Щепотка SVG, нотка 3D* и карты

Владимир Грищенко

ОАО НПЦ ЭЛВИС

HOLY JS

ЭЛВИС

Владимир Грищенко

@BooooBka

- последнее несколько лет занимаюсь фронтендом
- считаю, что я - вечный junior
- страдаю синдромом самозванца
- прокрастинатор

ЭЛВИС

- процессоры:
  - robodeus - 8 ядер, 16нм
  - СКИФ - 4 ядра Cortex A53, 28нм
- IP видеокамеры
- радиационно стойкие компоненты
- РЛС
- embedded

ORWELL 2K

SIGUR

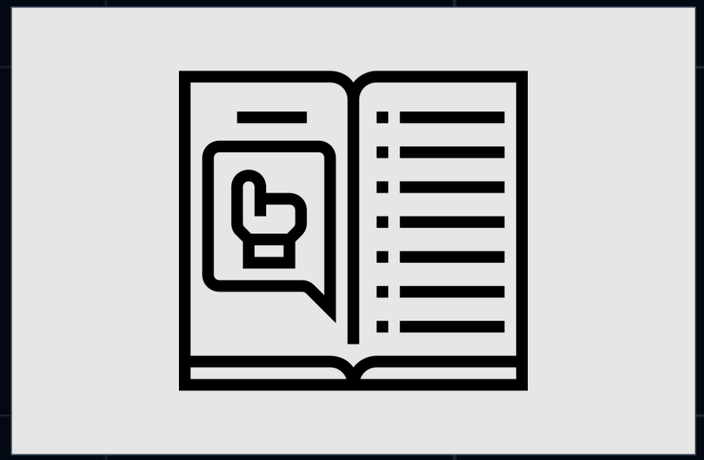
PSIM

BOLID
СИСТЕМЫ БЕЗОПАСНОСТИ

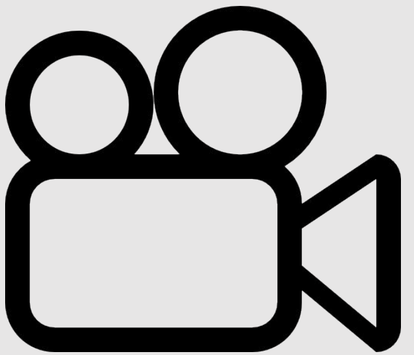
WEKEY

Рабочее место оператора

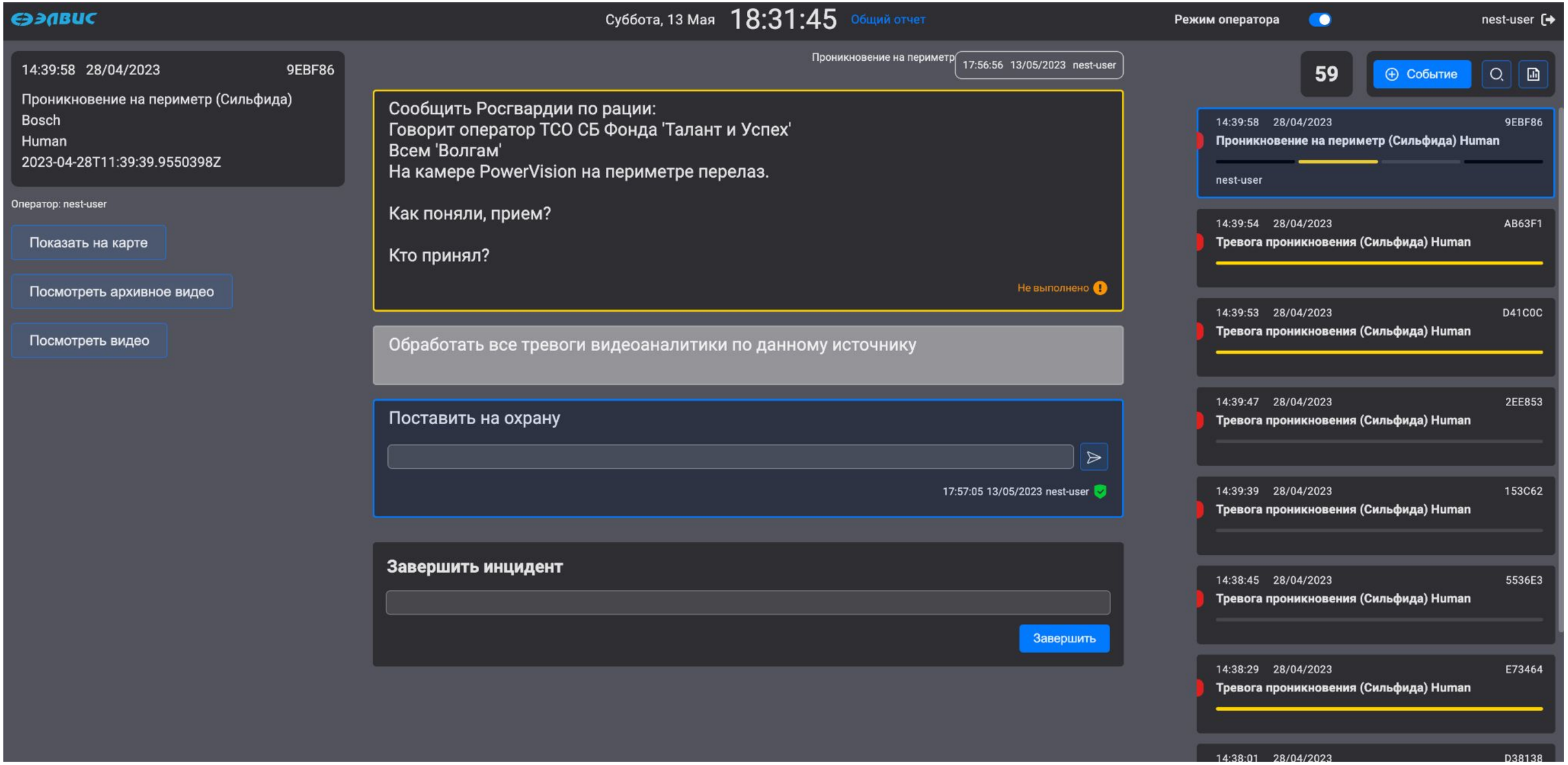
Карта



Инструкция



СВН


ЭЛВИС
Суббота, 13 Мая 18:31:45
Общий отчет
Режим оператора
nest-user
14:39:58 28/04/2023 9EBF86
Проникновение на периметр (Сильфида)
Bosch
Human
2023-04-28T11:39:39.9550398Z
Оператор: nest-user
Показать на карте
Посмотреть архивное видео
Посмотреть видео
Проникновение на периметр
17:56:56 13/05/2023 nest-user
Сообщить Росгвардии по рации:
Говорит оператор ТСО СБ Фонда 'Талант и Успех'
Всем 'Волгам'
На камере PowerVision на периметре перелаз.
Как поняли, прием?
Кто принял?
Не выполнено
Обработать все тревоги видеоаналитики по данному источнику
Поставить на охрану
17:57:05 13/05/2023 nest-user
Завершить инцидент
Завершить
59
Событие
14:39:58 28/04/2023 9EBF86
Проникновение на периметр (Сильфида) Human
nest-user
14:39:54 28/04/2023 AB63F1
Тревога проникновения (Сильфида) Human
14:39:53 28/04/2023 D41C0C
Тревога проникновения (Сильфида) Human
14:39:47 28/04/2023 2EE853
Тревога проникновения (Сильфида) Human
14:39:39 28/04/2023 153C62
Тревога проникновения (Сильфида) Human
14:38:45 28/04/2023 5536E3
Тревога проникновения (Сильфида) Human
14:38:29 28/04/2023 E73464
Тревога проникновения (Сильфида) Human
14:38:01 28/04/2023 D38138

- Геопривязанная система
- 3D здания
- Интерактивность зданий
- Отсутствие интернета на объектах внедрения
- Просмотр поэтажных планов
- Отображение ситуации в реальном времени

Карты

- Работал с различными API
- Тайлы брались с серверов провайдеров
- Не понимал как работать с 3D

CESIUM

- полноценное 3D
- большое комьюнити
- богатое API
- умеет работать с gltf

MapLibre

- использует 2.5D
- отличная документация с примерами
- быстрый старт локального тайлового сервера
- умеет работать с gltf
- **maplibregl** - форк mapbox 1 версии

MaplibreGL/React

- существует готовое решение: react-map-gl
- для задания параметров/слоев используется JSX
- готовое решение использует внутри себя mapbox v2
- можно сделать форк и использовать MapLibreGL

MaplibreGL/React

- Написали свою обертку
- Объект карты храним в mobx
- Явно вызываем методы для работы с картой

```
const mapNode = useRef(null);
useEffect(() => {
    const node = mapNode.current;

    if (typeof window === 'undefined' || node === null) return;

    const mapboxMap = new Map({
        container: node,
        ...MAP_DEFAULT_OPTIONS,
        ...props.options,
    });

    mapStore.setMap(mapboxMap);

    mapboxMap.once('load', onMapLoaded);

    return () => {
        mapboxMap.remove();
        if (props.onMapRemoved) props.onMapRemoved();
    };
}, []);

return <div ref={mapNode}/>;
}
```

Локальный тайловый сервер

TileServer-GL

- поддержка векторных тайлов(mbtiles)
- легкость в запуске в docker контейнере

```
docker run -p 3030:8080 -v ./tiles:/data maptileserver
```

- обширные настройки вида слоев карты
- возможность раздачи тайлов из нескольких источников
- написан на Node JS
- есть аналог mbtileserver написанный на Go

mbtiles

- формат хранения наборов тайлов
- открытая спецификация
- работает на базе бд SQLite
- спецификация давно не обновлялась

mbtiles

- существуют готовые на сайте maptiler.com
- бесплатные версии не актуальны
- актуальные тайлы стоят денег

Добываем тайлы

- ресурс Geofabrik (https://download.geofabrik.de)
- содержит данные OpenStreetMap для скачивания
- доступны форматы: osm.pbf, shp.zip, osm.bz2

Tilemaker

- opensource
- нет ограничений на коммерческое использование
- представляет собой 1 исполняемый файл
- умеет конвертировать osm.pbf в mbtiles
- умеет фильтровать и обрабатывать слои карты

```
./tilemaker --input ../../some.osm.pbf --output ../../some.mbtiles
    --process ../resources/process-openmaptiles.lua --config ../resources/config-openmaptiles.json
```

Tilemaker

- не поддерживает OSM diff
- не умеет создавать рельеф
- для конвертации морей и океанов требуется подключать отдельные ресурсы.
- конвертация с водной гладью достаточно ресурсозатратно

Osmium

- инструмет для работы с данными OSM
- используем относительно небольшие куски карты
- позволяет вырезать куски OSM карты

```
osmium extract --bbox=37.028746,55.900543,37.525416,56.046443
    --set-bounds --strategy=smart ./russia.osm.pbf --output ./russia-output.osm.pbf
```

- На карте может не быть здания
- Отрисовываем здание на osm.org, экспортируем изменения

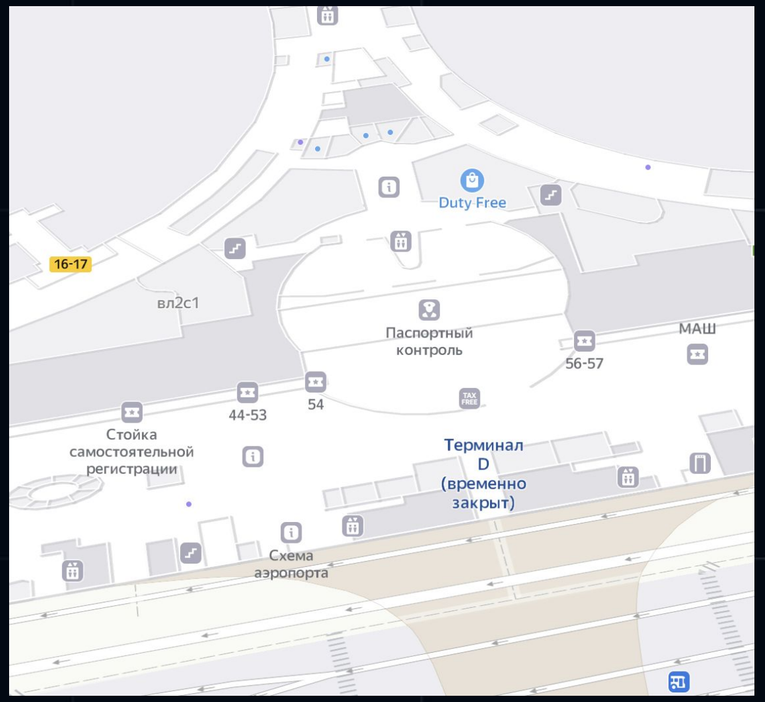
```
osmium apply-changes --output=russia-new.osm.pbf russia.osm.pbf changes.osc
```

- Обновляем тайлы
- Конвертировать в mbtiles мы уже умеем

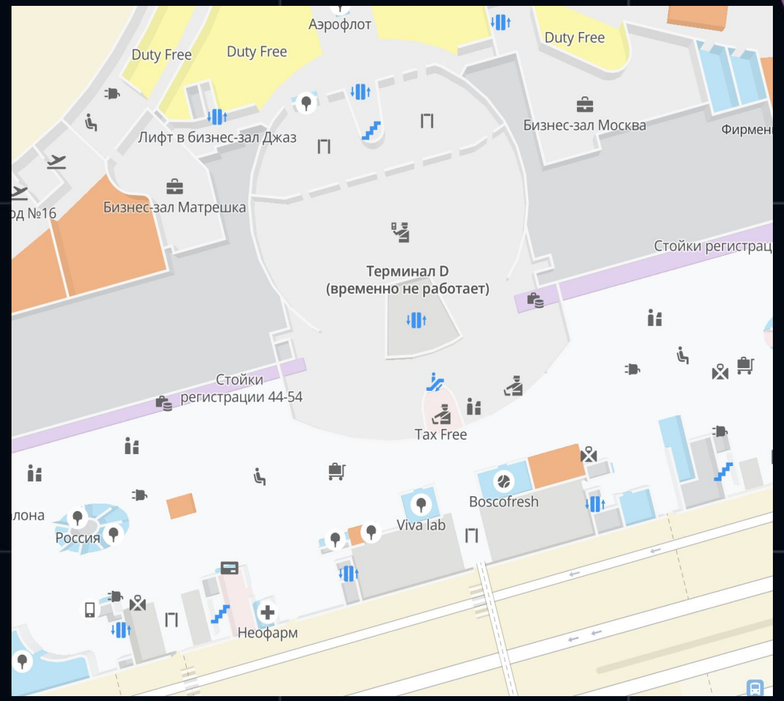
- не люблю UI
- cli автоматизируется лучше

Щепотка SVG

Отображаем большие svg



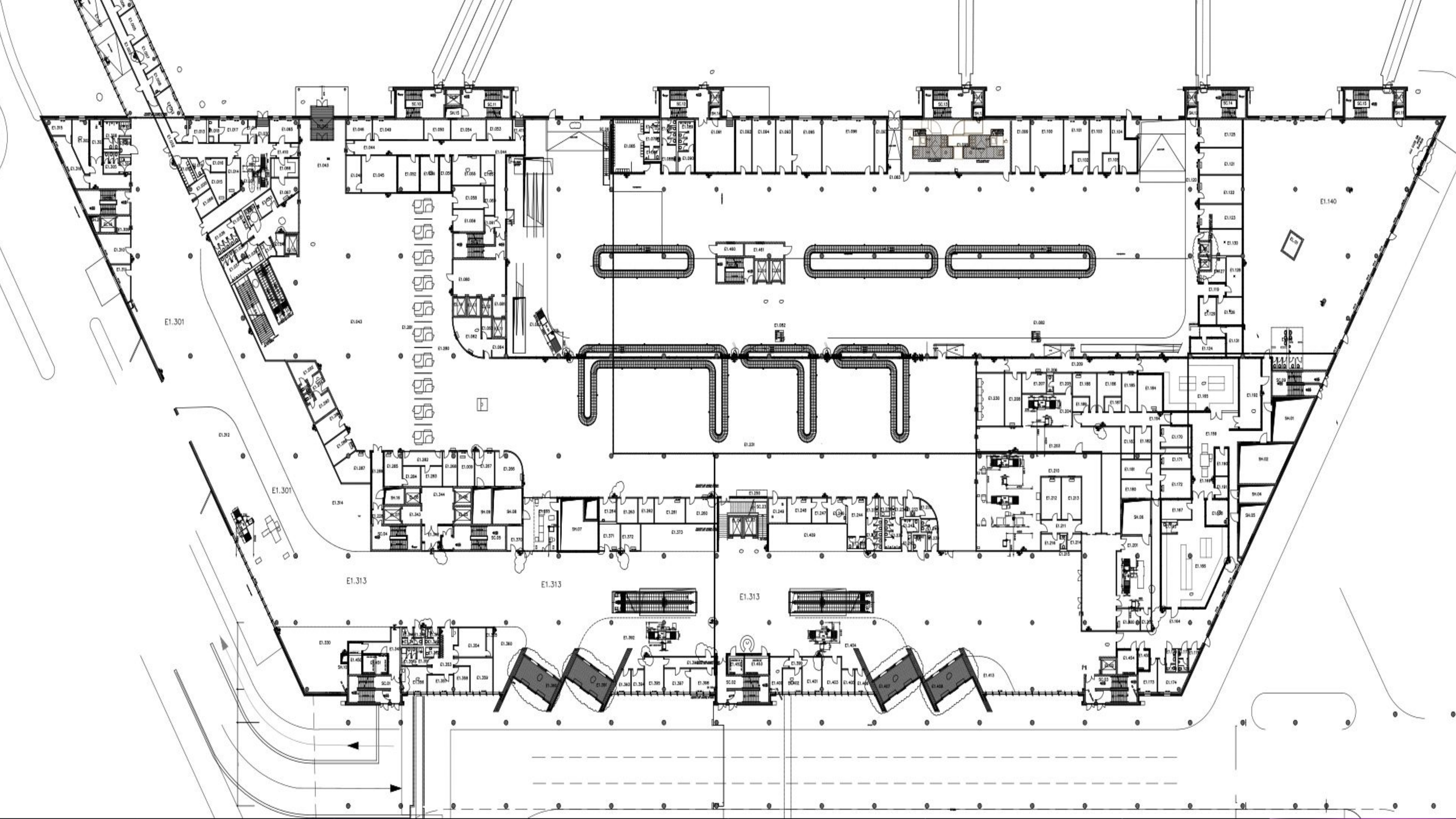
Яндекс карты



2GIS

Планы этажей

- расположение устройств есть в документации на каждый объект
- планы этажей для документации рисуются в AutoCad
- autoCad позволяет экспортировать план в PDF
- для экспорта pdf -> svg можно использовать inkscape


E1.301
E1.301
E1.313
E1.313
E1.313
E1.140

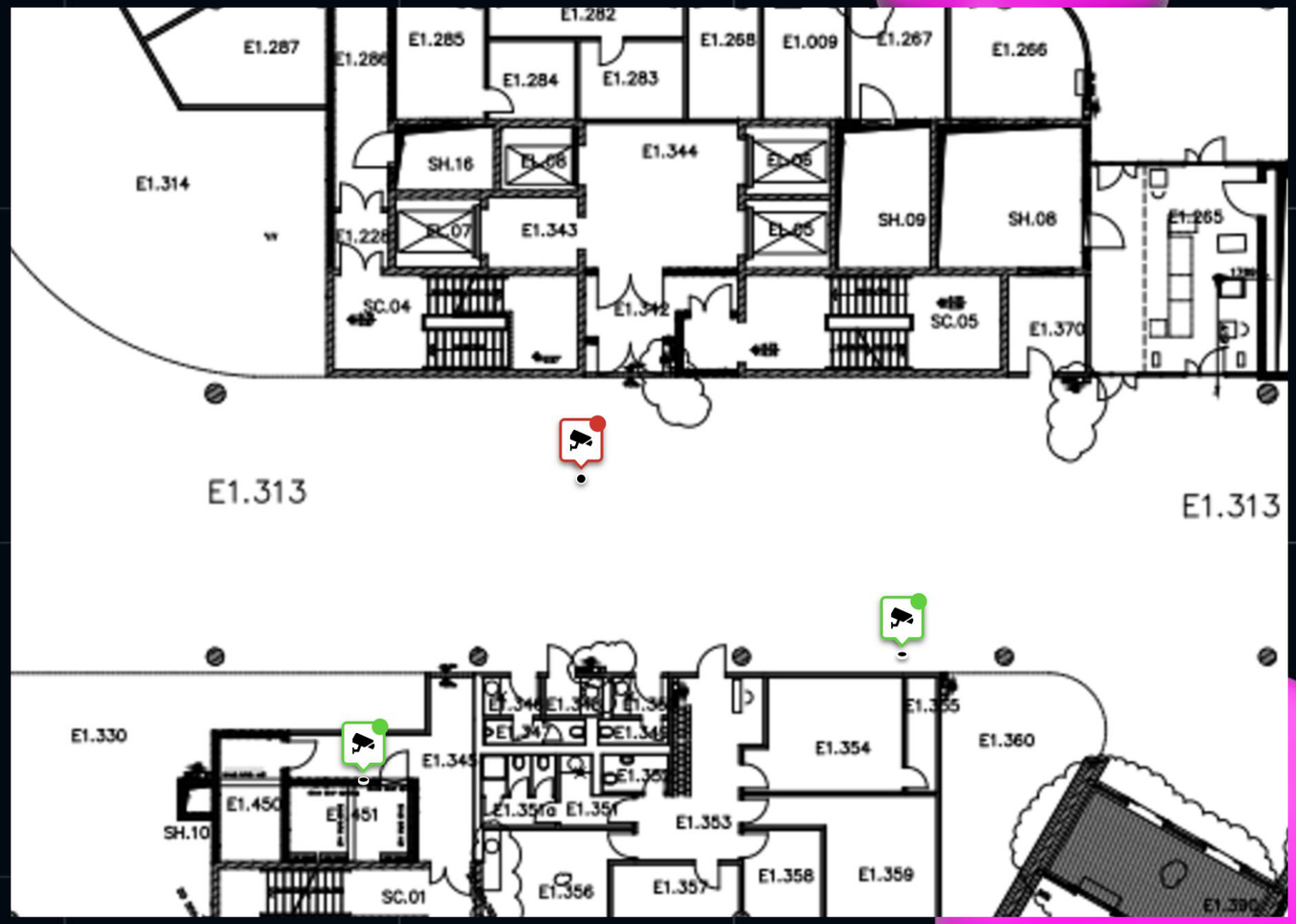
Планы этажей

- размеры svg файлов 20-150мб
- можно пытаться их оптимизировать и использовать готовый viewer
- нужна геопривязка

- для просмотра можно использовать карту
- картинку разрезать на тайлы

Отображаем большие svg

- растягиваем картинку
- конвертируем картинку в PNG
- разбиваем на тайлы
- формируем preview-плана
- NestJS/SharpJS/docker/swagger



SharpJS

```
await resize(file.path, IMAGE_SIZE)
        .tile({
                size: TILE_SIZE,
                depth: 'onetile',
                container: 'fs',
                background: BG_DEFAULT_COLOR,
                layout: 'google',
        })
        .toFile(`${path}`, (err) => {
                if (err) {
                        console.log(err);
                        return;
                }
        })]);
```

Maplibre + png tiles

- добавляем ресурс с растровыми тайлами
- для просмотра используем компонент обертку созданные на предыдущих слайдах

```
'version': 8,
'sources': {
        'raster-tiles': {
                'type': 'raster',
                'tiles': [
                        `/tiles/${planId}/{z}/{y}/{x}.png`,
                ],
                'tileSize': 256,
        },
},
'layers': [
        {
                'id': 'plan-tiles',
                'type': 'raster',
                'source': 'raster-tiles',
        },
],
```

Нотка 3D*

- карта это canvas

- нам нужен доступ к WebGLRenderingContext

Custom Layer Interface

- позволяет рендерить объекты в webgl контекст карты.
- дает доступ к камере карты
- добавляется как слой карты
- проецирует меркатор координаты в gl [0,0] - верхний левый угол, [1, 1] - правый нижний

Custom Layer Interface

```
export type CustomLayerInterface = {
    id: string,
    type: "custom",
    renderingMode: "2d" | "3d",
    render: CustomRenderMethod,
    prerender: ?CustomRenderMethod,
    renderToTile: ?(gl: WebGLRenderingContext, tileId: MercatorCoordinate) => void,
    shouldRerenderTiles: ?() => boolean,
    onAdd: ?(map: Map, gl: WebGLRenderingContext) => void,
    onRemove: ?(map: Map, gl: WebGLRenderingContext) => void
}
```

- для каждой модели:
  - сцена
  - камера
  - raycaster
  - расположение

```
export class CustomLayer {
    scene: THREE.Scene;
    camera: THREE.Camera;
    raycaster: THREE.Raycaster;
    modelAsMercatorCoordinate: MercatorCoordinate;
    modelRotate: number[];


    constructor(data: any) {
        this.scene = new THREE.Scene();
        this.camera = new THREE.Camera();
        this.raycaster = new THREE.Raycaster();

        this.modelAsMercatorCoordinate = MercatorCoordinate.fromLngLat(
            data.position,
            0,
        );

        this.modelRotate = [Math.PI / 2, 0, 0];
    }
};
```

Задаем положение модели

- поворота
- позиции
- масштаба
- реальные метры модели преобразуем в mercator

```
position = DEFAULT_MAP_CENTER;

private  modelAsMercatorCoordinate = MercatorCoordinate.fromLngLat(
    this.position,
    0,
);

modelRotate = [Math.PI / DIVIDER, 0, 0];

private modelTransform = {
    translateX: this.modelAsMercatorCoordinate.x,
    translateY: this.modelAsMercatorCoordinate.y,
    translateZ: this.modelAsMercatorCoordinate.z,
    rotateX: this.modelRotate[0],
    rotateY: this.modelRotate[1],
    rotateZ: this.modelRotate[2],
    scale: this.modelAsMercatorCoordinate.meterInMercatorCoordinateUnits(),
};
```

Добавляем модель на карту

- метод onAdd имеет доступ к gl контексту карты
- в приложении используем gltf модели
- для загрузки моделей используем Three GLTFLoader
- добавляем модель на сцену

```
loader.load(
    this.getUrl(),
    model => this.scene.add(model.scene),
);
```

Добавляем модель на карту

- для отображения сцены Threejs WebGLRenderer
- сцену добавляем в уже существующий gl контекст карт

```
this.renderer = new THREE.WebGLRenderer({
    canvas: map.getCanvas(),
    context: gl,
    antialias: true,
});
```

Подсветка сцены

```
const directionalLight = new THREE.DirectionalLight(0xffffff);
directionalLight.position.set(0, -70, 100).normalize();
this.scene.add(directionalLight);

const directionalLight2 = new THREE.DirectionalLight(0xffffff);
directionalLight2.position.set(0, 70, 100).normalize();
this.scene.add(directionalLight2);
```

Render method

- вызывается при каждой перерисовке карты
- имеет доступ к
  - WebGLRendering контексту карты
  - матрице камеры карты
- можно вызывать перерендер вызывая метод triggerRepaint

Кастомный рендерер карты

- говорим камере откуда мы смотрим
- отрисовываем сцену на карте
- перерисовываем карту
- добавляем параметры для изменения размеров/угла поворота

```
const rotationX = new THREE.Matrix4().makeRotationAxis(
    new THREE.Vector3(1, 0, 0),
    this.modelTransform.rotateX,
);
const rotationY = new THREE.Matrix4().makeRotationAxis(
    new THREE.Vector3(0, 1, 0),
    this.modelTransform.rotateY,
);
const rotationZ = new THREE.Matrix4().makeRotationAxis(
    new THREE.Vector3(0, 0, 1),
    this.modelTransform.rotateZ,
);

const m = new THREE.Matrix4().fromArray(matrix);
const l = new THREE.Matrix4()
    .makeTranslation(
        this.modelTransform.translateX,
        this.modelTransform.translateY,
        this.modelTransform.translateZ,
    )
    .scale(
        new THREE.Vector3(
            this.modelTransform.scale * this.scale,
            -this.modelTransform.scale * this.scale,
            this.modelTransform.scale * this.scale,
        ),
    )
    .multiply(rotationX)
    .multiply(rotationY)
    .multiply(rotationZ);

this.camera.projectionMatrix = m.multiply(l);
this.renderer.resetState();
this.renderer.render(this.scene, this.camera);
this.map.triggerRepaint();
```

Добавляем на карту

```
createGLTFModel = () => {
    return {
        id: this.getID(),
        type: 'custom',
        renderingMode: '3d',
        onAdd: onAdd.bind(this),
        render: renderGLTFModel.bind(this),
    };
};
```

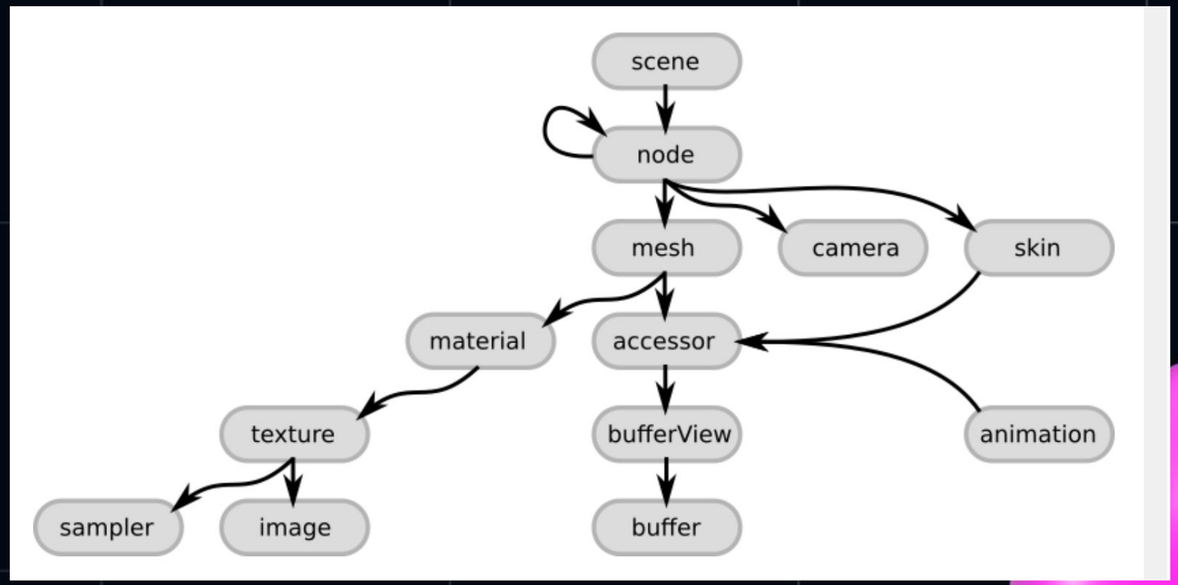
```
const mapLayer = layer.createGLTFModel();
this.map.addLayer(mapLayer);
```


14 c2
14 c3
14 c4
14 c13
14 c15

Gltf/Glb

- формат для хранения 3D сцен и моделей.
- структура формат JSON
- GLB - бинарная версия формата, содержит в себе не только структуру, но и материалы
- строгая структура формата
- правосторонняя система координат
- углы в радианах
- линейные расстояния в метрах
- положительное вращение

по часовой стрелке



Откуда брать модели?!

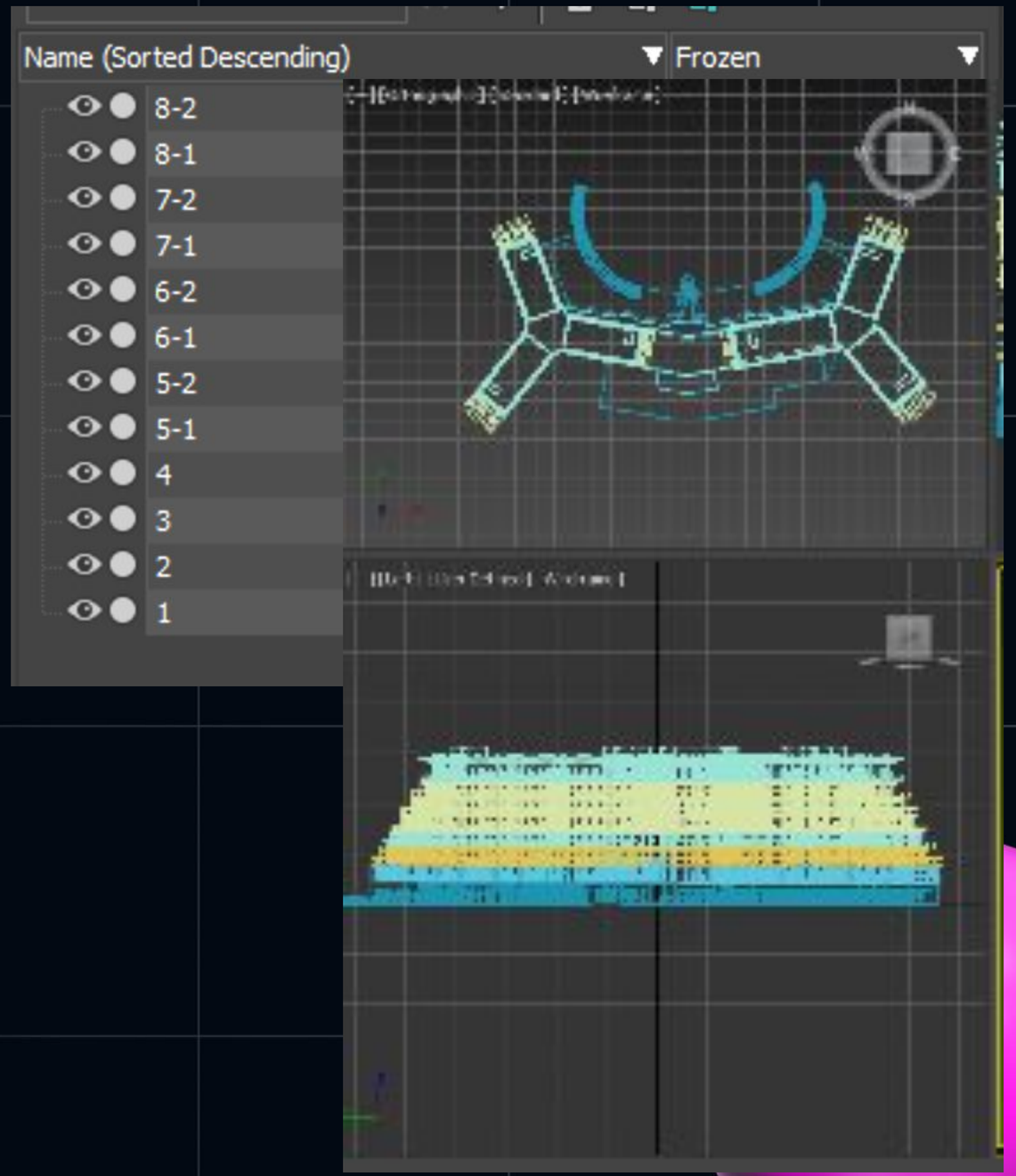
- в предыдущий реализации использовались скриншоты из 3DsMax.
- в моем доступе оказались модели в формате *.max
- в команде есть дизайнер способный их отрисовывать
- для точности необходимы раскладки фасадов

Babylon file exporter

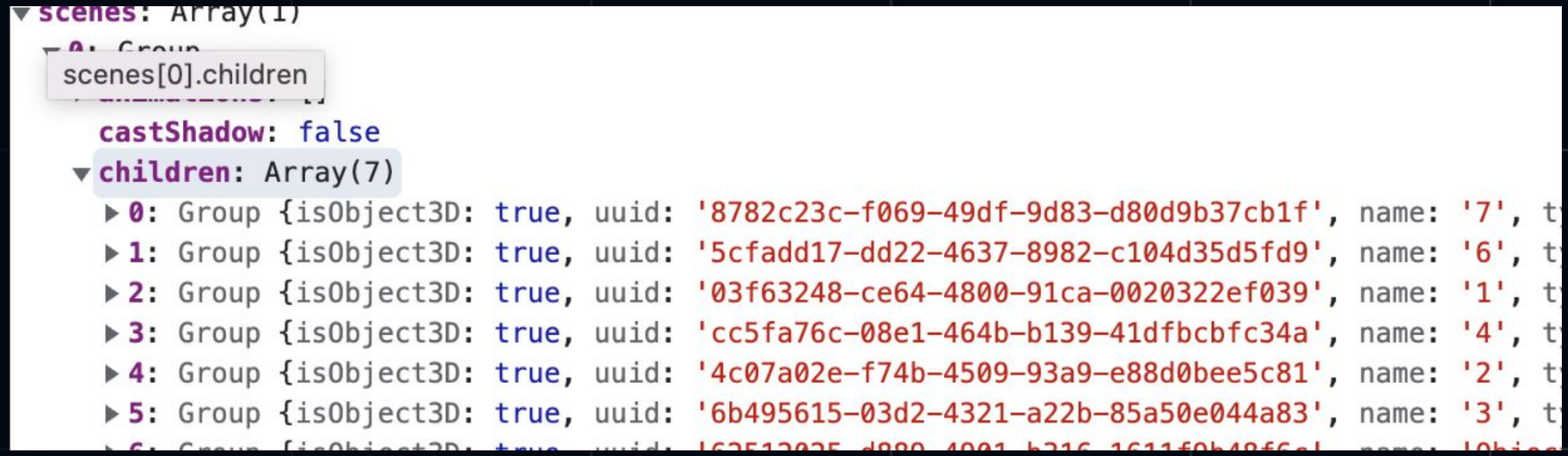
- конвертирует *.max модель в gltf/glb
- в качестве стандарта используется gLTF 2.0
- маршрут конвертации max -> babylon -> gltf
- не экспортируется подсветка модели
- есть проблемы с потерей материалов

Готовим модели

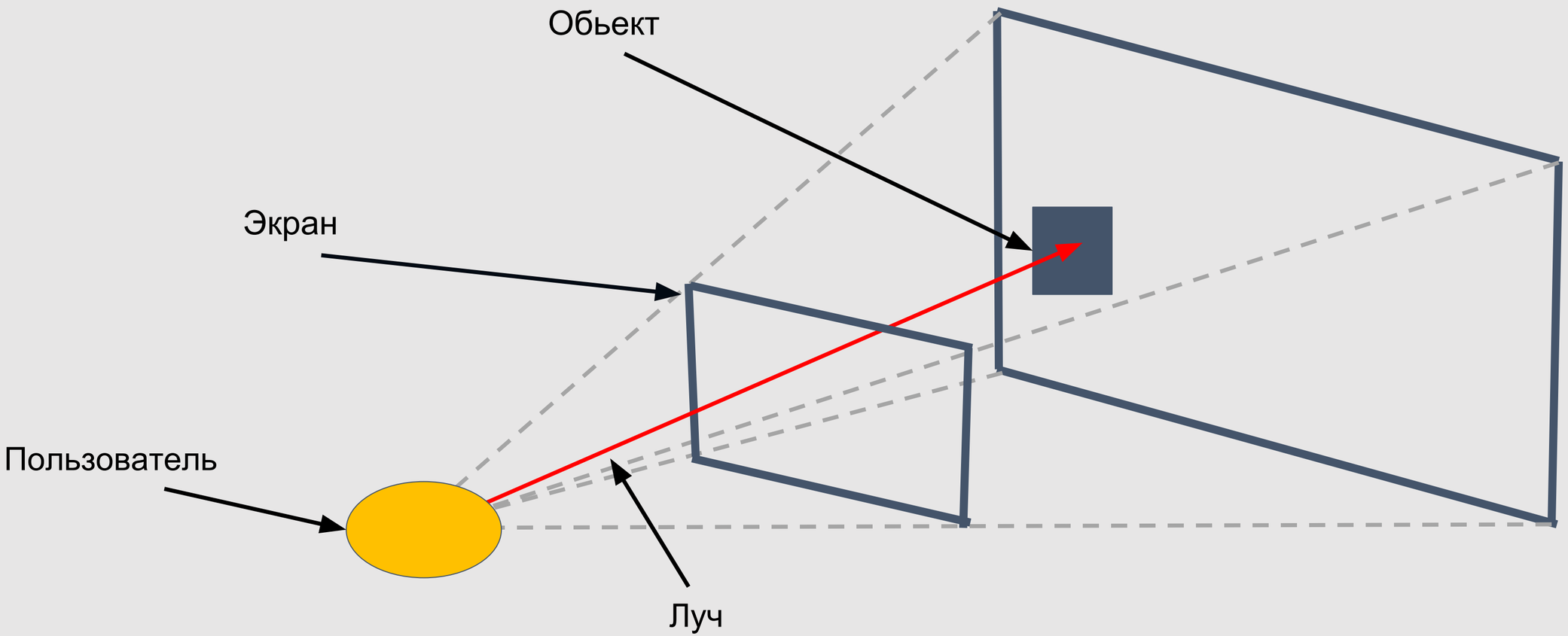
- модель должна быть отцентрована по осям
- объединяем все примитивы модели в один объект формата Editable Poly
- разрезаем модель на этажи
- даем имена согласно номеру этажа



- модель будет превращена в gltf json с children типа Group для каждого этажа
- порядок этажей не гарантирован, первым будет последний измененный poly 3D Max модели



Raycast



Ищем модель на карте

- вычисляем положение курсора в нормализованной системе координат
- системы координат сцены и карты не совпадают
- получаем массив объектов которые пересек луч

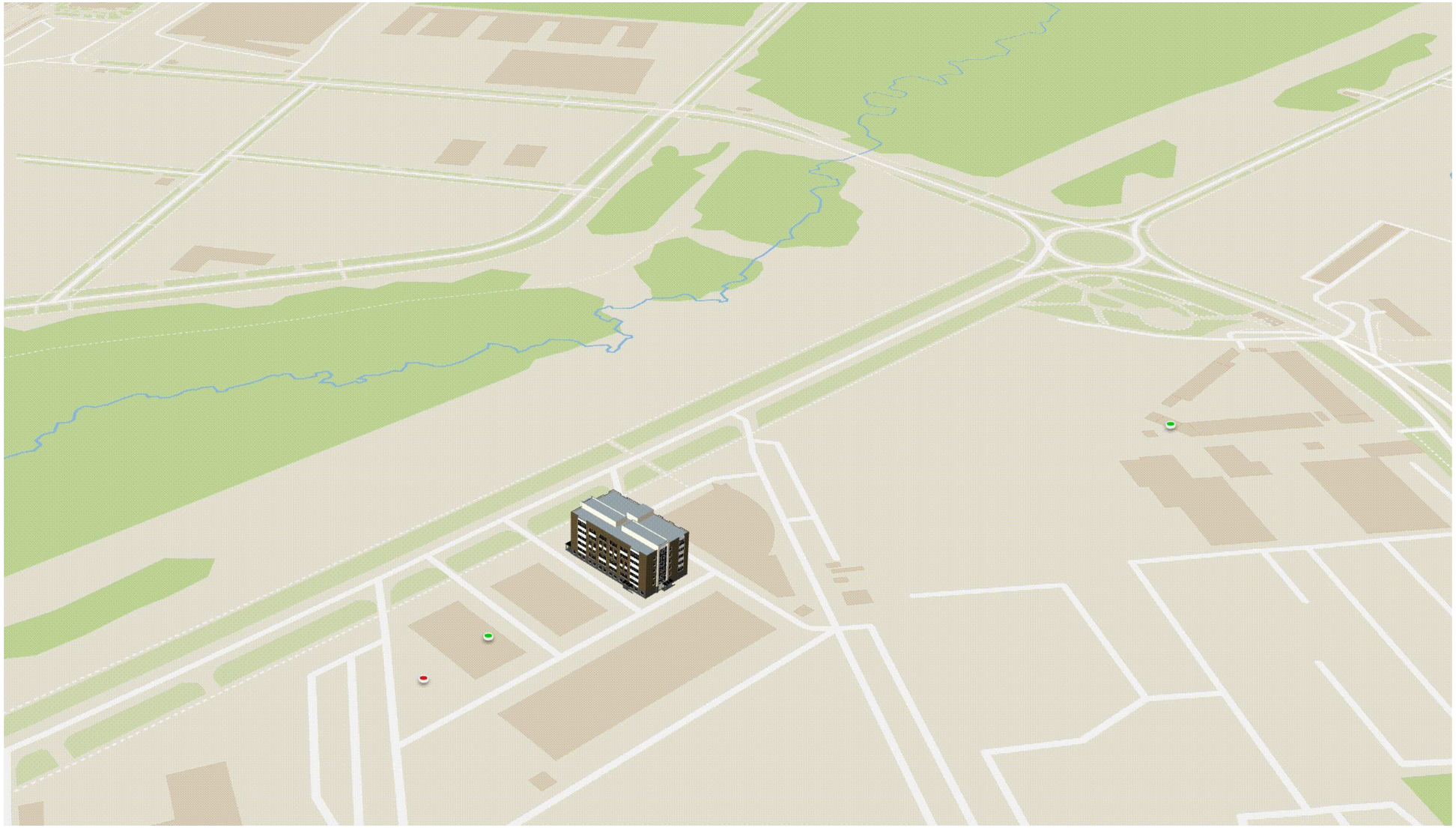
Ищем модель на карте

```
const point = e.point;
const mouse = new THREE.Vector2();
mouse.x = ( point.x / this.map.transform.width ) * DIVIDER - 1;
mouse.y = 1 - ( point.y / this.map.transform.height ) * DIVIDER;

const camInverseProjection = new THREE.Matrix4().copy(this.camera.projectionMatrix).invert();
const cameraPosition = new THREE.Vector3().applyMatrix4(camInverseProjection);
const mousePosition = new THREE.Vector3(mouse.x, mouse.y, 1).applyMatrix4(camInverseProjection);
const viewDirection = mousePosition.clone().sub(cameraPosition).normalize();
this.raycaster.set(cameraPosition, viewDirection);

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

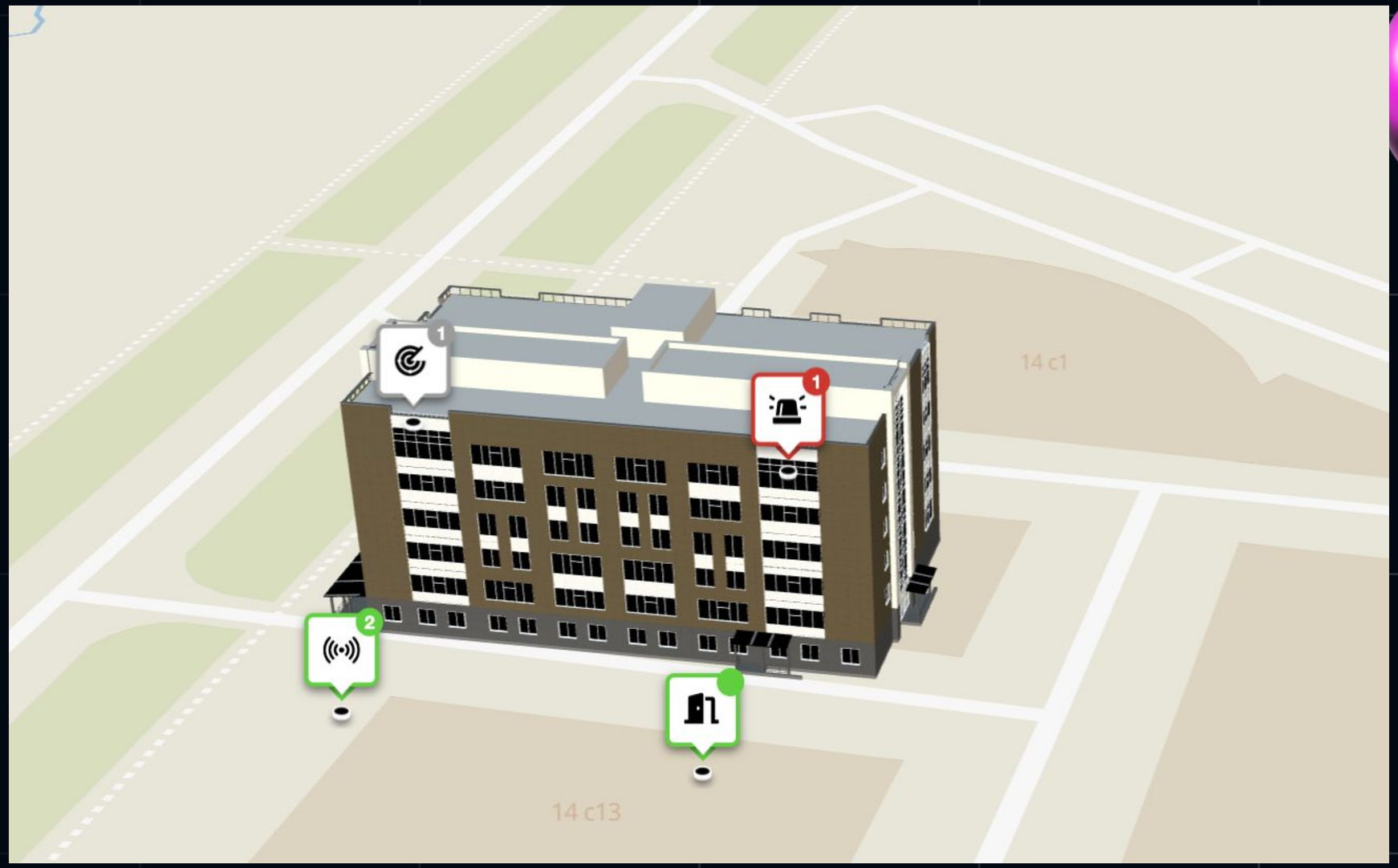
Подсвечиваем модель

- сохраняем материалы Mesh модели в хеш таблицу
- обходим всю модель и меняем материалы child на подсвеченные либо возвращаем исходные

```
model.scene.traverse((child: SceneChild) => {
    if(child.type === 'Mesh') {
        this.highlightedModelMaterials[child.uuid] = child.material.clone();
        const highlightedMaterial = new THREE.MeshStandardMaterial({
            color: color,
            side: child.material.side,
        });
        child.material = highlightedMaterial;
    }
});
```



Проблема со слоями



BBox

- строим Box3 для каждой модели
- получаем координаты углов

```
const boundingBox = new  THREE.Box3();
const buildingModelChildren = building.model.scene.children;
boundingBox.setFromObject(buildingModelChildren[buildingModelChildren.length - 1]);

const low = boundingBox.min;
const high = boundingBox.max;

const corner1 = new THREE.Vector3(low.x, low.y,  low.z);
coordes.push(getCoordsFromVector(corner1.project(building.model.camera), map));

const corner8 = new THREE.Vector3(high.x, high.y, high.z);
coordes.push(getCoordsFromVector(corner8.project(building.model.camera), map));

return coordes;
```

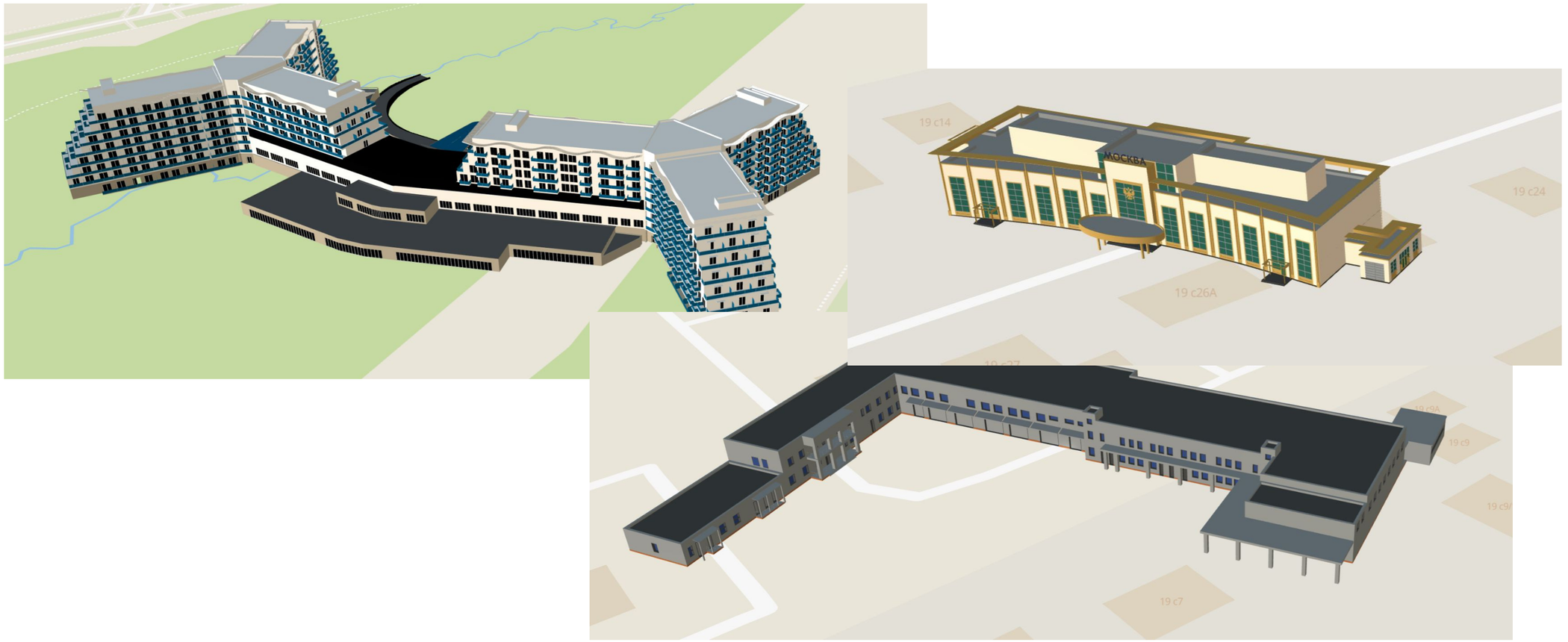
BBox

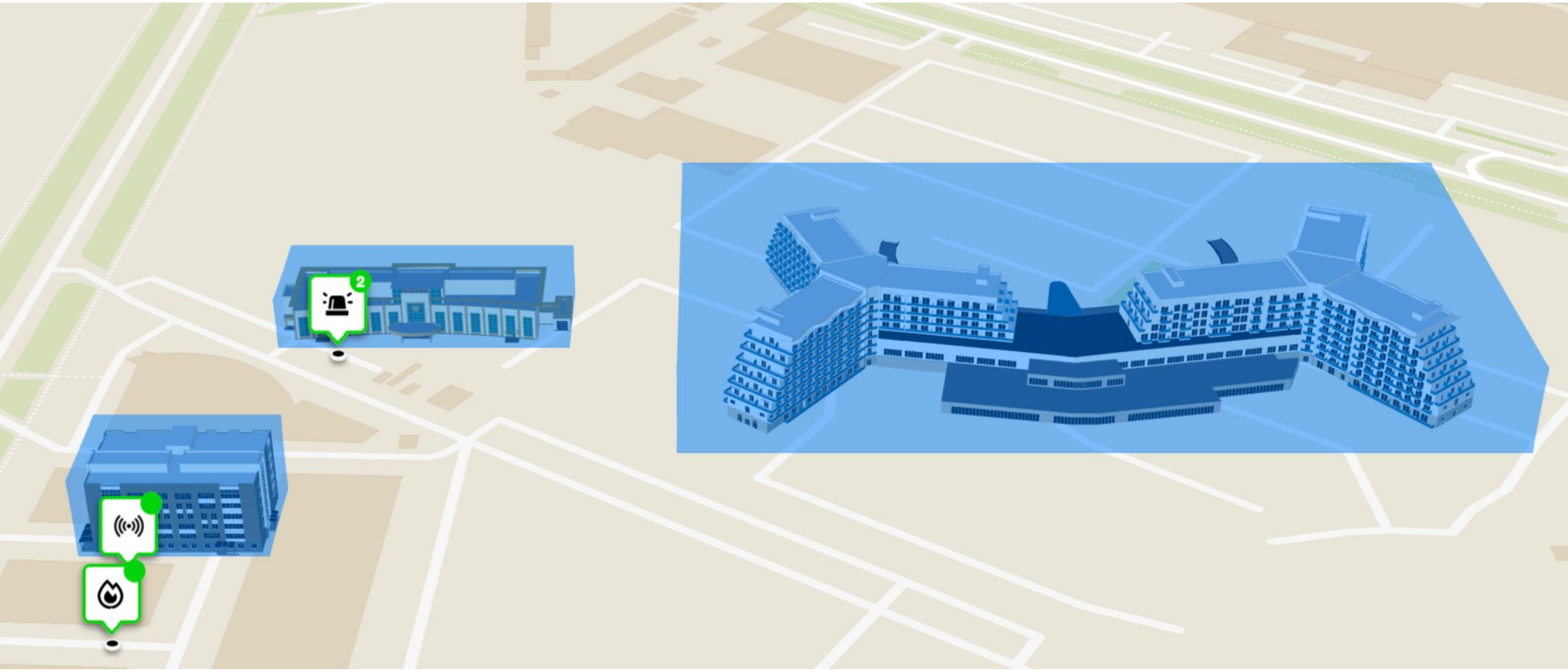
- получаем координаты карты из углов bbox модели

```
export const getCoordsFromVector = (vector: THREE.Vector3, map: Map) => {
    const x = Math.round((0.5 + vector.x / 2) * (map.getCanvas().width / window.devicePixelRatio));
    const y = Math.round((0.5 - vector.y / 2) * (map.getCanvas().height / window.devicePixelRatio));
    const mapCoordes = map.unproject({ x, y } as PointLike);
    return mapCoordes;
};
```

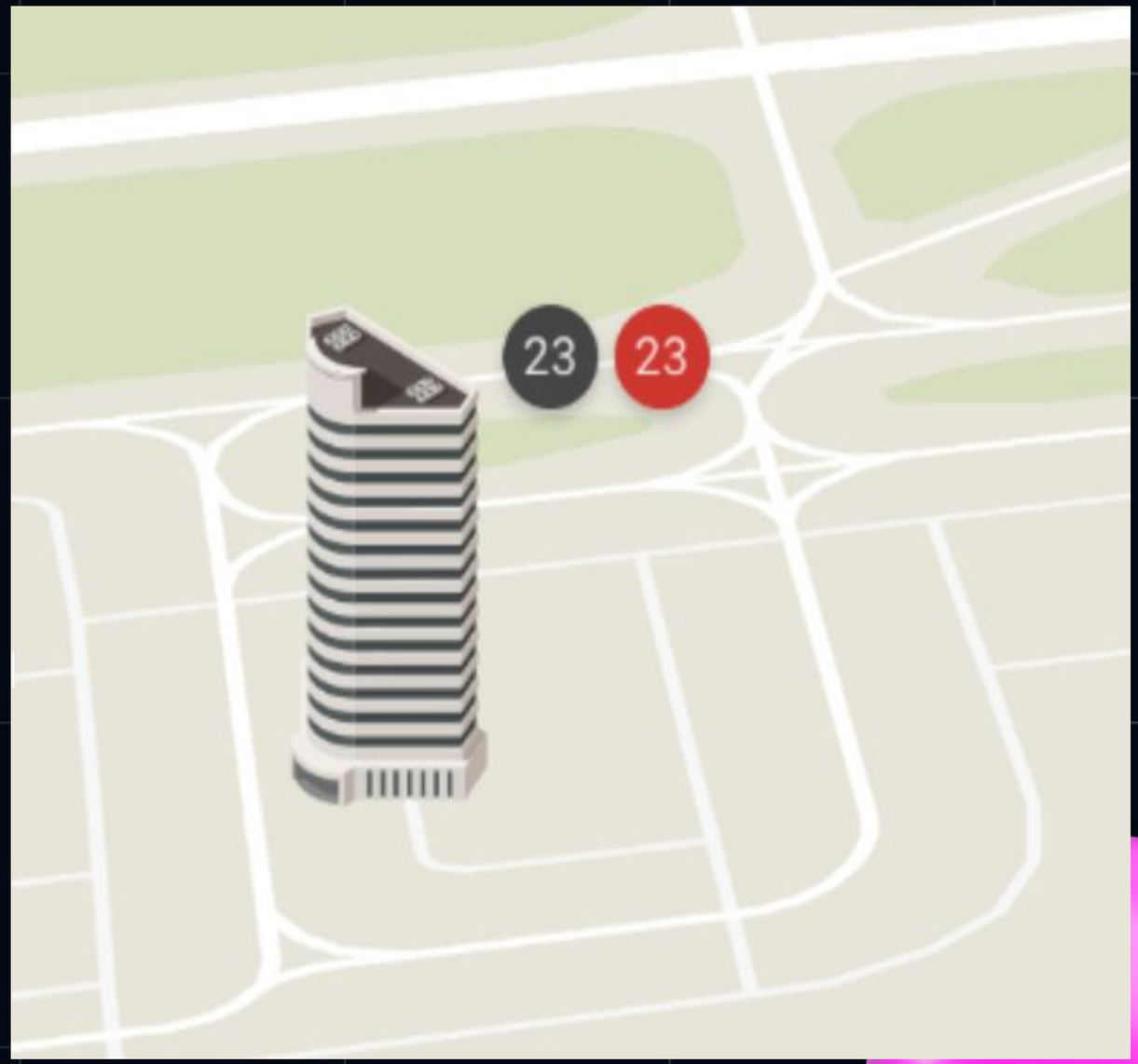
BBox

- используя turf.js строим полигон
- скрываем точки входящие в него


МОСКВА
19 с14
19 с24
19 с26А
19 с9
19 с7


2

Данный подход для отрисовки количества событий и инцидентов у здания.

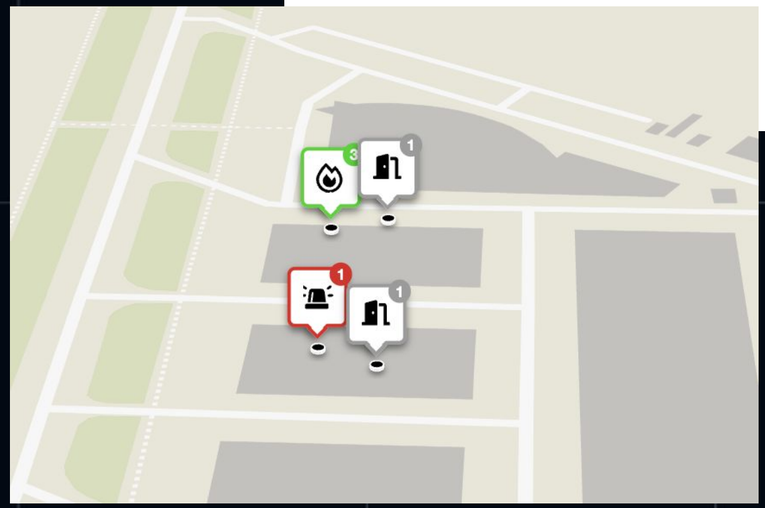


- tileserver-gl делаем прозрачный слой
- добавляем еще один инстанс карты
- синхронизируем 2 карты

```
{
  "id": "aeroway_fill",
  "type": "fill",
  "source": "openmaptiles",
  "source-layer": "aeroway",
  "minzoom": 4,
  "filter": ["==", "class", "aerodrome"],
  "paint": {
    "fill-color": "transparent",
    "fill-opacity": 0.7
  }
},
```

- выносим здания на отдельный слой с прозрачной картой

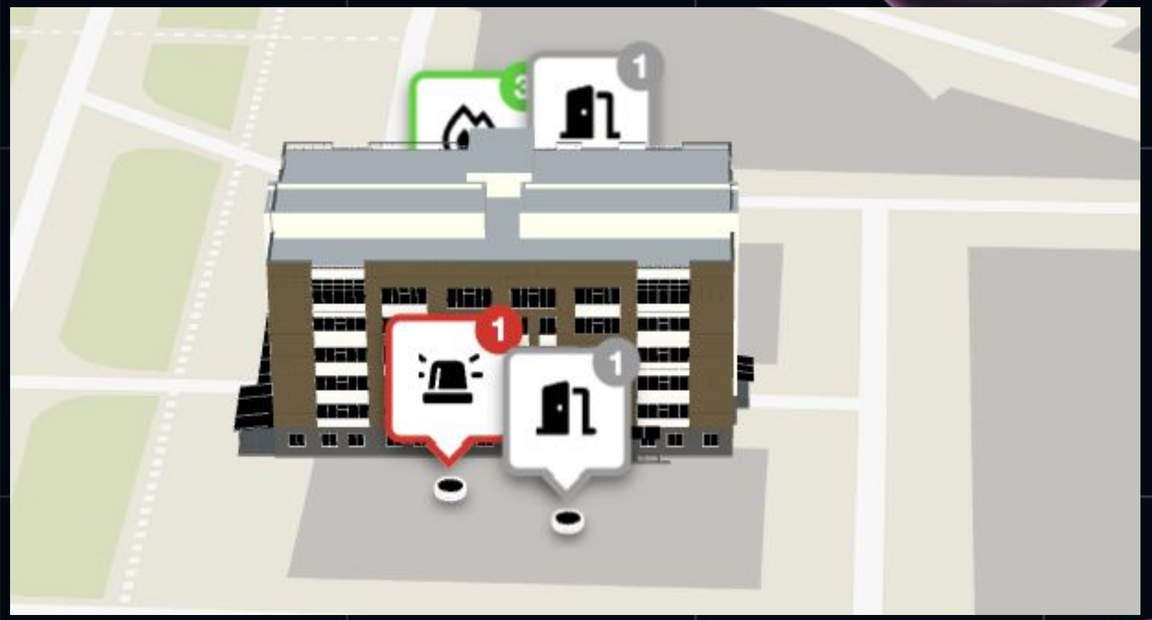
- отрисовываем его поверх карты с маркерами




3
1

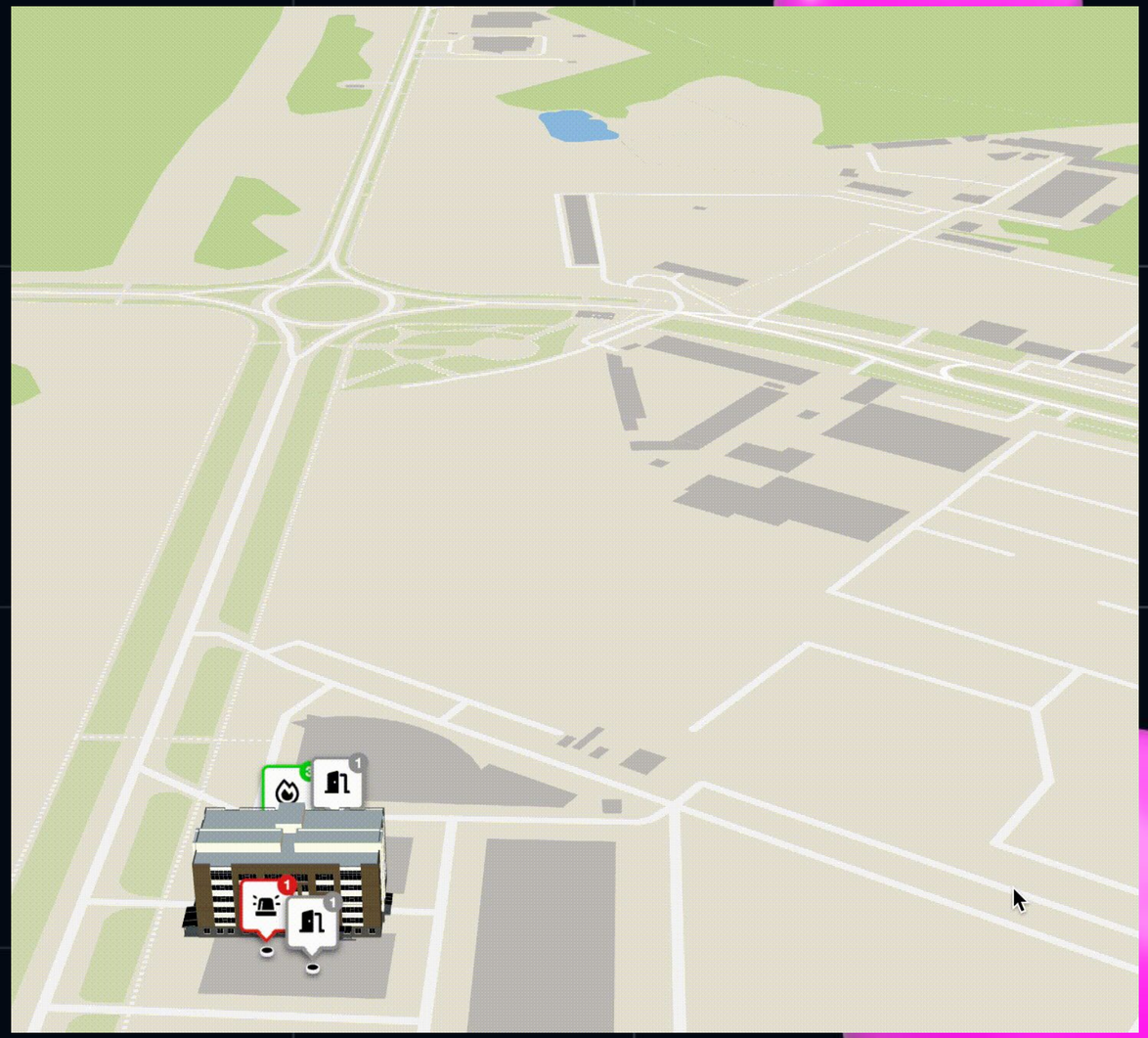
Решаем проблему добавляем прозрачный слой

- выбираем здания и точки
- raycast
- увеличиваем z-index



Взамодействие с маркерами

- при клике на карту если это не здание скрываем карту со зданиями
- по координатам определяем есть ли на них точка
- создаем MouseEvent на точке
- возвращаем карту со зданиями



Карты - шикарны

3D в браузере огонь

Но иногда больно

Не надо боятся непонятного

Спасибо!

@BooooBka