occur when requested. `camera` configures the camera position and field of view. `gl` configures the WebGL renderer, with preserveDrawingBuffer set to true to allow capturing the canvas as an image.

Inside the <Canvas>, there is a <Suspense> component that provides a fallback UI (the <CanvasLoader /> component) while the content is loading.

The <Computers> component is rendered, passing the isMobile prop to control the scaling and positioning of the computer model. Finally, the <Preload> component from @react-three/drei is used to ensure all assets are preloaded before rendering the scene.

Tech.jsx

```jsx
import React from "react";

import { BallCanvas } from "./canvas";
import { SectionWrapper } from "../hoc";
import { technologies } from "../constants";

const Tech = () => {
  return (
    <div className='flex flex-row flex-wrap justify-center gap-10'>
      {technologies.map((technology) => (
        <div className='w-28 h-28' key={technology.name}>
          <BallCanvas icon={technology.icon} />
        </div>
      ))}
    </div>
  );
};

export default SectionWrapper(Tech, "");
```

The Tech component returns a that sets up a flex container that displays its children in a row, wraps them to the next line when necessary, centers them horizontally, and applies a gap of 10 units between them.

Inside , there is a mapping over the technologies array using the map method. For each technology object, a element with the class name 'w-28 h-28' is rendered. This sets the width and height of each container element to 28 units.

Inside each container element, the BallCanvas component is rendered. The icon prop is passed to the `BallCanvas` component, which represents the icon or image associated with the technology in the `Ball`(s).

Ball.jsx pt.1

```jsx
import React, { Suspense } from "react";
import { Canvas } from "@react-three/fiber";
import {
  Decal,
  Float,
  OrbitControls,
  Preload,
  useTexture,
} from "@react-three/drei";

import CanvasLoader from "../Loader";

const Ball = (props) => {
  const [decal] = useTexture([props.imgUrl]);

  return (
    <Float speed={1.75} rotationIntensity={1} floatIntensity={2}>
      <ambientLight intensity={0.25} />
      <directionalLight position={[0, 0, 0.05]} />
      <mesh castShadow receiveShadow scale={2.75}>
        <icosahedronGeometry args={[1, 1]} />
        <meshStandardMaterial
          color='#fff8eb'
          polygonOffset
          polygonOffsetFactor={-5}
          flatShading
        />
```

The Ball component represents a 3D ball with an image `decal` applied to its surface. It receives the imgUrl prop, which is the URL of the image.

Inside the Ball component, a <Float> component from the @react-three/drei library is used to add floating animation to the ball. It receives `speed`, `rotationIntensity`, and `floatIntensity` props to control the animation behavior.

Several lighting elements are defined within the <Float> component: <ambientLight> which provides ambient lighting with an intensity of 0.25 and <directionalLight> which represents a directional light positioned at [0, 0, 0.05].

A <mesh> component is used to render the ball geometry. It has a scale of 2.75 and casts and receives shadows. The ball geometry is created using an <icosahedronGeometry> component with radius 1 and detail 1.

The ball is rendered with a <meshStandardMaterial> component and applies polygon offset, and uses flat shading.

Ball.jsx pt.2

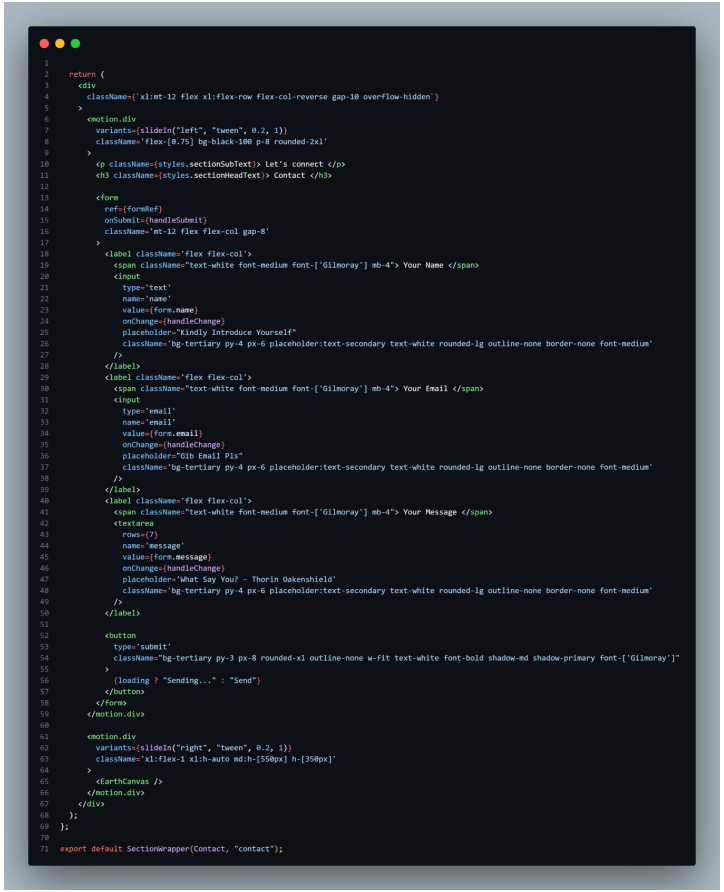
```
28          <Decal
29            position={[0, 0, 1]}
30            rotation={[2 * Math.PI, 0, 6.25]}
31            scale={1}
32            map={decal}
33            flatShading
34          />
35        </mesh>
36      </Float>
37    );
38  };
39
40  const BallCanvas = ({ icon }) => {
41    return (
42      <Canvas
43        frameloop='demand'
44        dpr={[1, 2]}
45        gl={{ preserveDrawingBuffer: true }}
46      >
47        <Suspense fallback={<CanvasLoader />}>
48          <OrbitControls enableZoom={false} />
49          <Ball imgUrl={icon} />
50        </Suspense>
51
52        <Preload all />
53      </Canvas>
54    );
55  };
56
57  export default BallCanvas;
58
```

A <Decal> component is added as a child of the <mesh>. It represents the decal applied to the ball's surface. It receives props like position, rotation, scale, map of the decal texture, and `flatShading`. The BallCanvas component represents the canvas for rendering the 3D ball with the decal. It receives the icon prop, which likely represents the URL of the decal image.

Inside the BallCanvas component, a <Canvas> component from the @react-three/fiber library is used to create the WebGL canvas for rendering the 3D scene.The <Canvas> component is configured with props like frameloop set to 'demand', dpr set to [1, 2] for device pixel ratio, and gl with preserveDrawingBuffer set to true to enable capturing the canvas as an image. The <OrbitControls> component from @react-three/drei is added to enable the user to orbit around the scene.

The Ball component is rendered within the <Suspense>, passing the imgUrl prop to specify the decal image. And the <Preload> component from @react-three/drei is used to ensure all assets are preloaded before rendering the scene.

Contact.jsx



```jsx
return (
  <div
    className={`xl:mt-12 flex xl:flex-row flex-col-reverse gap-10 overflow-hidden`}
  >
    <motion.div
      variants={slideIn("left", "tween", 0.2, 1)}
      className='flex-[0.75] bg-black-100 p-8 rounded-2xl'
    >
      <p className={styles.sectionSubText}> Let's connect </p>
      <h3 className={styles.sectionHeadText}> Contact </h3>

      <form
        ref={formRef}
        onSubmit={handleSubmit}
        className='mt-12 flex flex-col gap-8'
      >
        <label className='flex flex-col'>
          <span className="text-white font-medium font-['Gilmoray'] mb-4"> Your Name </span>
          <input
            type='text'
            name='name'
            value={form.name}
            onChange={handleChange}
            placeholder="Kindly Introduce Yourself"
            className='bg-tertiary py-4 px-6 placeholder:text-secondary text-white rounded-lg outline-none border-none font-medium'
          />
        </label>
        <label className='flex flex-col'>
          <span className="text-white font-medium font-['Gilmoray'] mb-4"> Your Email </span>
          <input
            type='email'
            name='email'
            value={form.email}
            onChange={handleChange}
            placeholder="Gib Email Pls"
            className='bg-tertiary py-4 px-6 placeholder:text-secondary text-white rounded-lg outline-none border-none font-medium'
          />
        </label>
        <label className='flex flex-col'>
          <span className="text-white font-medium font-['Gilmoray'] mb-4"> Your Message </span>
          <textarea
            rows={7}
            name='message'
            value={form.message}
            onChange={handleChange}
            placeholder='What Say You? - Thorin Oakenshield'
            className='bg-tertiary py-4 px-6 placeholder:text-secondary text-white rounded-lg outline-none border-none font-medium'
          />
        </label>

        <button
          type='submit'
          className="bg-tertiary py-3 px-8 rounded-xl outline-none w-fit text-white font-bold shadow-md shadow-primary font-['Gilmoray']"
        >
          {loading ? "Sending..." : "Send"}
        </button>
      </form>
    </motion.div>

    <motion.div
      variants={slideIn("right", "tween", 0.2, 1)}
      className='xl:flex-1 xl:h-auto md:h-[550px] h-[350px]'
    >
      <EarthCanvas />
    </motion.div>
  </div>
);
};

export default SectionWrapper(Contact, "contact");
```

The `Contact` component combines a form for user input and an interactive visual element to create a contact section in a larger application. It calls <EarthCanvas /> which renders a rotating globe 3D element.

Earth.jsx

```jsx
import React, { Suspense } from "react";
import { Canvas } from "@react-three/fiber";
import { OrbitControls, Preload, useGLTF } from "@react-three/drei";

import CanvasLoader from "../Loader";

const Earth = () => {
  const earth = useGLTF("./planet/scene.gltf");

  return (
    <primitive object={earth.scene} scale={2.5} position-y={0} rotation-y={0} />
  );
};

const EarthCanvas = () => {
  return (
    <Canvas
      shadows
      frameloop='demand'
      dpr={[1, 2]}
      gl={{ preserveDrawingBuffer: true }}
      camera={{
        fov: 45,
        near: 0.1,
        far: 200,
        position: [-4, 3, 6],
      }}
    >
      <Suspense fallback={<CanvasLoader />}>
        <OrbitControls
          autoRotate
          enableZoom={false}
          maxPolarAngle={Math.PI / 2}
          minPolarAngle={Math.PI / 2}
        />
        <Earth />

        <Preload all />
      </Suspense>
    </Canvas>
  );
};

export default EarthCanvas;
```

The `Earth` component uses the `useGLTF` hook from @react-three/drei to load a 3D model of the Earth from a GLTF file. The loaded Earth model is rendered as a primitive component with the object prop set to the scene property of the loaded model.

The `EarthCanvas` component is defined as a functional component. It renders a Canvas component from react-three-fiber, which provides the 3D rendering context. The Canvas component is configured with props:`shadows` enabling shadows in the scene. `frameloop='demand'` which specifies that the rendering loop should be triggered manually. `gl={{ preserveDrawingBuffer: true }}` enabling preservation of the drawing buffer for capturing the canvas content. `camera`: Configures the camera used in the scene, including the field of view, near and far clipping planes, and initial position.

The OrbitControls component from @react-three/drei is added to enable orbit controls for the camera. It allows the user to rotate the camera around the scene. The Earth component is rendered inside the Suspense component, which displays the loaded Earth model when it is ready. The Preload component from @react-three/drei is used to preload all assets, ensuring that the 3D model and any associated resources are fully loaded before rendering.