# Анализ оптимизации 3D-сцены

## Bruno Simon Portfolio 2025

## Обзор проекта

Интерактивное 3D-портфолио с открытым миром, построенное на Three.js с использованием WebGPU/WebGL рендеринга. Проект демонстрирует продвинутые техники оптимизации для достижения плавной производительности при использовании динамического освещения и теней в реальном времени.

## 1. Система материалов

### 1.1 Типы материалов

**Кастомный материал `MeshDefaultMaterial`**

- Базируется на `MeshLambertNodeMaterial`
- Использует Three.js Shading Language (TSL) для процедурной генерации
- Поддерживает как текстурированные, так и процедурные материалы

**Примеры использования:**

```
// С текстурой
const material = new MeshDefaultMaterial({
    colorNode: texture(paletteTexture).rgb,
    alphaNode: texture(alphaMap),
    hasCoreShadows: true,
    hasDropShadows: true
})

// Процедурный градиент
const material = new MeshDefaultMaterial({
    colorNode: mix(colorA, colorB, uv().y)
})
```

### 1.2 Текстуры в проекте

| Тип | Назначение | Оптимизация |
|---|---|---|
| `paletteTexture` | Цветовая палитра объектов | Одна текстура для множества объектов |
| `terrainTexture` | Данные ландшафта | Процедурная генерация + текстура |
| `floorSlabsTexture` | Детали пола | Тайлинг для экономии памяти |
| Perlin/Voronoi/Hash | Процедурные шумы | Переиспользуемые noise-текстуры |

| Тип | Назначение | Оптимизация |
|---|---|---|
| `DataTexture` | Следы колес | Динамическая генерация в runtime |

# 2. Система освещения и теней

## 2.1 Архитектура освещения

**Один источник света:**

- `DirectionalLight` с динамическими тенями
- Интенсивность: 5
- Динамическое движение (day/night cycle)

**Отсутствие запеченного освещения:**

- ❌ Нет lightmaps
- ❌ Нет ambient occlusion maps
- ☑ Все освещение вычисляется в реальном времени

## 2.2 Двухуровневая система теней

Ключевая оптимизация проекта - гибридный подход к теням:

**Core Shadows (Основные тени)**

Вычисляются в шейдере без shadow map:

```
// Дешевый расчет на основе нормали и направления света
coreShadowMix = normalWorld
    .dot(lightDirection)
    .smoothstep(coreShadowEdgeHigh, coreShadowEdgeLow)
```

**Преимущества:**

- Нулевая стоимость shadow map
- Мгновенный расчет
- Подходит для статичных объектов

**Drop Shadows (Отбрасываемые тени)**

Реальные тени из shadow map:

```
// Перехват теней из Three.js pipeline
receivedShadowNode = Fn(([ shadow ]) => {
    catchedShadow.mulAssign(shadow.r)
    return float(1)
})
```

**Применение:**

- Динамические объекты (машина, игрок)
- Важные визуальные элементы

**Комбинирование теней**

```
const combinedShadowMix = max(
    coreShadowMix,    // Дешевые тени
    dropShadowMix,    // Дорогие тени
    customShadowNode  // Дополнительные тени
).clamp(0, 1)

const shadowColor = baseColor.mul(shadowColorUniform)
outputColor = mix(outputColor, shadowColor, combinedShadowMix)
```

---

# 3. Адаптивное качество

## 3.1 Уровни качества

```
// Уровень 0 - Высокое качество
quality.level === 0:
    - shadowMapSize: 2048x2048
    - shadowRadius: 3
    - bloomMips: 5
    - DOF: включен

// Уровень 1 - Производительность
quality.level === 1:
    - shadowMapSize: 512x512
    - shadowRadius: 2
    - bloomMips: 2
    - DOF: отключен
```

## 3.2 Динамическая shadow camera

```
// Область теней привязана к видимой зоне
this.shadowAmplitude = this.game.view.optimalArea.radius
this.depth = this.game.view.optimalArea.radius * 2

// Камера следует за игроком
this.light.position
    .setFromSpherical(this.spherical)
    .add(optimalRoundedPosition)
```

**Результат:** Тени рендерятся только там, где они видны игроку.

---

# 4. Имитация глобального освещения

## 4.1 Fake Light Bounce

Вместо дорогого ray tracing - дешевая имитация отраженного света:

```
// Ориентация поверхности (смотрит ли вниз?)
const bounceOrientation = normal
    .dot(vec3(0, -1, 0))
    .smoothstep(lightBounceEdgeLow, lightBounceEdgeHigh)

// Расстояние от земли
const bounceDistance = lightBounceDistance
    .sub(max(0, positionWorld.y))
    .div(lightBounceDistance)
    .max(0)
    .pow(2)

// Цвет земли как источник отраженного света
const bounceColor = terrain.colorNode(terrainData)

// Смешивание
outputColor = mix(
    outputColor,
    bounceColor,
    bounceOrientation * bounceDistance * bounceMultiplier
)
```

**Эффект:** Объекты получают цветовой оттенок от поверхности под ними, создавая иллюзию глобального освещения.

---

# 5. Дополнительные оптимизации

## 5.1 Instancing

```
// Повторяющиеся объекты используют InstancedGroup
const instancedGroup = new InstancedGroup()
// Один draw call для множества объектов
```

## 5.2 Процедурная генерация

**Terrain:**

- Генерируется процедурно в шейдере

- Экономия памяти на геометрии
- Динамическое LOD

**Эффекты:**

- Следы колес через `DataTexture`
- Снег и вода процедурно
- Ветер через noise-функции

### 5.3 Селективные возможности материалов

```
const material = new MeshDefaultMaterial({
    hasCoreShadows: true,      // Включить основные тени
    hasDropShadows: false,     // Отключить отбрасываемые тени
    hasLightBounce: true,      // Включить отраженный свет
    hasFog: true,              // Включить туман
    hasWater: false            // Отключить эффект воды
})
```

Каждый материал может отключить ненужные эффекты для экономии производительности.

### 5.4 Post-processing

```
// Адаптивный post-processing
if (quality.level === 0) {
    // Высокое качество: DOF + Bloom
    postProcessing.outputNode = cheapDOFPass.add(bloomPass)
} else {
    // Производительность: только Bloom
    postProcessing.outputNode = scenePassColor.add(bloomPass)
}
```

---

## 6. Метрики производительности

### 6.1 Оптимизация shadow map

| Параметр | Высокое качество | Производительность | Экономия |
|---|---|---|---|
| Разрешение | 2048x2048 | 512x512 | **16x** памяти |
| Радиус размытия | 3 | 2 | **33%** расчетов |
| Bloom mips | 5 | 2 | **60%** текстур |

### 6.2 Техники экономии draw calls

- **Instancing**: Множество объектов → 1 draw call

- **Texture atlas**: Одна палитра для всех объектов
- **Процедурная геометрия**: Меньше mesh-объектов

---

# 7. Выводы и рекомендации

## 7.1 Ключевые техники

1. **Гибридные тени** - комбинация дешевых shader-based теней и реальных shadow maps
2. **Адаптивное качество** - автоматическое снижение нагрузки на слабых устройствах
3. **Fake GI** - имитация глобального освещения без ray tracing
4. **Процедурная генерация** - экономия памяти и гибкость
5. **Селективные эффекты** - каждый материал контролирует свои возможности

## 7.2 Применимость для других проектов

**Подходит для:**

- Открытых миров с динамическим освещением
- Проектов, требующих работы на широком спектре устройств
- Сцен с большим количеством объектов

**Требует адаптации для:**

- Интерьеров (лучше использовать lightmaps)
- Статичных сцен (запеченное освещение эффективнее)
- Фотореалистичных рендеров (нужен полноценный GI)

## 7.3 Рекомендации по внедрению

1. Начните с двухуровневой системы теней
2. Внедрите адаптивное качество с самого начала
3. Используйте процедурную генерацию где возможно
4. Создайте систему селективных эффектов для материалов
5. Оптимизируйте shadow camera под вашу сцену

---

# 8. Технический стек

- **Рендерер**: Three.js WebGPU/WebGL
- **Шейдеры**: Three.js Shading Language (TSL)
- **Оптимизация**: Custom материалы, instancing, процедурная генерация
- **Post-processing**: Bloom, DOF, custom passes

---

**Дата анализа:** 2026-02-14
**Версия проекта:** Bruno Simon Portfolio 2025
**Анализ выполнен:** Kiro AI Assistant