УДК 004.415+794.8

Адаптивная оптимизация Cascaded Shadow Maps с использованием OpenCL для динамических игровых сцен

Н.В. Хомичук[*1], С.А. Зори[*2]
[*1] магистрант, Донецкий национальный технический университет,
kristoll1995@gmail.com
[*2] д.т.н., доцент, Донецкий национальный технический университет,
ik.ivt.rec@mail.ru

Аннотация
В статье представлен адаптивный метод оптимизации Cascaded Shadow Maps (CSM) с использованием технологии OpenCL для повышения качества изображений динамических игровых сцен. Подход динамически регулирует параметры CSM, обеспечивая баланс между качеством и производительностью. Анализ результатов показывает существенное улучшение эффективности рендеринга теней и визуальной детализации. В дальнейших исследованиях планируется интеграция и всесторонняя оценка прототипа непосредственно в Unreal Engine 5, усовершенствование алгоритма адаптации параметров теней, а также проведение сравнительного анализа влияния разработанного решения на различные GPU.

Введение

В современных видеоиграх реалистичная графика и плавный игровой процесс являются ключевыми факторами погружения, делая оптимизацию рендеринга критически важной задачей [1, 2]. Особую сложность представляет рендеринг теней [3] в динамических сценах, где постоянные изменения освещения и геометрии требуют адаптивных решений.

Традиционные методы часто не справляются с поддержанием стабильной производительности и высокого качества изображения. В связи с этим, актуальной задачей является разработка эффективных алгоритмов рендеринга теней, способных динамически адаптироваться к изменяющимся условиям сцены.

Целью работы является разработка и оценка метода, позволяющего достичь оптимального баланса между качеством изображения и частотой кадров в динамических игровых окружениях. Предлагаемый метод направлен на решение проблемы нестабильной производительности и артефактов рендеринга, характерных для традиционных подходов, за счет динамической регулировки параметров CSM на основе анализа характеристик сцены и текущей производительности.

Анализ существующих подходов к оптимизации CSM в контексте OpenCL

Для определения наиболее перспективных направлений разработки адаптивного алгоритма рендеринга теней для динамических игровых сцен, проведён тщательный анализ существующих методов оптимизации Cascaded Shadow Maps (CSM) [4] в контексте OpenCL [5]. Ключевая цель анализа – выявление достоинств, недостатков и ограничений существующих решений для определения пробелов в текущих подходах и формулировки требований к новому адаптивному алгоритму.

Методы фильтрации, такие как Percentage Closer Filtering (PCF) [6], Variance Shadow Maps (VSM) [7] и Exponential Shadow Maps (ESM) [8], широко используются для уменьшения артефактов, связанных с алиасингом и дискретизацией карт теней [9]. PCF реализует сглаживание границ теней путём усреднения значений глубины соседних пикселей в карте теней. VSM использует дисперсию глубины для аппроксимации затенения, создавая более мягкие тени. ESM применяет экспоненциальное преобразование глубины для уменьшения артефактов самозатенения [9].

Однако, увеличение количества выборок при использовании PCF напрямую влияет на производительность. VSM и ESM требуют аккуратной настройки параметров для предотвращения размытия или появления артефактов на границах теней. OpenCL-реализация эффективно распараллеливает процесс фильтрации на GPU, значительно снижая нагрузку на CPU.

Техники направлены на динамическое изменение разрешения карт теней для каждого каскада в зависимости от плотности геометрии и близости к камере. Идея – выделение большего разрешения областям сцены, находящимся ближе к камере или содержащим больше деталей, и использование меньшего разрешения для удаленных областей. К примеру, на рисунке 1 тени на расстоянии (A) имеют соответствующее разрешение, тогда как тени вблизи камеры (B) демонстрируют искажение перспективы [10].

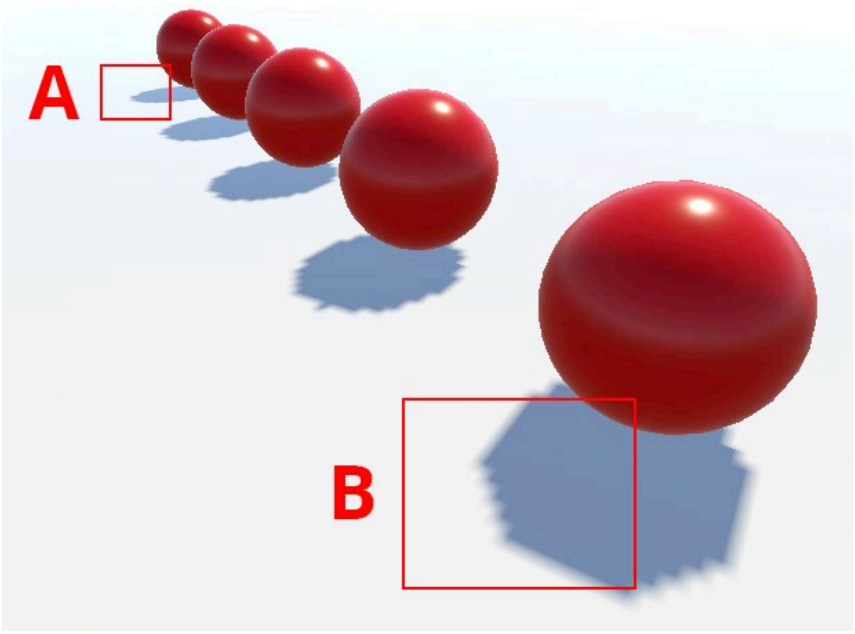

Рисунок 1 - Разрешение теней вблизи камеры и на расстоянии

Объекты, ближайшие к глазу, требуют более высокого разрешения, чем более удаленные объекты. Эффективность адаптивного разрешения каскадов зависит от точности оценки плотности геометрии и от алгоритма распределения разрешения [10]. Если настроить теневые каскады можно использовать 0, 2 или 4 каскада. Чем больше каскадов используется, тем меньше на тени влияет сглаживание перспективы [10]. Некорректная оценка приводит к неоптимальному распределению ресурсов и ухудшению качества теней. OpenCL-реализация эффективно распараллеливает процесс оценки плотности геометрии и динамически изменяет разрешение карт теней на GPU. Наложение перспективы менее заметно при использовании мягких теней и при использовании более высокого разрешения для карты теней. Однако эти решения используют больше памяти и пропускной способности при рендеринге [10].

Методы направлены на динамическую корректировку границ между каскадами CSM для точного распределения разрешения карт теней и минимизации артефактов на границах. Резкие переходы между каскадами с разным разрешением приводят к видимым артефактам. Динамическая корректировка границ смягчает эти переходы и улучшает качество теней [10]. На рисунке 2 представлена динамическая корректировка границ каскадов CSM.

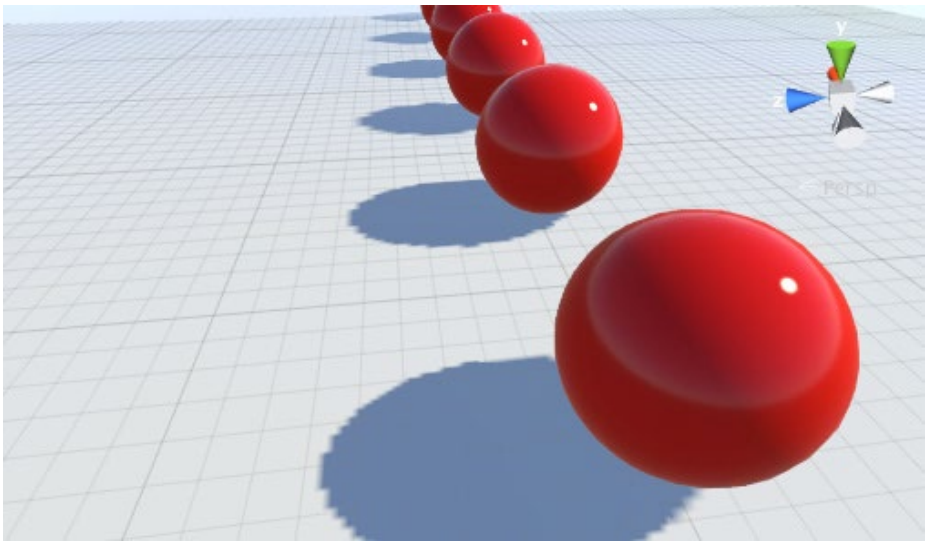

Рисунок 2 - Корректировка границ каскадов

Реализация динамической корректировки границ каскадов требует разработки эффективных алгоритмов определения оптимальных границ, учитывающих распределение геометрии и освещения в сцене [10]. Использование OpenCL позволяет реализовать такие алгоритмы на GPU, снижая нагрузку на CPU и обеспечивая плавную адаптацию границ каскадов к изменяющимся условиям сцены.

Разработка прототипа адаптивного алгоритма CSM на основе OpenCL

На основе анализа существующих подходов и выявленных ограничений, разработан прототип нового адаптивного алгоритма Cascaded Shadow Maps (CSM) для рендеринга теней в динамических игровых сценах с использованием OpenCL. Алгоритм направлен на достижение оптимального баланса между качеством изображения и производительностью за счёт динамической адаптации параметров CSM к изменяющимся условиям сцены (рис. 3 - 6).

Алгоритм начинается с анализа характеристик сцены, включающего оценку плотности геометрии, освещённости и текущей загрузки GPU. На основе этой информации определяется оптимальное количество каскадов, разрешение карт теней для каждого каскада и границы каскадов, стремясь к минимизации артефактов и обеспечению плавного перехода между уровнями детализации. Ключевой особенностью алгоритма является механизм динамической регулировки параметров CSM, основанный на обратной связи между характеристиками сцены, производительностью и качеством изображения.

Алгоритм измеряет частоту кадров (FPS), время рендеринга карт теней и загрузку GPU, а затем использует эти метрики для принятия решений об адаптации параметров CSM:

1) если FPS падает ниже заданного порога, уменьшается разрешение карт теней и/или количество каскадов;

2) если загрузка GPU низкая, увеличивается разрешение карт теней и/или количество каскадов;

3) если наблюдаются артефакты на границах каскадов, корректируются границы каскадов.

Прототип алгоритма реализован с использованием OpenCL для эффективного использования ресурсов GPU. Он включает ядра OpenCL для анализа сцены, рендеринга карт теней, фильтрации карт теней и рендеринга сцены с тенями.

Для минимизации задержек, связанных с передачей данных между CPU и GPU, используется эффективное управление памятью, включающее использование локальной и глобальной памяти GPU, а также минимизацию копирования данных между CPU и GPU с применением техник zero-copy, где это возможно.

```
#include <iostream>
#include <vector>

// Предполагаем, что у нас есть структуры данных для представления сцены, освещения, камеры и т.д.
struct Scene { /* ... */ };
struct Light { /* ... */ };
struct Camera { /* ... */ };
struct ShadowMap { /* ... */ };
struct Cascade { /* ... */ }; // Каскад содержит параметры камеры, разрешение карты теней и пр.

// И структуры для представления OpenCL context и command queue
struct CLContext { /* ... */ };
struct CLCommandQueue { /* ... */ };

// Предполагаем наличие OpenCL функций для:
// - Компиляции и запуска ядер
// - Чтения и записи данных между CPU и GPU
// - Синхронизации
extern void cl_compile_kernel(CLContext& context, const char* kernel_source, const char* kernel_name, cl_kernel& kernel);
extern void cl_launch_kernel(CLCommandQueue& queue, cl_kernel& kernel, size_t global_work_size, size_t local_work_size, /* ... */);
extern void cl_read_buffer(CLCommandQueue& queue, /* ... */);
extern void cl_write_buffer(CLCommandQueue& queue, /* ... */);

// Функции для определения характеристик сцены
float calculate_geometry_density(const Scene& scene, const Cascade& cascade);
float calculate_light_intensity(const Scene& scene, const Cascade& cascade, const Light& light);

// Функция для измерения загрузки GPU (требует специфичной для GPU реализации)
float measure_gpu_load();
float measure_fps();

// Функция для корректировки границ каскадов (нужен алгоритм определения оптимальных границ)
void adjust_cascade_boundaries(std::vector<Cascade>& cascades, const Scene& scene);
```

Рисунок 3 - Прототип адаптивного алгоритма CSM (часть 1)

```
int main() {
    // 1. Инициализация
    CLContext context;
    CLCommandQueue queue;
    cl_compile_kernel(context, "analysis_kernel.cl", "analysisKernel", analysisKernel);
    cl_compile_kernel(context, "shadow_map_kernel.cl", "shadowMapKernel", shadowMapKernel);
    cl_compile_kernel(context, "filter_kernel.cl", "filterKernel", filterKernel);
    cl_compile_kernel(context, "scene_kernel.cl", "sceneKernel", sceneKernel);

    Scene scene;
    Light light;
    Camera camera;

    int numCascades = 3;  // Начальное количество каскадов
    std::vector<Cascade> cascades(numCascades);

    float targetFPS = 60.0f;
    float minResolution = 256.0f;
    float maxResolution = 2048.0f;
    float lowGPULoadThreshold = 0.7f; // 70%

    // 2. Основной цикл рендеринга
    while (true) {
        // 2.1 Анализ характеристик сцены (на CPU, передаём на GPU для параллельной обработки)
        std::vector<float> geometryDensity(numCascades);
        std::vector<float> lightIntensity(numCascades);

        for (int i = 0; i < numCascades; ++i) {
          geometryDensity[i] = calculate_geometry_density(scene, cascades[i]);
          lightIntensity[i] = calculate_light_intensity(scene, cascades[i], light);
        }
```

Рисунок 4 - Прототип адаптивного алгоритма CSM (часть 2)

```
        //  2.2 Получение загрузки GPU (зависит от платформы и API)
        float gpuLoad = measure_gpu_load();
        float currentFPS = measure_fps();

        // 2.3 Определение оптимальных параметров CSM
        if (currentFPS < targetFPS) {
            // Производительность низкая, уменьшаем параметры
            if (cascades[0].shadowMapResolution > minResolution) {
                for (int i = 0; i < numCascades; ++i) {
                    cascades[i].shadowMapResolution /= 2.0f;
                }
            } else if (numCascades > 1) {
                numCascades--;
                cascades.resize(numCascades);
            }
        } else if (gpuLoad < lowGPULoadThreshold) {
            // GPU не загружен, увеличиваем параметры
            if (cascades[0].shadowMapResolution < maxResolution) {
                for (int i = 0; i < numCascades; ++i) {
                    cascades[i].shadowMapResolution *= 2.0f;
                }
            } else if (numCascades < 4) { // Предположим максимум 4 каскада
                numCascades++;
                cascades.resize(numCascades);
            }
        }
```

Рисунок 5 - Прототип адаптивного алгоритма CSM (часть 3)

```
    // 2.4 Корректировка границ каскадов (минимизация артефактов)
    adjust_cascade_boundaries(cascades, scene);

    // 2.5 Рендеринг карт теней (выполняется на GPU с помощью shadowMapKernel)
    std::vector<ShadowMap> shadowMaps(numCascades);
    for (int i = 0; i < numCascades; ++i) {
        // Установить параметры камеры для данного каскада
        // cl_launch_kernel(queue, shadowMapKernel, ..., scene, cascades[i], /* результаты -> shadowMaps[i]*/);
         std::cout << "Рендеринг карты теней для каскада " << i << std::endl;
    }

    // 2.6 Фильтрация карт теней (выполняется на GPU с помощью filterKernel)
     std::vector<ShadowMap> filteredShadowMaps(numCascades);
    for (int i = 0; i < numCascades; ++i) {
        // Установить параметры фильтрации для данного каскада
         //cl_launch_kernel(queue, filterKernel, ..., shadowMaps[i], /* результаты -> filteredShadowMaps[i]*/);
          std::cout << "Фильтрация карты теней для каскада " << i << std::endl;
    }

    // 2.7 Рендеринг сцены с тенями (выполняется на GPU с помощью sceneKernel)
     //cl_launch_kernel(queue, sceneKernel, ..., scene, camera, light, filteredShadowMaps, /* результаты -> frameBuffer*/);
    std::cout << "Рендеринг сцены" << std::endl;

    // 2.8 Отображение изображения на экране (зависит от движка и платформы)
    //display_framebuffer(frameBuffer);
    std::cout << "Отображение изображения" << std::endl;
}

return 0;
```

Рисунок 6 - Прототип адаптивного алгоритма CSM (часть 4)

Асинхронные операции позволяют перекрывать время ожидания на передачу данных и выполнение вычислений, дополнительно повышая производительность.

Блок-схема разработанного прототипа адаптивного алгоритма CSM на основе OpenCL представлена на рисунке 7, визуализируя последовательность этапов и взаимосвязи между ними.

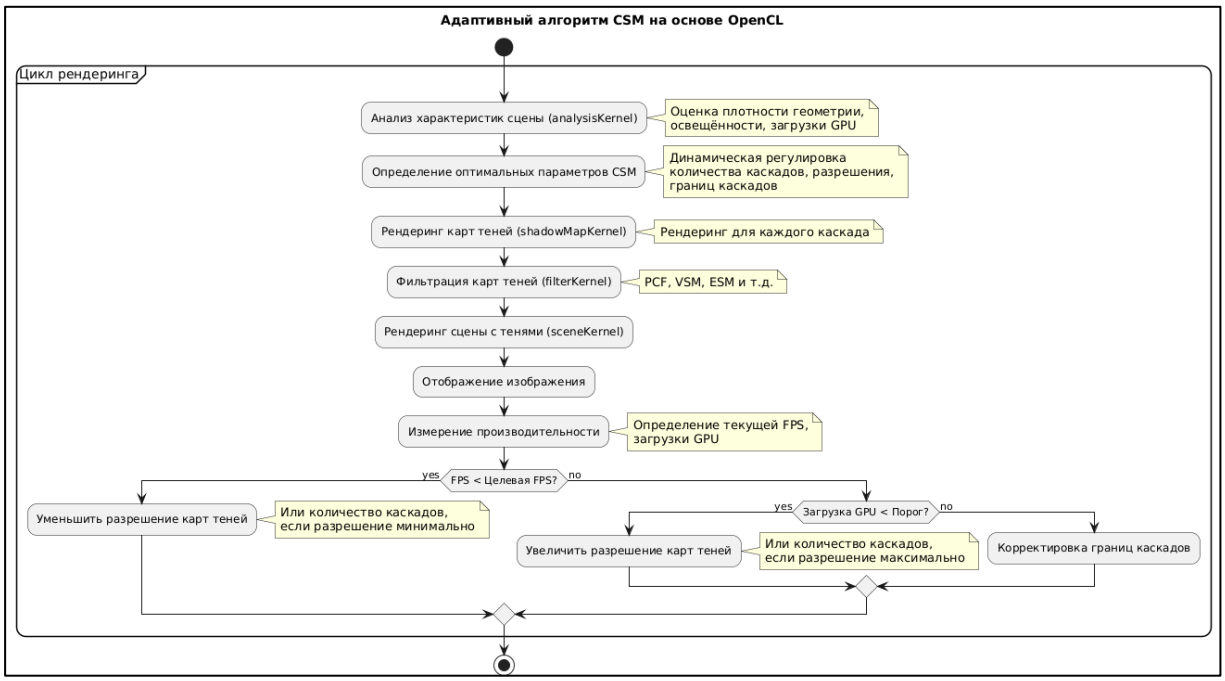

Рисунок 7 - Блок-схема разработанного прототипа адаптивного алгоритма CSM

Прогнозирование результатов внедрения прототипа в Unreal Engine 5

Внедрение разработанного адаптивного алгоритма Cascaded Shadow Maps (CSM) на основе OpenCL в Unreal Engine 5 (UE5) представляет собой перспективное направление для улучшения производительности рендеринга теней в динамических игровых сценах. Прогнозируется, что интеграция прототипа приведёт к ряду положительных результатов, связанных как с производительностью, так и с качеством изображения:

1. Снижение нагрузки на CPU. Перенос значительной части вычислений, связанных с

рендерингом теней (анализ сцены, рендеринг карт теней, фильтрация), на GPU с использованием OpenCL позволит снизить нагрузку на CPU. Это особенно важно для сложных сцен с большим количеством объектов и динамических источников света, где CPU может быть перегружен задачами, не связанными с рендерингом.

2. Улучшение частоты кадров (FPS). Сочетание снижения нагрузки на CPU и эффективного использования ресурсов GPU позволит добиться значительного улучшения частоты кадров в динамических игровых сценах. Это особенно важно для игр, требующих высокой частоты кадров для обеспечения плавного и отзывчивого игрового процесса.

3. Минимизация артефактов. Адаптивная регулировка параметров CSM (количества каскадов, разрешения карт теней, границ каскадов) позволит минимизировать артефакты, связанные с алиасингом, полосатостью и неправильным позиционированием теней.

4. Улучшение детализации теней. Использование более высокого разрешения карт теней в областях сцены с высокой детализацией позволит улучшить детализацию теней и обеспечить более реалистичное отображение мелких деталей.

5. Более плавные тени. Применение эффективных методов фильтрации карт теней (например, PCF, VSM, ESM) позволит сгладить границы теней и сделать их более плавными и реалистичными.

Заключение

Проведенное исследование позволило провести анализ возможностей OpenCL в контексте оптимизации рендеринга в игровых движках. Разработанный прототип с адаптивным алгоритмом рендеринга Cascaded Shadow Maps, оптимизированным с помощью OpenCL, представляет собой перспективное решение для повышения производительности и улучшения качества теней в динамических игровых сценах. Прогнозируется, что внедрение данного прототипа в современные игровые движки, включая Unreal Engine 5, позволит достичь увеличения скорости рендеринга, снижения нагрузки на GPU и, как следствие, более стабильной частоты кадров.

В целом, работа демонстрирует значительный потенциал OpenCL в контексте адаптивной оптимизации рендеринга теней и открывает новые возможности для создания более производительных и визуально привлекательных игр. В дальнейших исследованиях планируется интеграция и всесторонняя оценка прототипа непосредственно в Unreal Engine 5, дальнейшее усовершенствование алгоритма адаптации параметров теней, а также проведение сравнительного анализа влияния разработанного решения на различные GPU.

Литература

1. Зори, С. А. Использование средств аппаратной поддержки для повышения производительности систем 3D-пространственной визуализации / С. А. Зори, А. Я. Аноприенко, Р. В. Мальчева, О. А. Авксентьева // Информатика и кибернетика. - Донецк: ДонНТУ, 2019. - № 1 (15). - С. 5-12.

2. Хомичук, Н.В. Оптимизация производительности рендеринга в игровых движках с помощью технологии OpenCL / С. А. Зори, Н. В. Хомичук // Информатика и кибернетика. - Донецк: ДонНТУ, 2024. - № 4 (38). - С. 5-11.

3. Shadowing in 3D Graphics [Электронный ресурс] – Режим доступа: https://translated.turbopages.org/proxy_u/en-ru.ru.1f4386ae-680e64c9-23fbb5ac-74722d776562/https/www.tutorialspoint.com/computer_graphics/shadowing_in_3d_graphics.htm

4. Cascaded Shadow Maps // Learn [Электронный ресурс] – Режим доступа: https://learn.microsoft.com/en-us/windows/win32/dxtecharts/cascaded-shadow-maps OpenCL Architecture and AMD Accelerated Parallel Processing Technology [Электронный ресурс]. – Режим доступа: https://nov26.readthedocs.io/en/latest/Programming_Guides/Opencl-programming-guide.html#opencl-overview

5. OpenCL Architecture and AMD Accelerated Parallel Processing Technology [Электронный ресурс]. – Режим доступа: https://nov26.readthedocs.io/en/latest/Programming_Guides/Opencl-programming-guide.html#opencl-overview

6. Chapter 11. Shadow Map Antialiasing – 11.2 Percentage-Closer Filtering [Электронный ресурс] – Режим доступа: https://developer.nvidia.com/gpugems/gpugems/part-ii-lighting-and-shadows/chapter-11-shadow-map-antialiasing

7. Variance Shadow Maps (VSM) [Электронный ресурс] – Режим доступа: https://github.com/Delt06/toon-rp/wiki/Variance-Shadow-Maps-(VSM)

8. Тени / Майя // Руководство Вердж3Д [Электронный ресурс] – Режим доступа: https://www.soft8soft.com/docs/manual/ru/maya/Shadows.html

9. Learn OpenGL. Урок 5.3 — Карты теней // Хабр [Электронный ресурс] – Режим доступа: https://habr.com/ru/articles/353956/

10. Каскады теней // UnityHub [Электронный ресурс] – Режим доступа: https://unityhub.ru/manual/shadow-cascades

Хомичук Н.В., Зори С.А. Адаптивная оптимизация Cascaded Shadow Maps с использованием OpenCL для динамических игровых сцен. *В статье представлен адаптивный метод оптимизации Cascaded Shadow Maps (CSM) с OpenCL с использованием технологии OpenCL для повышения качества изображений динамических игровых сцен. Подход динамически регулирует параметры CSM, обеспечивая баланс между качеством и производительностью. Анализ результатов показывает существенное улучшение эффективности рендеринга теней и визуальной детализации. В дальнейших исследованиях планируется интеграция и всесторонняя оценка прототипа непосредственно в Unreal Engine 5, усовершенствование алгоритма адаптации параметров теней, а также проведение сравнительного анализа влияния разработанного решения на различные GPU.*

Ключевые слова: *OpenCL, Cascaded Shadow Maps, рендеринг, динамические сцены, адаптивные алгоритмы.*

Khomichuk N.V., Zori S.A. Adaptive optimization of Cascaded Shadow Maps using OpenCL for dynamic game scenes. *The article presents an adaptive optimization method for Cascaded Shadow Maps (CSM) with OpenCL for image quality enhancement of dynamic game scenes. The approach dynamically adjusts CSM parameters, ensuring a balance between quality and performance. The analysis of the results shows a significant improvement in the efficiency of shadow rendering and visual detail. In further research, it is planned to integrate and comprehensively evaluate the prototype directly into Unreal Engine 5, improve the algorithm for adapting shadow parameters, as well as conduct a comparative analysis of the impact of the developed solution on various GPUs.*

Key words: *OpenCL, Cascaded Shadow Maps, rendering, dynamic scenes, adaptive algorithms.*

Статья поступила в редакцию 20.01.2025
Рекомендована к публикации профессором Мальчевой Р. В.