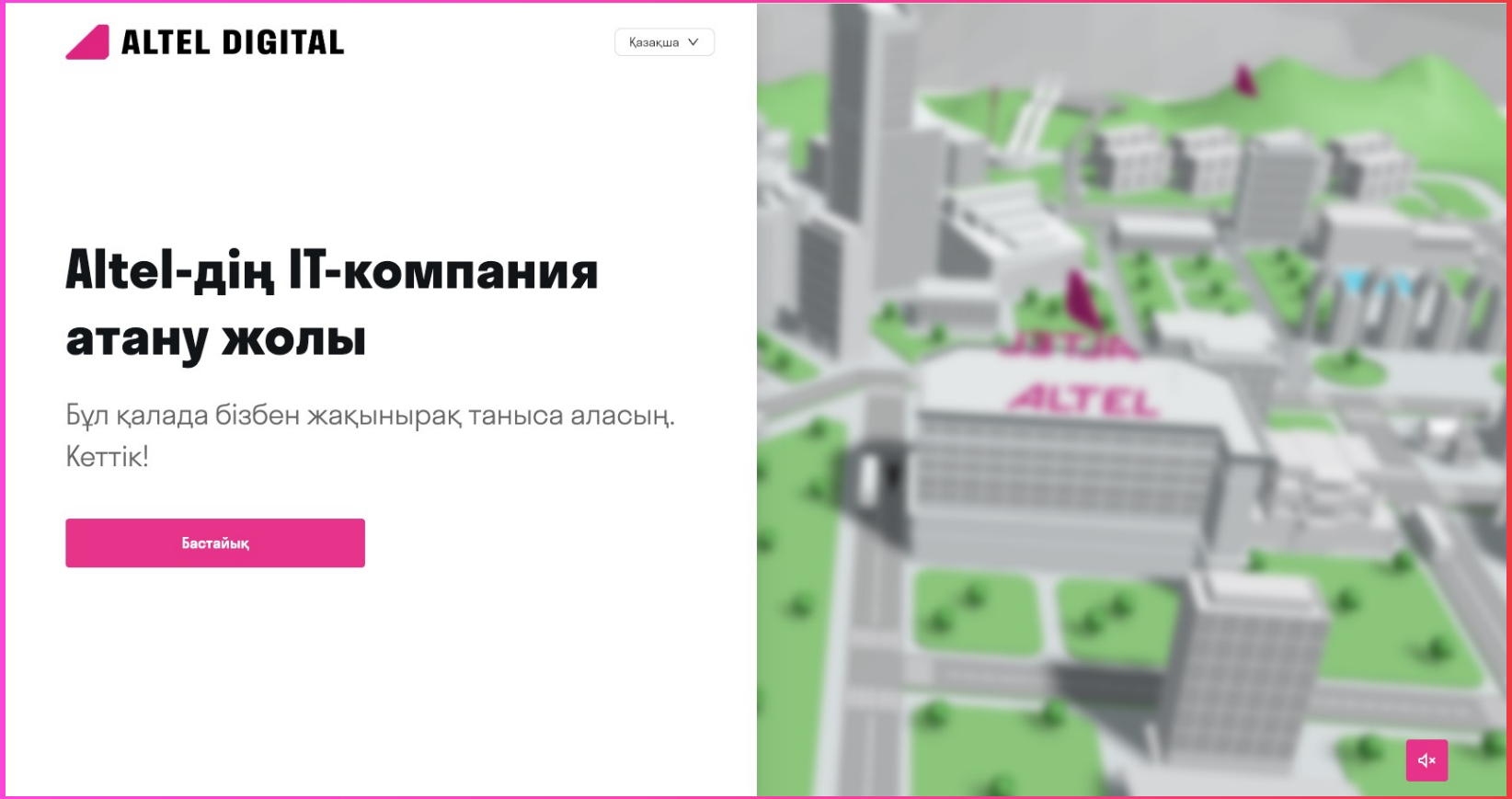
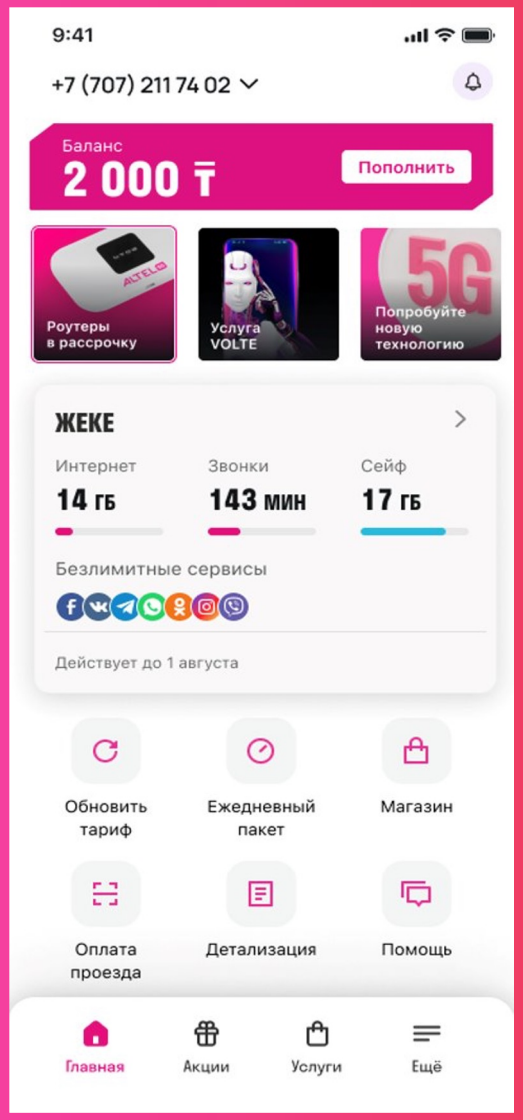
Раз, два, три джс!
Как я делал свой первый 3D сайт?

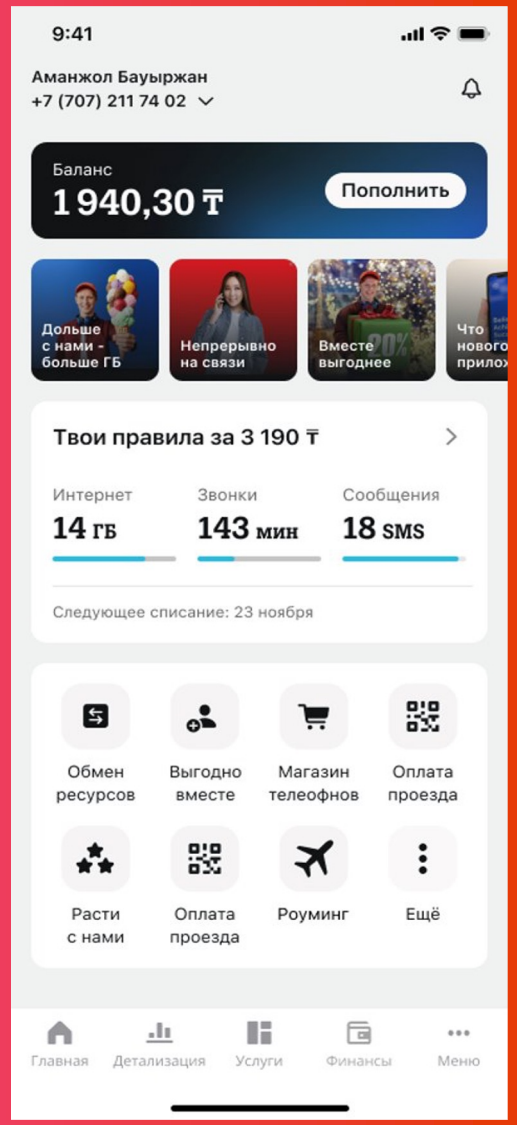


```
const speaker = {
        name: “Bekzhan”,
        surname: “Borbekov”,
        speciality: “frontend dev”,
        experience: “over 5 years”,
        hobby: [
                “audiobooks”,
                “react native”,
                “edm”,
        ]
}
```

Tele2 Kazakhstan / Altel Digital

- 2 основных мобильных приложения
- 10+ веб-приложения
- 50+ сервисов
- 7 млн. пользователей (36% населения Казахстана)





План доклада

- Как формировались требования и происходил анализ
- Как я обучался новой для себя технологии
- Почему я решил попробовать обернуть threeJS в React
- Как мы прорабатывали 3D-модель города Алматы
- Как я оптимизировал получившийся продукт
- Что проект ждет в будущем
- Какие выводы я сделал

Как формировались требования и происходил анализ

- Создать интерактивный 3D-сайт
- Сформировать правильное восприятие трансформации из телекома в IT
- Демонстрация уровня профессионализма команды разработки

Как я обучался новой для себя технологии

- Документация threejs.org
- Курс Three.js Journey

Как я обучался новой для себя технологии

- Poimandres documentations
- Blender reference manual
- YouTube

Почему я решил попробовать обернуть threeJS в React

Минусы нативного three.js:

- Сложность управления состоянием
- Более высокий порог входа
- Все надо самому контролировать

Почему я решил попробовать обернуть threeJS в React

Плюсы React + three.js:

- Удобство и гибкость
- Компонентный подход
- Интеграция с современным веб-стеком

Почему я решил попробовать обернуть threeJS в React

Плюсы React + three.js:

- Удобство и гибкость
- Компонентный подход
- Интеграция с современным веб-стеком
- **@react-three/fiber и @react-three/drei**

Native three.js vs @react-three/fiber

```
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 )
```

VS

```
<mesh>
    <boxGeometry />
    <meshBasicMaterial color="red" />
</mesh>
```

Как мы прорабатывали 3D-модель города Алматы

Как мы прорабатывали 3D-модель города Алматы

1. Создать 3D модели

Как мы прорабатывали 3D-модель города Алматы

1. Создать 3D модели
2. Отрисовать 3D модели в браузере

Как мы прорабатывали 3D-модель города Алматы

1. Создать 3D модели
2. Отрисовать 3D модели в браузере
3. Добавить реалистичности

Как мы прорабатывали 3D-модель города Алматы

1. Создать 3D модели
2. Отрисовать 3D модели в браузере
3. Добавить реалистичности
4. Добавить интерактивности

Как мы прорабатывали 3D-модель города Алматы

1. Создать 3D модели
2. Отрисовать 3D модели в браузере
3. Добавить реалистичности
4. Добавить интерактивности
5. Анимировать транспорт

Как мы прорабатывали 3D-модель города Алматы

1. Создать 3D модели
2. Отрисовать 3D модели в браузере
3. Добавить реалистичности
4. Добавить интерактивности
5. Анимировать транспорт
6. Реализовать клик по зданию

Создаем 3D модели

- OBJ
- FBX
- STL
- COLLADA
- GLTF

Создаем 3D модели

- OBJ

baseMap.obj
Plain Text Document - 1,8 MB

- FBX
- STL
- COLLADA
- GLTF

Создаем 3D модели

- OBJ

baseMap.obj
Plain Text Document - 1,8 MB

- FBX

baseMap.fbx
Document - 644 KB

- STL
- COLLADA
- GLTF

Создаем 3D модели

- OBJ

baseMap.obj
Plain Text Document - 1,8 MB

- FBX

baseMap.fbx
Document - 644 KB

- STL

baseMap.stl
Standard Tessellated Geometry File Format - 882 KB

- COLLADA
- GLTF

Создаем 3D модели

- OBJ

baseMap.obj
Plain Text Document - 1,8 MB

- FBX

baseMap.fbx
Document - 644 KB

- STL

baseMap.stl
Standard Tessellated Geometry File Format - 882 KB

- COLLADA

baseMap.dae
Plain Text Document - 2,7 MB

- GLTF

Создаем 3D модели

- OBJ

  baseMap.obj
  Plain Text Document - 1,8 MB

- FBX

  baseMap.fbx
  Document - 644 KB

- STL

  baseMap.stl
  Standard Tessellated Geometry File Format - 882 KB

- COLLADA

  baseMap.dae
  Plain Text Document - 2,7 MB

- GLTF

  baseMap.glb
  Document - 834 KB

  baseMap.glb
  Document - 235 KB

GL Transmission Format

GL Transmission Format

- Полная сцена в одном файле

GL Transmission Format

- Полная сцена в одном файле
- JSON-структура

GL Transmission Format

- Полная сцена в одном файле
- JSON-структура
- Поддержка формата

GL Transmission Format

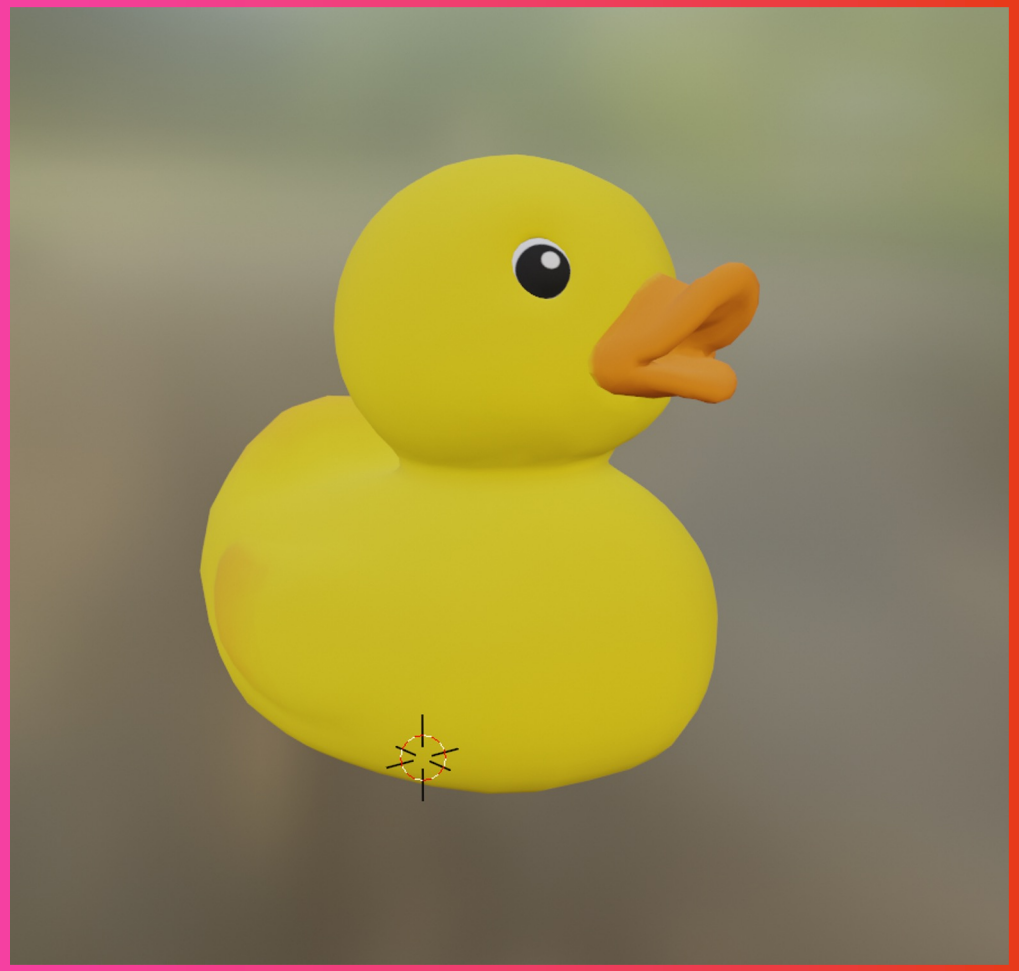
- Полная сцена в одном файле
- JSON-структура
- Поддержка формата
- Был разработан для веба и для мобильных приложений

GL Transmission Format

- Полная сцена в одном файле
- JSON-структура
- Поддержка формата
- Был разработан для веба и для мобильных приложений
- Имеет несколько форматов хранения

Форматы GLTF

- gltf
- gltf – Binary
- gltf – Embedded
- gltf – Draco



gltf

- Duck.gltf - представляет собой JSON, содержит различную информацию, такую как камеры, освещение, сцена, материалы и т.д.
- Duck0.bin - двоичный файл, содержит данные геометрии и всю информацию связанную с вершинами
- DuckCM.png - это просто текстура утки

Duck.gltf	5 KB
Duck0.bin	102 KB
DuckCM.png	16 KB

gltf - Binary

- Этот формат состоит только из одного двоичного файла
- Содержит все данные, о которых мы говорили в формате glTF по умолчанию
- Не можем изменить его данные

Duck.glb	120 KB

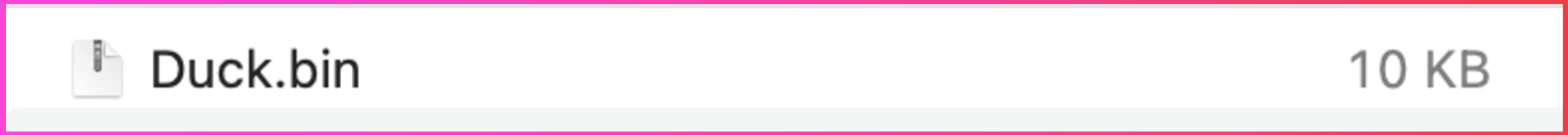
gltf - Embedded

- Этот формат состоит только из одного файла
- Файл представляет собой JSON, который можно открыть в редакторе
- Можем изменить его данные

Duck.gltf	163 KB

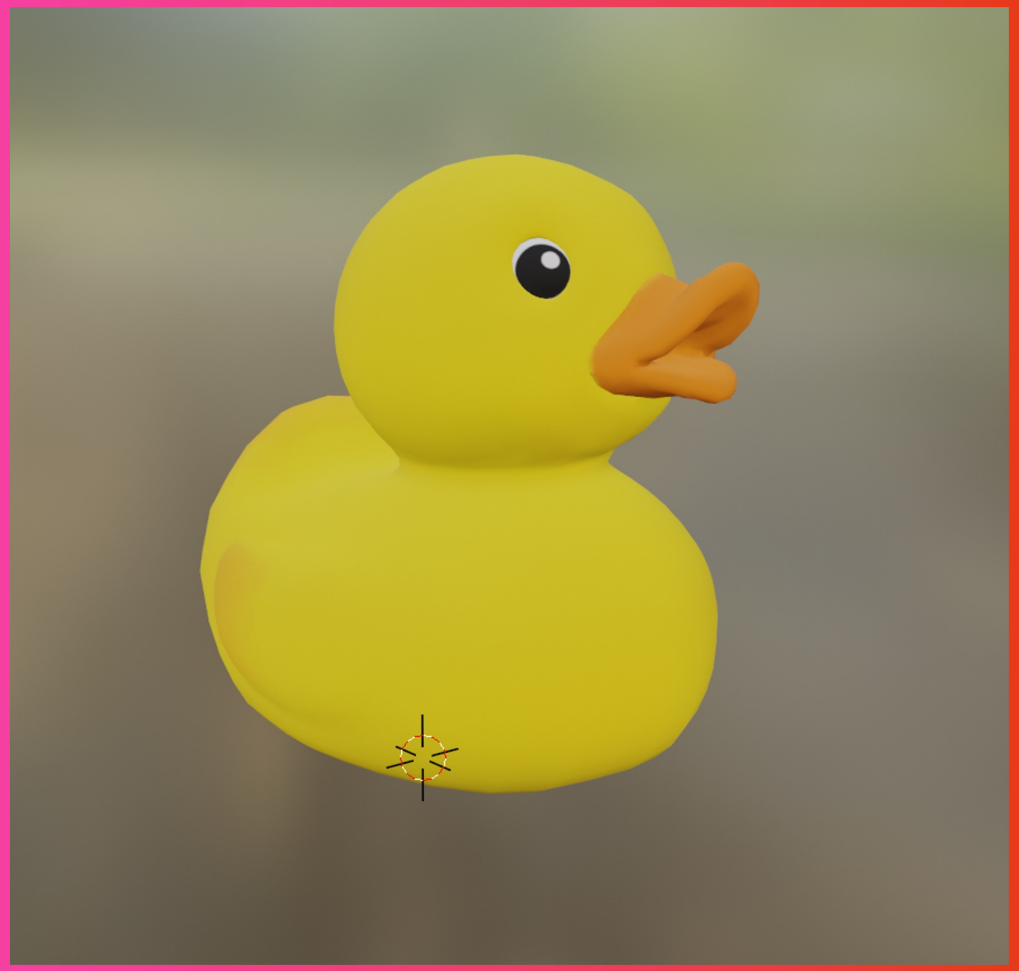
gltf - Draco

- Этот формат состоит только из одного файла
- Геометрия модели сжимается с использованием алгоритма Draco
- Нужен специальный лоадер и декодер



Форматы GLTF

- gltf (123kb)
- gltf – Binary (120kb)
- gltf – Embedded (163kb)
- gltf – Draco (10kb)

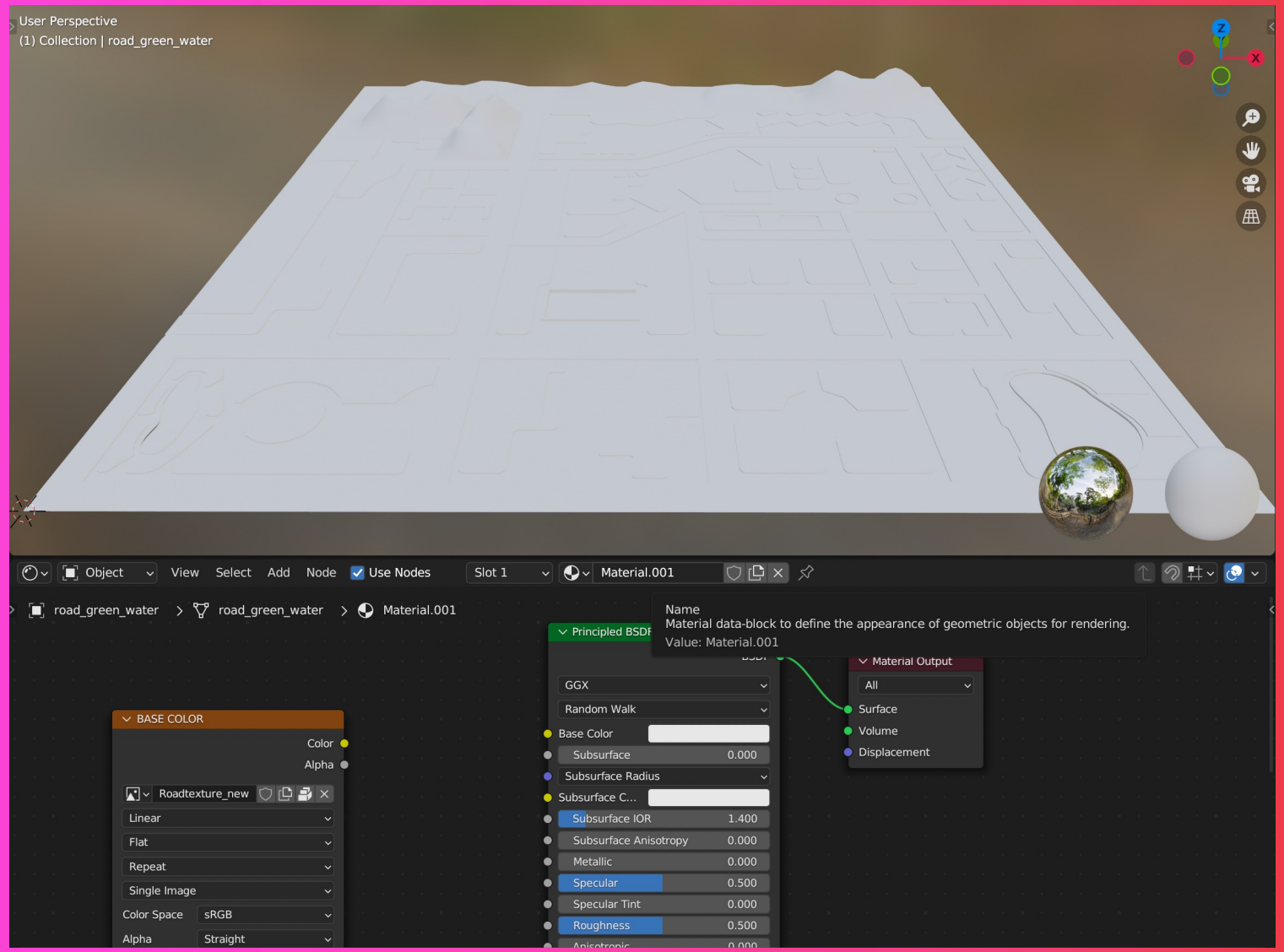


Про текстуры

- Изображение, которое покрывает поверхность нашей геометрии
- Есть разные типы текстур, которые по-разному влияют на внешний вид нашей геометрии

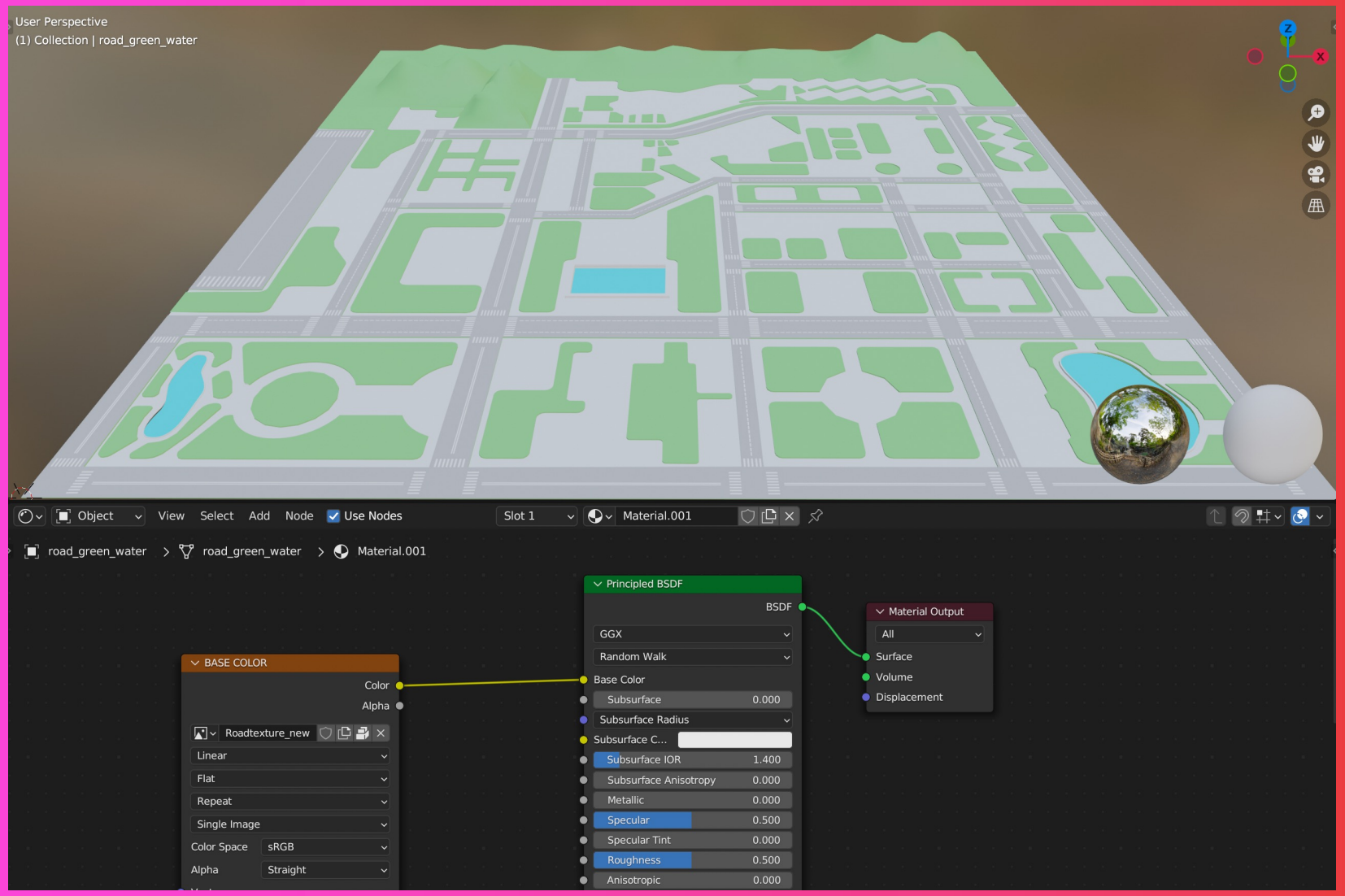
```
const textureLoader = new THREE.TextureLoader()

const colorTexture = textureLoader.load('/textures/color-texture.jpg')
```

Модель в блендере



Текстура плоскости

Модель в блендере с текстурой



Почему такой подход

Почему такой подход

- Загружается одним файлом

Почему такой подход

- Загружается одним файлом
- Снижает нагрузку на рендеринг в реальном времени

Почему такой подход

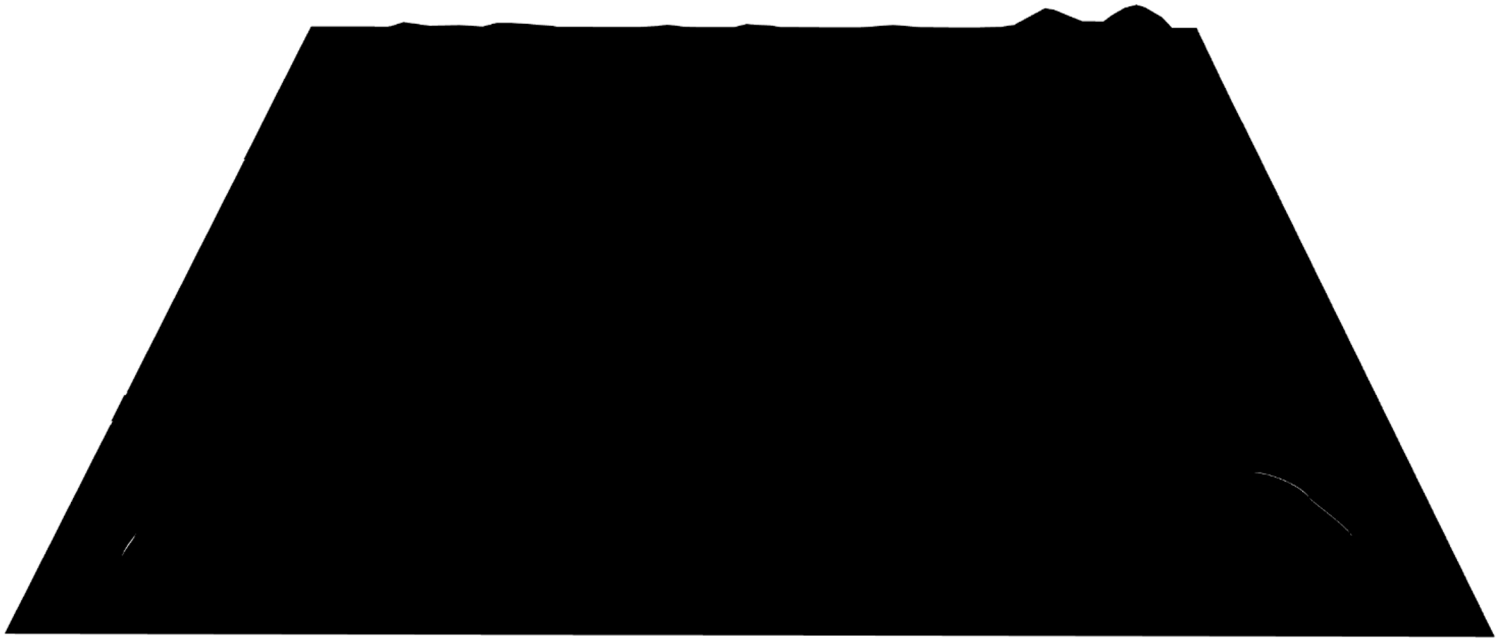
- Загружается одним файлом
- Снижает нагрузку на рендеринг в реальном времени
- Позволяет предварительно вычислить эффекты освещения, теней и другие визуальные детали

Отрисовка модели в браузере

- Загружаем 3D модель

```
import { useGLTF } from '@react-three/drei'

const BaseMap = () => {

    const baseMap = useGLTF('./models/baseMap/base-map.glb')

    return <primitive object={baseMap.scene} />
}

export default BaseMap
```

Отрисовка модели в браузере



Про материалы

- В Three.js, материал определяет внешний вид и поверхностные характеристики объектов
- Материалы влияют на то, как объект отражает свет, его цвет, прозрачность, блесткость и другие визуальные аспекты

Про материалы

MeshBasicMaterial	MeshStandardMaterial
Простейший материал, не реагирующий на освещение. Идеален для объектов, которые должны отображаться равномерно освещёнными, независимо от сцены.	Более реалистичный материал, поддерживающий физически корректное отображение (Physically Based Rendering, PBR). Он обеспечивает более реалистичное взаимодействие с освещением и тенями.

Свет

Свет определяет, как объекты освещаются и отображаются, включая то, как они отражают или поглощают свет, как формируются тени и как в целом воспринимается сцена.

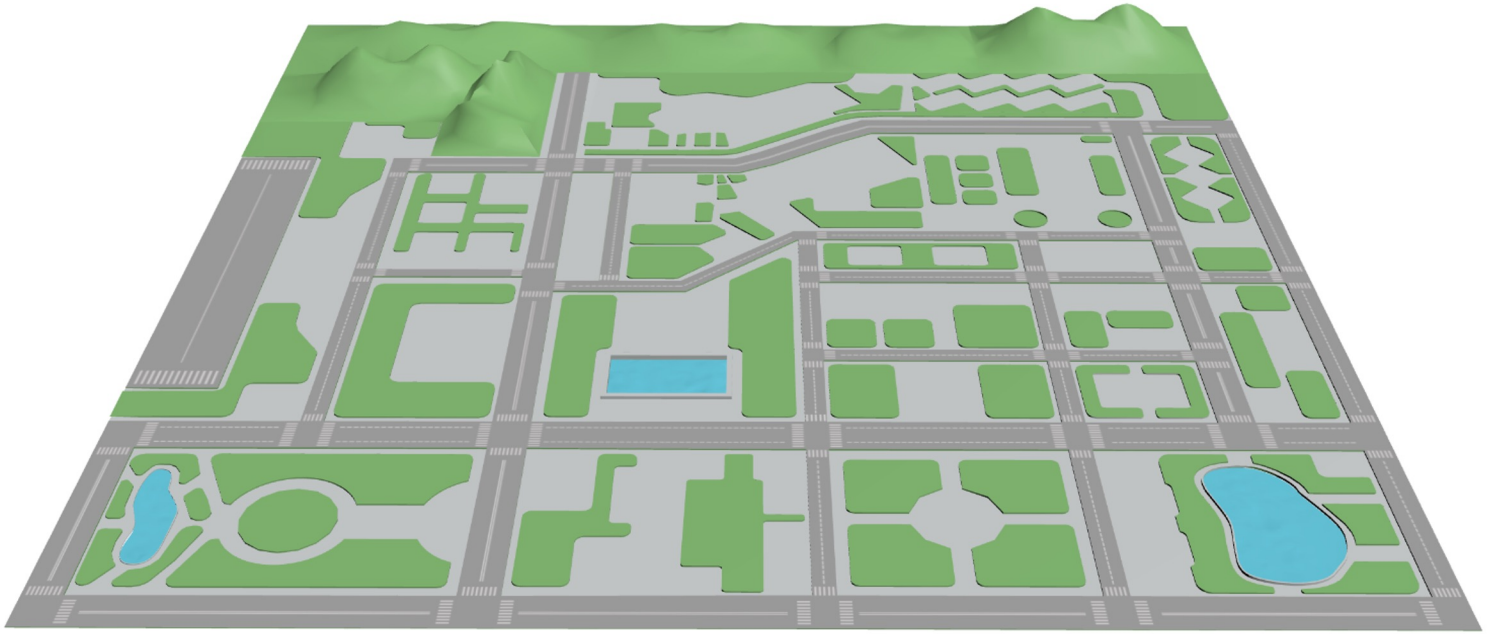
- Directional Light
- Ambient Light

Directional Light (список по одному элементу)

- Имитирует солнечный свет, освещая все объекты одинаково с заданного направления
- Создает тени
- Обычно используется для имитации дневного света
- Исходящие от него лучи параллельны

```
const directionalLight = new THREE.DirectionalLight(0xffffff, 1)
scene.add(directionalLight)

directionalLight.position.set(5, 10, 5)
```

Directional Light

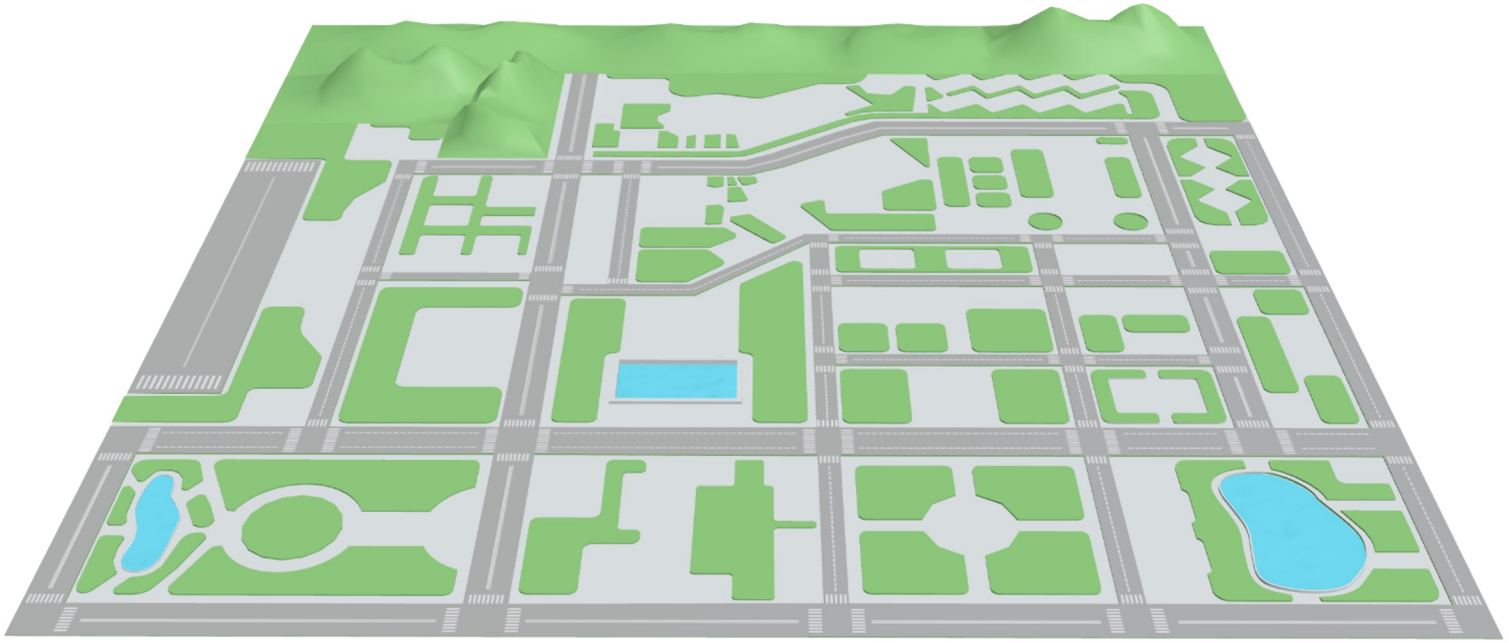


Ambient Light

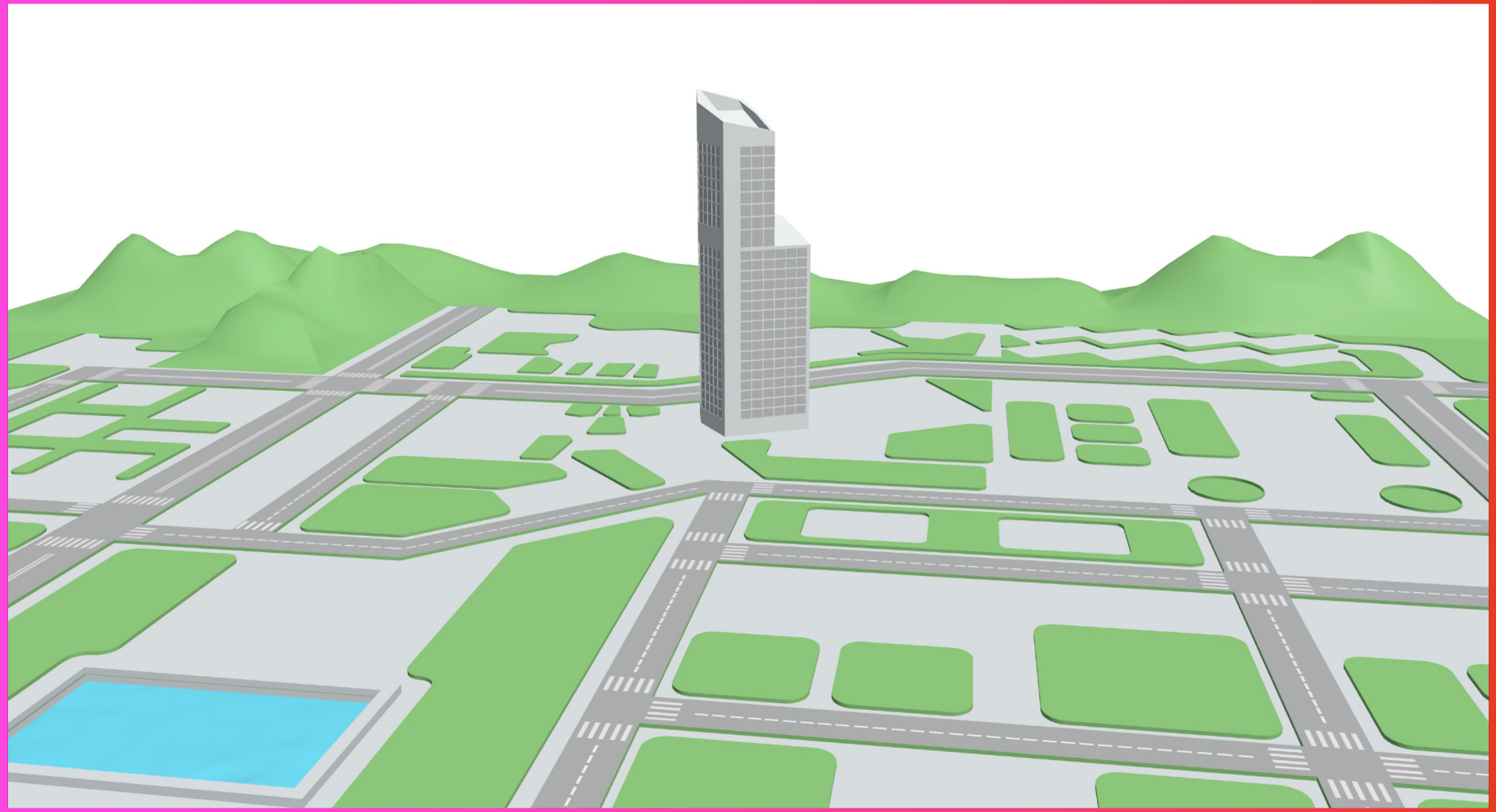
- Равномерное освещение всех объектов со всех сторон
- Не создает тени
- Не имеет направления

```
const ambientLight = new THREE.AmbientLight(0xffffff, 0.5)
scene.add(ambientLight)

const coldAmbientLight = new THREE.AmbientLight(0xadd8e6, 0.2)
scene.add(coldAmbientLight)
```

Ambient Light



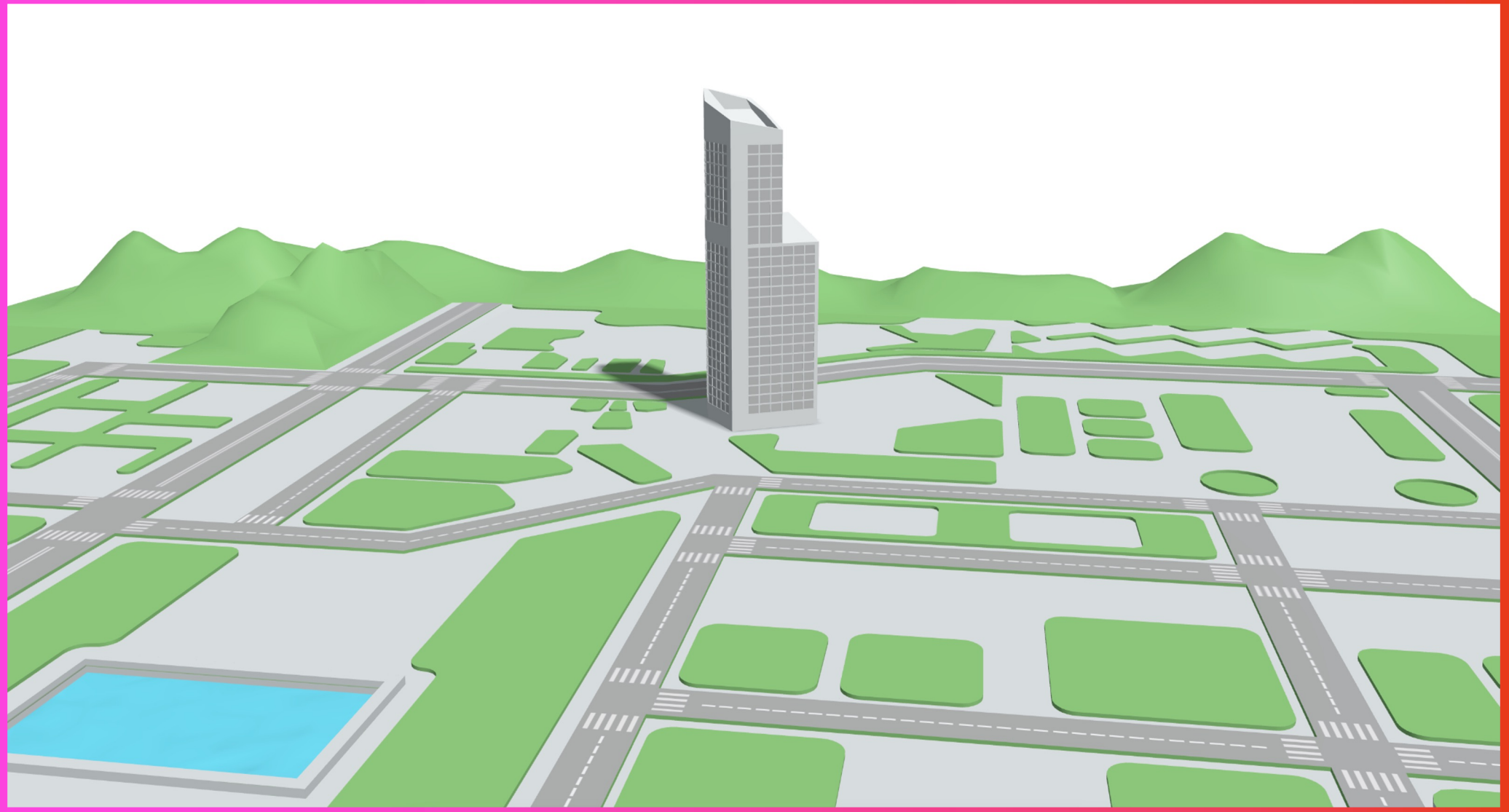
Создаем тени



Создаем тени

Источники света	
Настройка объектов для генерации теней	building.castShadow = true
Настройка объектов для приема теней	baseMap.receiveShadow = true
Настройка источника света для проецирования теней	directionalLight.castShadow = true
Настройка рендерера	renderer.shadowMap.enabled = true

Создаем тени



Рисуем здания и объекты

- Кликабельные здания
- Уникальные здания
- Все остальные объекты

Рендерим кликабельные здания

Рендерим уникальные здания

Рендерим остальные объекты

Рендерим остальные объекты

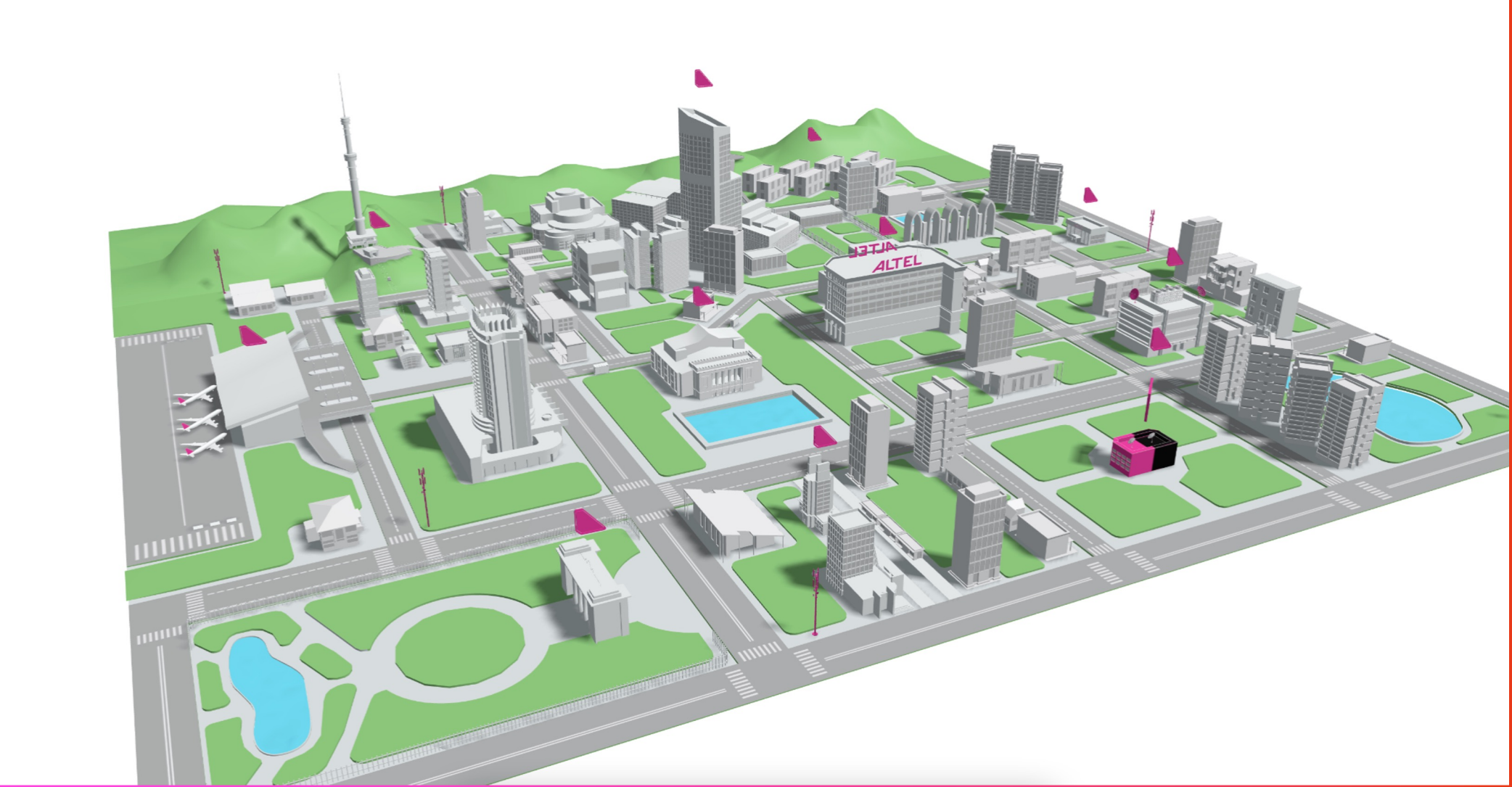
```
{
    "name": "global_4",
    "coords": [
        {
            "position": [9.97, 0, -1.11],
            "rotation": -1.570
        },
        {
            "position": [17.97, 0, -3.45],
            "rotation": 6.283
        },
        {
            "position": [6.51, 0, -8.9],
            "rotation": 1.570
        },
        {
            "position": [7.185, 0, -13.5],
            "rotation": 3.141
        },
        {
            "position": [9.15, 0, -9.6],
            "rotation": 10.131
        }
    ]
},
```

Рендерим остальные объекты

```json
{
    "name": "global_4",
    "coords": [
        {
            "position": [9.97, 0, -1.11],
            "rotation": -1.570
        },
        {
            "position": [17.97, 0, -3.45],
            "rotation": 6.283
        },
        {
            "position": [6.51, 0, -8.9],
            "rotation": 1.570
        },
        {
            "position": [7.185, 0, -13.5],
            "rotation": 3.141
        },
        {
            "position": [9.15, 0, -9.6],
            "rotation": 10.131
        }
    ]
},
```

```jsx
import { useGLTF, Clone } from '@react-three/drei';

const CloneBuildings = ({ config }) => {
    const getModel = (modelName) => {
        const modelPath = `./models/global/${modelName}.glb`;
        return useGLTF(modelPath);
    };

    return (
        <>
            {config.map((item, index) =>
                item.coords?.map((p, positionIndex) => (
                    <Clone
                        castShadow
                        key={`${index}-${positionIndex}`}
                        object={getModel(item.name).scene}
                        position={p.position}
                        rotation-y={p.rotation}
                    />
                ))
            )}
        </>
    );
};

export default CloneBuildings;
```

Рендерим логотип Altel

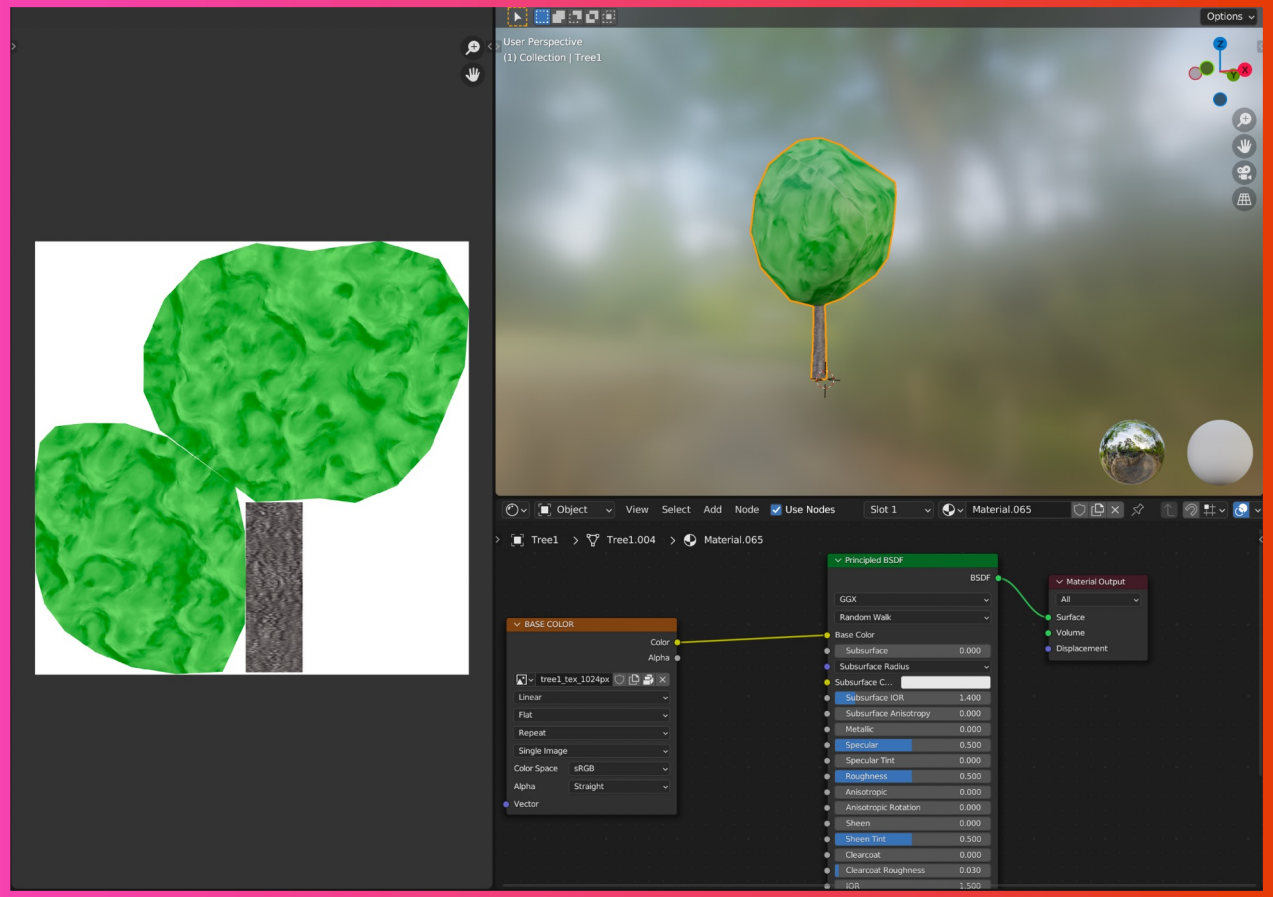


Рендерим логотип Altel

```
const sign = useGLTF('./models/global/altel-sign.glb');
const groupRef = useRef();

useFrame(() => {
    if (groupRef.current) {
        groupRef.current.children.map(i => (i.rotation.y += .02))
    }
})

return (
    <group ref={groupRef}>
        {
            altelSigns.signs?.map(i => (
                <Clone
                    key={`${i.x} ${i.y} ${i.z}`}
                    object={sign.scene}
                    position={[i.x, i.y, i.z]}
                />
            ))
        }
    </group>
)
```

Рендерим деревья

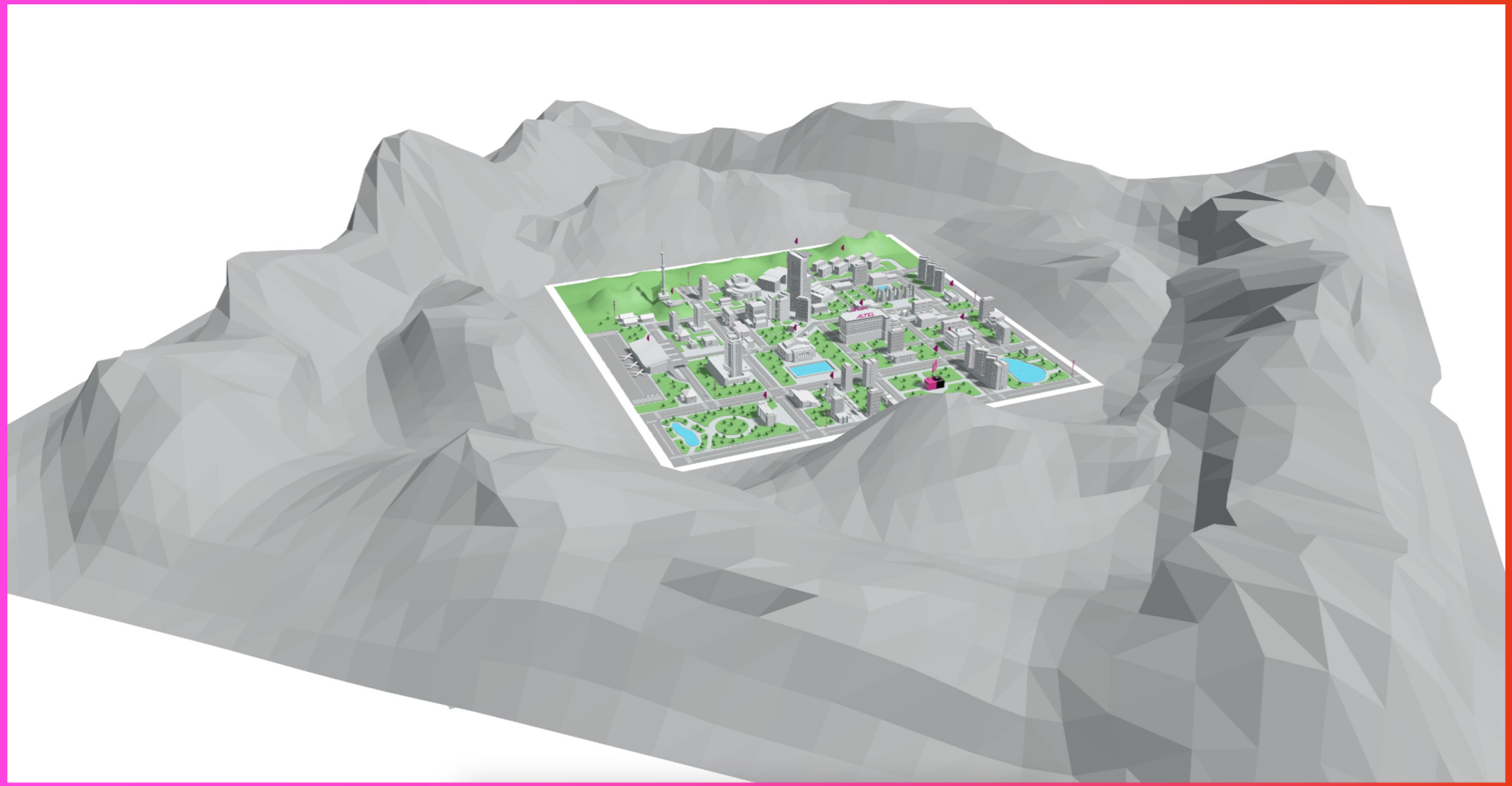
```
import { Clone, useGLTF } from '@react-three/drei'
import zones from '../../Configs/zones.json'

const Tree = () => {
    const tree = useGLTF('./models/plants/tree.glb')
    return (
        <>
            {
                zones.data?.map(i => (
                    <Clone
                        castShadow
                        key={`${i.x} ${i.z}`}
                        object={tree.scene}
                        position-x={i.x}
                        position-z={i.z}
                    />
                ))
            }
        </>
    )
}

export default Tree
```



Рендерим деревья

Добавляем фон

Добавляем горы



Контроллеры

Контроллеры

- Это инструменты, позволяющие управлять камерой или объектами в сцене интерактивным образом
- Предоставляют пользователю способ взаимодействовать с трехмерным миром
- Three.js предлагает несколько типов контроллеров для различных сценариев использования

MapControls

- Позволяет пользователю легко вращать, масштабировать и перемещать камеру
- Пользователь может управлять камерой с помощью мыши или сенсорного устройства

```
import { MapControls } from "@react-three/drei"

<MapControls
    minDistance={1}
    maxDistance={13}
    maxPolarAngle={1.3}
/>
```

Анимация транспорта

- Сделали несколько контуров
- Анимировали в 3ds max
- Запускали анимацию в three.js

Анимация транспорта

```
const train = useGLTF('./models/global/transports/train.glb')
const trainAnimation = useAnimations(train.animations, train.scene)
trainAnimation.actions.Anim_0.play()
```

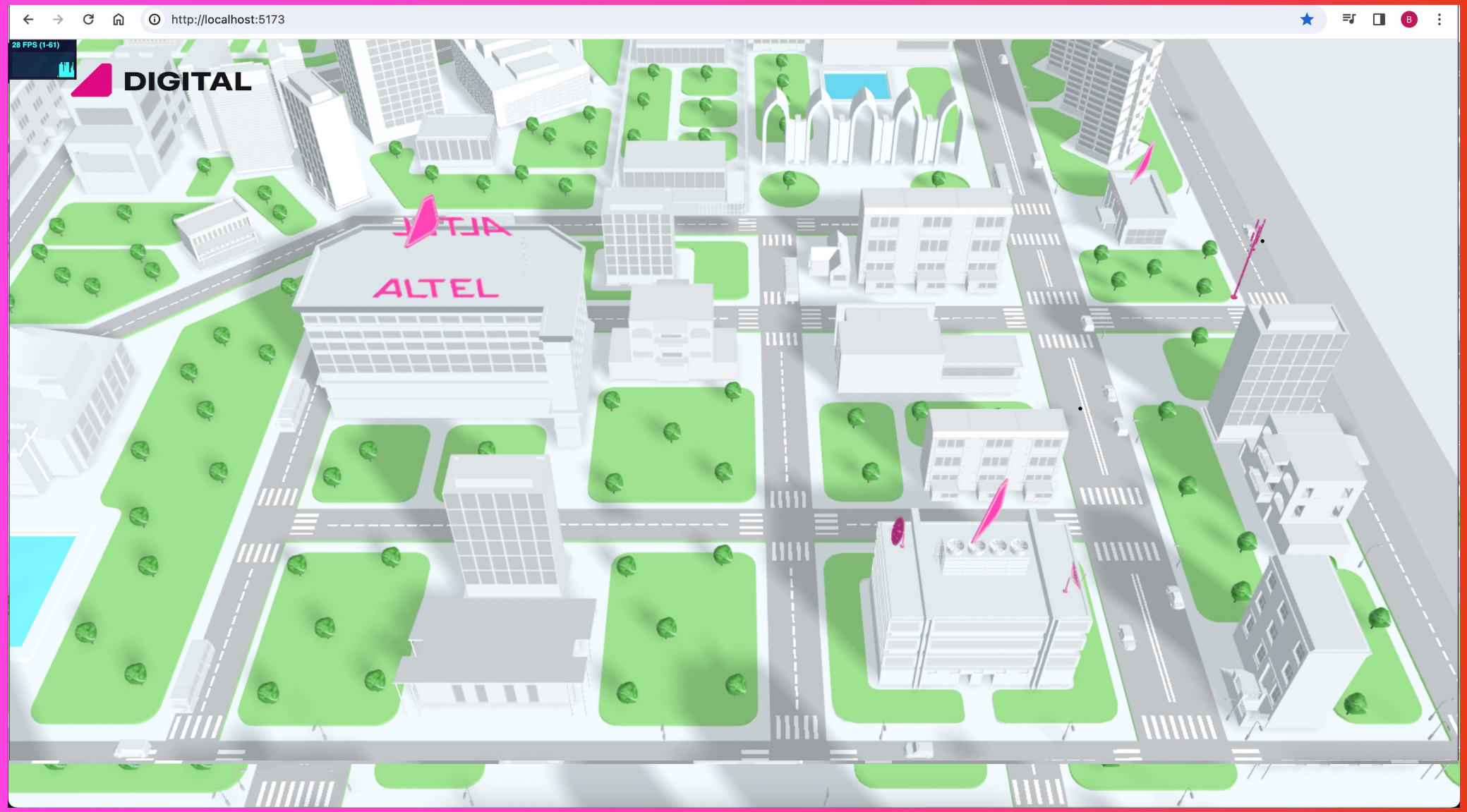
Реализуем клик по зданию

```
mainBuildings.map(i => {
    return (
        <primitive
            key={i.scene.mesh.name}
            object={i.scene}
            onPointerOver={handleOver}
            onPointerOut={handleOut}
            onClick={handleClick}
        />
    )
})
```

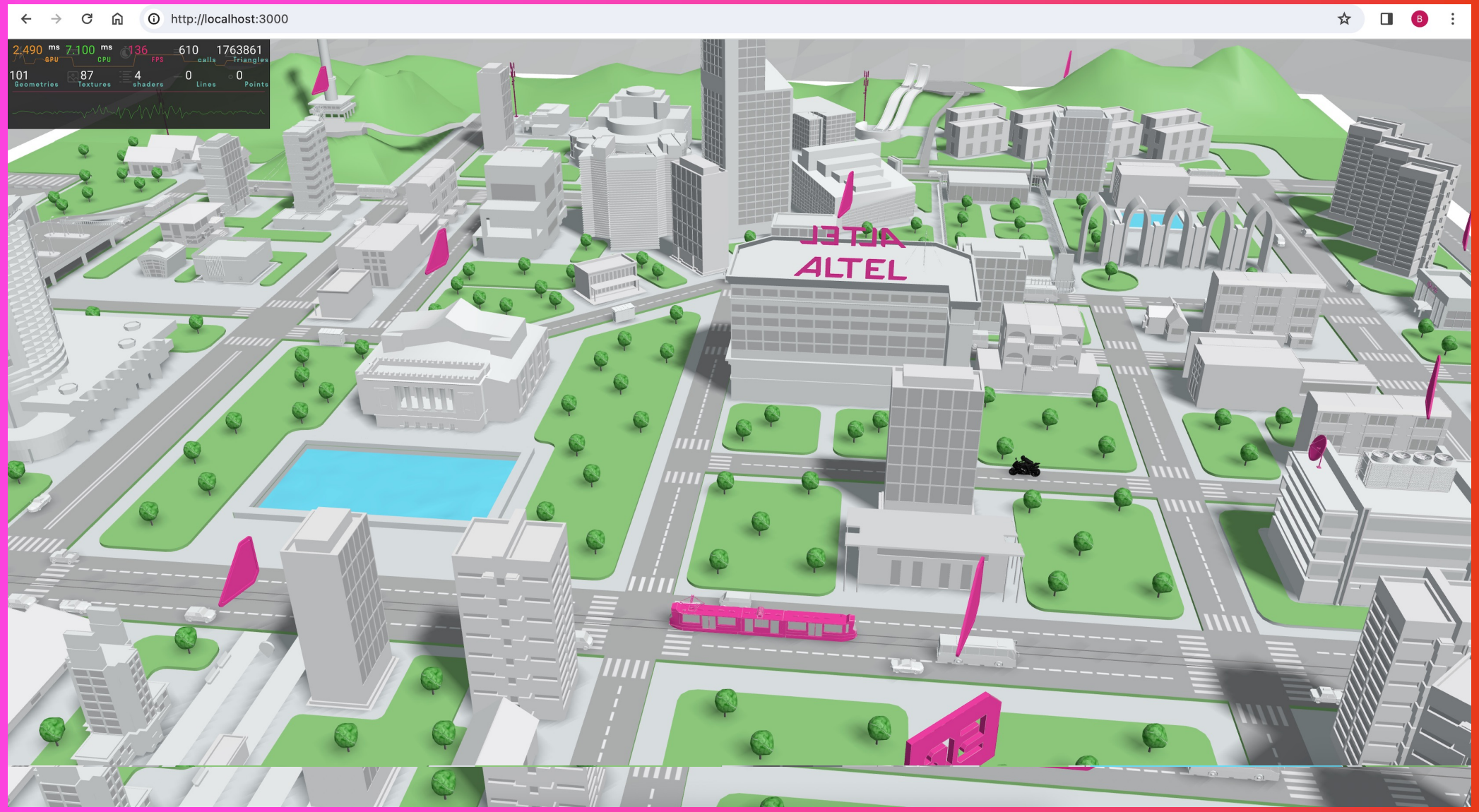
Raycaster

Raycaster может направлять луч в определенном направлении и проверять, какие объекты с ним пересекаются.

Оптимизация



Оптимизация



Shadow Map

- Разрешение теневых карт
- Ограничение области просмотра
- Алгоритмы фильтрации теней

Разрешение теневых карт

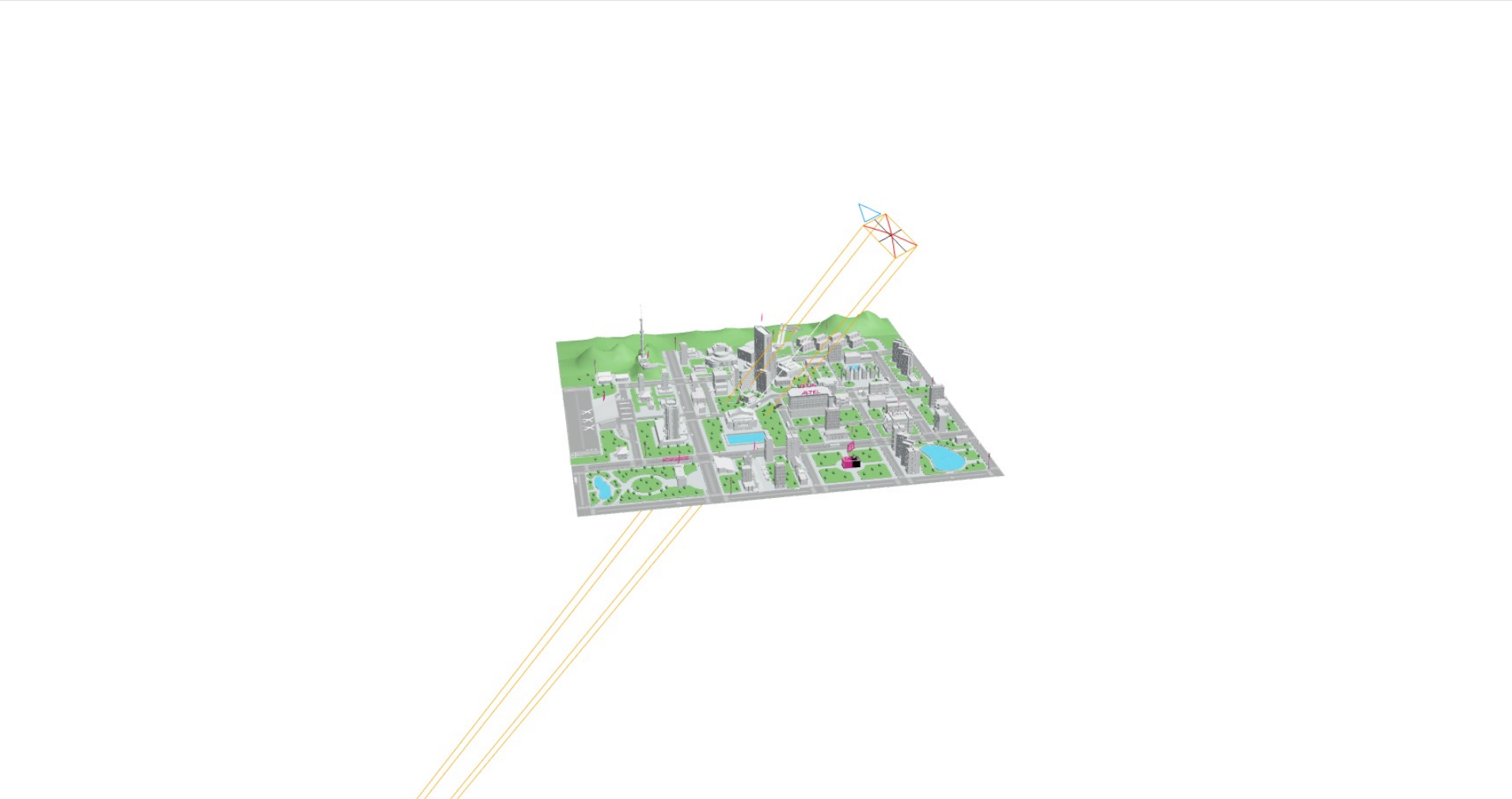
- Для больших экранов

```
directionalLight.shadow.mapSize.width = 1024
directionalLight.shadow.mapSize.height = 1024
```

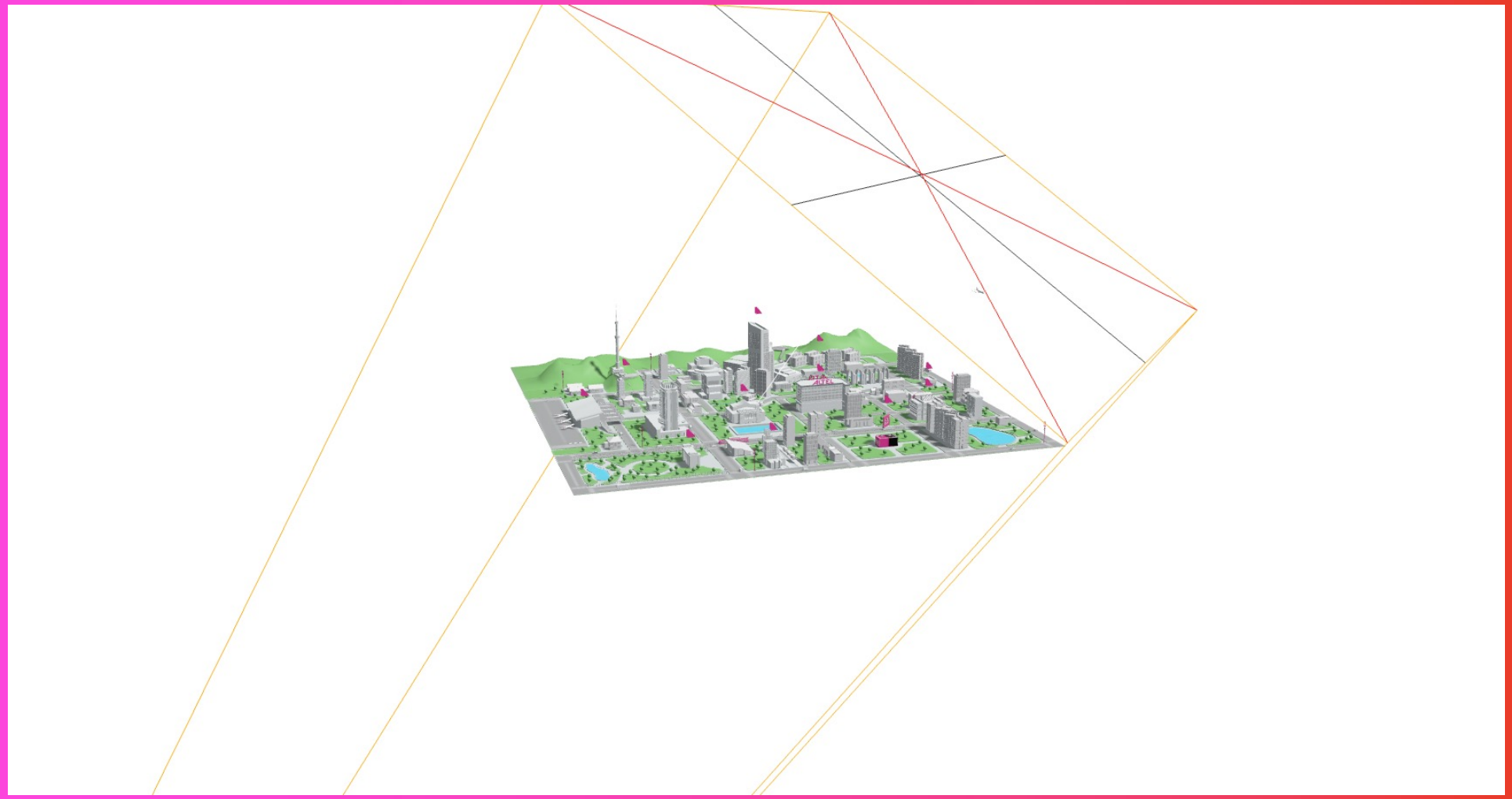
- Для мобильных устройств

```
directionalLight.shadow.mapSize.width = 512
directionalLight.shadow.mapSize.height = 512
```

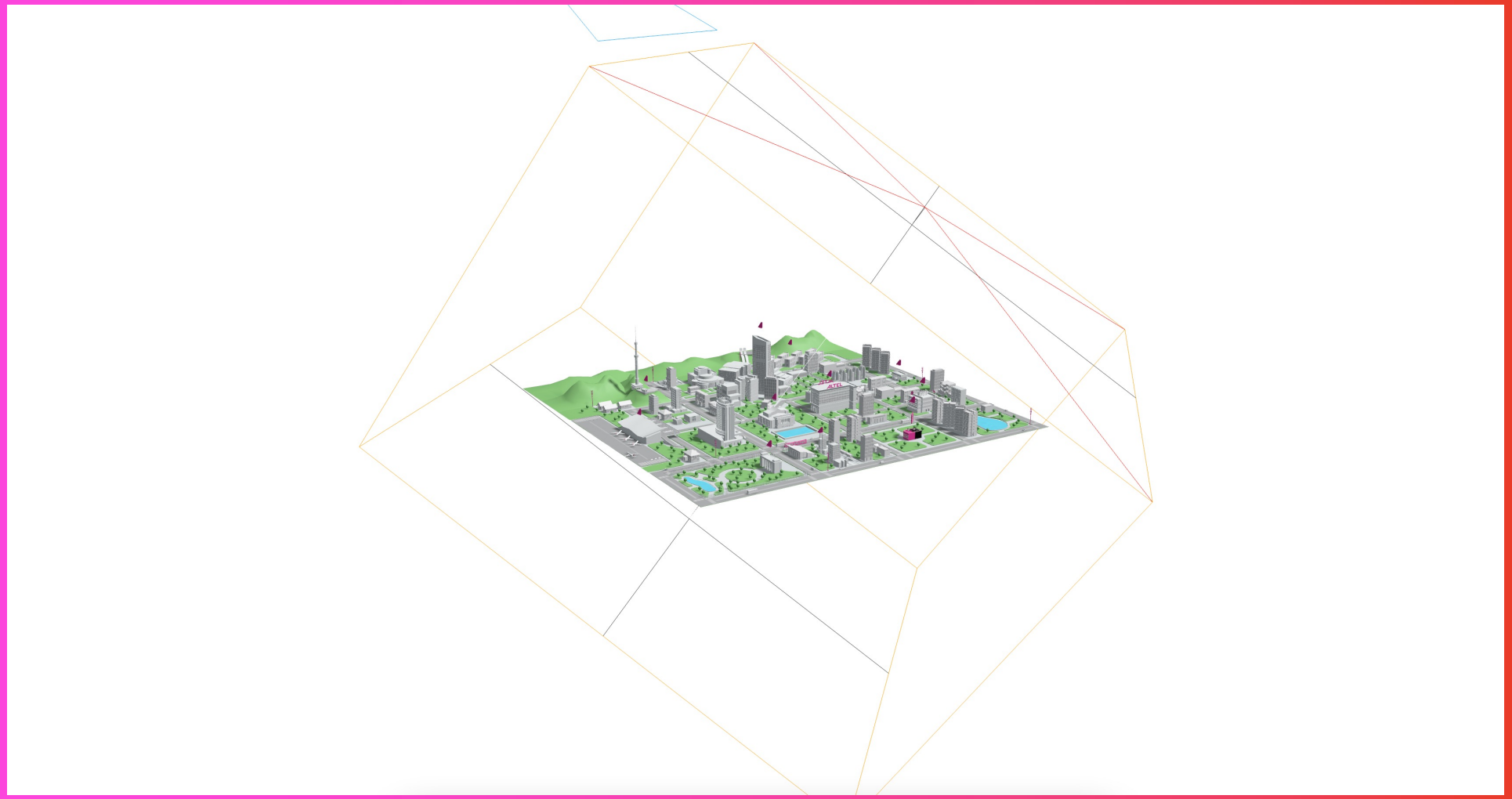
Ограничение области просмотра



Ограничение области просмотра



Ограничение области просмотра



Ограничение области просмотра

```
import { useRef } from 'react'
import * as THREE from 'three'
import { useHelper } from "@react-three/drei"


export const DirectionaLLight = () => {
    const camera = useRef()
    useHelper(camera, THREE.CameraHelper)

    return (
        <directionalLight castShadow position={[5, 10, 5]}>
            <orthographicCamera ref={camera} attach="shadow-camera" args={[-13, 13, 13, -13, 1, 20]} />
        </directionalLight>
    )
}
```

Алгоритм фильтрации теней

В three.js реализованы различные типы фильтрации теней, которые можно выбирать в зависимости от требований к качеству и производительности.

- BasicShadowMap
- PCFShadowMap
- PCFSoftShadowMap
- VSMShadowMap

Оптимизация

- Уменьшили качество и размеры текстур

Оптимизация

- Уменьшили качество и размеры текстур
- Сжимали 3D-модели через алгоритм Draco

baseMap.glb
Document - 834 KB

baseMap.glb
Document - 235 KB

Оптимизация

- Уменьшил качество и размеры текстур
- Сжимал 3D-модели через алгоритм Draco
- Подключил AdaptiveDpr

Оптимизация

- Уменьшили качество и размеры текстур
- Сжимали 3D-модели через алгоритм Draco
- Подключили AdaptiveDpr
- Копирование объектов просходил на вебе

Что проект ждет в будущем

- Хотим сделать день и ночь

Что проект ждет в будущем

- Хотим сделать день и ночь
- Хотим больше контента

Что проект ждет в будущем

- Хотим сделать день и ночь
- Хотим больше контента
- Хотим плавную анимацию при переходе между зданиями

Какие выводы я сделал

- Важность производительности

Какие выводы я сделал

- Важность производительности
- Значение освещения и теней

Какие выводы я сделал

- Важность производительности
- Значение освещения и теней
- Тестирование на разных устройствах

Какие выводы я сделал

- Важность производительности
- Значение освещения и теней
- Тестирование на разных устройствах
- Правильное управление ресурсами

Какие выводы я сделал

- Важность производительности
- Значение освещения и теней
- Тестирование на разных устройствах
- Правильное управление ресурсами
- Обучение и документация

THX
TIME FOR QA

https://digital.altel.kz

@borbekov