Санкт-Петербургский государственный университет

Математическое обеспечение и администрирование информационных систем

Мосягин Олег Сергеевич

Реализация компьютерной игры

Отчет по учебной практике

Научный руководитель:
доцент кафедры системного программирования, к.т.н. Литвинов Юрий Викторович

Санкт-Петербург
2021

Оглавление

1. Введение

В настоящее время все большее значение играет компьютерная графика. Она используется для самых разных целей: от проектирования до развлечения. Компьютерные игры составляют значимую часть обширной области ее применения.

Часто для задач построения кадров создатели игр используют сторонние готовые графические движки [15]. Данный подход упрощает создание игр, однако, для некоторых целей гибкость, предоставляемая графическими движками, может оказаться недостаточной. Многие создатели игр разрабатывают собственные движки или пишут игру одновременно с системой рендеринга.

Для написания системы рендеринга необходимо знание графических API, предоставляющих доступ к мощностям видеокарт для задач компьютерной графики. По этой причине знания программных интерфейсов для работы с графикой сейчас особенно актуальны. Из основных графических API (OpenGL [20], DirectX [14], Vulkan [27], Metal [9]) был выбран Vulkan по причине его современности и поддержке на большинстве современных платформ.

Чтобы изучить Vulkan и получить опыт в задачах компьютерной графики было решено реализовать воксельную [22] игру про строительство. Этот формат игры был выбран по причине отсутствия надобности заниматься 3D моделированием, так как ландшафт можно генерировать процедурно.

1.1. Постановка задачи

1.1.1. Цель

Целью работы является изучение Vulkan API, путем создания компьютерной игры позволяющей делать постройки из кубических блоков (далее блоков).

1.1.2. Задачи

В рамках учебной практики для достижения поставленной цели были поставлены следующие задачи.

- Провести обзор наиболее популярной игры про строительство из блоков (Minecraft [16]).
- Разработать архитектуру приложения.
- Реализовать игру про строительство из кубов при помощи Vulkan.

2. Обзор

2.1. Обзор существующих решений

В этом разделе приведен анализ наиболее популярной игры про строительство из блоков (Minecraft) с целью изучения технических решений, использованных в ней. В данном разделе будут содержаться заключения по устройству Minecraft Java Edition 1.16.5, полученных в результате запуска игры в отладчике NVIDIA Nsight Graphics [10].

2.1.1. Текстуры

Большинство текстур в игре имеют размер 16 на 16 пикселей. Такой небольшой размер позволяет упаковать эти текстуры в атлас [21], чем и пользуются авторы игры. Так как сами текстуры, которые будут наложены на границы блоков, имеют такой небольшой размер, то для всего атласа создаются всего 5 уровней детализации [17] (низкие уровни детализации нужны для того, чтобы не возникал эффект муара [18] при наложении текстуры большого разрешения, по сравнению с областью экрана, которую занимает объект с данной текстурой).

2.1.2. Устройство мира игры

Для простоты управления открытым миром [19], он разделен на чанки (от англ. «chunk» — «кусок»), кластеры блоков 16 в длину, 16 в ширину и 256 в высоту. Они являются единицей загрузки и выгрузки частей мира. Рендер чанков происходит отдельно. При этом рисуются только те стороны блоков, которые не соприкасаются с непрозрачными блоками.

2.1.3. Сохранения

В сохранении на диск записываются файлы, содержащие информацию о целой области мира, состоящей из 32x32 чанков. С целью уменьшения размеров файлов данные сжимаются.

2.2. Использованные библотеки

2.2.1. Графическая библиотека

Для взаимодействия с видеокартой используетсяz Vulkan. Это современный и кроссплатформенный графический API. Его преимущество над более старыми интерфейсами (такими, как OpenGL) заключается в эффективном расходовании ресурсов процессора.

В Vulkan для вызова команд отрисовки нужно заполнить буфер команд необходимыми инструкциями и отправить его на выполнение. Существуют два вида командных буферов: первичный и вторичный. Только первичный командный буфер может быть отправлен на выполнение в какую-либо очередь видеокарты. Вторичный командный буфер может вызываться из первичного буфера при помощи команды vkCmdExecuteCommands [25]. Это удобно для составления первичных коммандных буферов, так как их можно составлять из вторичных коммандных буферов.

Для загрузки точек входа [4] Vulkan и его расширений используется Volk [12]. В Vulkan множество функций в качестве первого аргумента принимают дескриптор логического устройства (специальный объект, который является абстракцией видеокарты или группы видеокарт, с которыми будет осуществляется работа в данной функции). При выполнении таких функций обычно затрачивается время на диспетчеризацию вызова к конкретному логическому устройству. Для избежания накладных расходов в Vulkan предусмотрена функция vkGetDeviceProcAddr [26], позволяющая получить указатель на функцию для конкретного логического устройства. Помимо этого, функции, которые добавляются расширениями Vulkan, необходимо получать при помощи фунций, подобных vkGetDeviceProcAddr. Volk упрощает процесс загрузки точек входа Vulkan и его расширений, предоставляя указатели на загруженные функции.

2.2.2. Вспомогательные библиотеки

В решении используется Boost [1] для упрощения работы с многопоточностью, синтаксического анализа XML. Выбрана именно эта библиотека по причине наличия в ней широкого спектра нужных функций.

Для загрузки изображений используется stb [24]. Данная библиотека выбрана по причине простоты интерфейса и удобства добавления в проект. Помимо stb рассматривались CImg [2] и FreeImage [6]

Для создания окна выбрана GLFW [7], так как она поддерживает основные операционные системы и проста в использовании. Еще рассматривалась библиотека SDL [11].

В качестве математической библиотеки (для работы с такими объектами, как вектор и матрица) выбрана GLM [8] по той причине, что она поддерживает использование с Vulkan, имеет несложный интерфейс и был опыт использования данной библиотеки. Также рассматривалась библиотека Eigen [3].

Для генерации шума используется FastNoiseLite [5], так как данная библиотека удобно добавляется в проект и легка в использовании. Дополнительно рассматривались реализации шума в библиотеках stb (не подходит из-за небольшого периода функции шума) и FastNoise2 [**?**] (не подходит из-за сложного интерфейса).

Для сжатия данных используется библиотека zlib [23]. Она используется по причине кроссплатформенности, простоты использования и относительного удобства добавления в проект. Другие альтернативы не рассматривались, так как zlib — это стандарт де-факто для сжатия.

3. Реализация приложения

В процессе выполнения учебной практики было реализовано приложение на языке C++, позволяющее создавать постройки из блоков. Кадр из игры продемонстрирован на рисунке 1. Исходный код опубликован на GitHub[1].

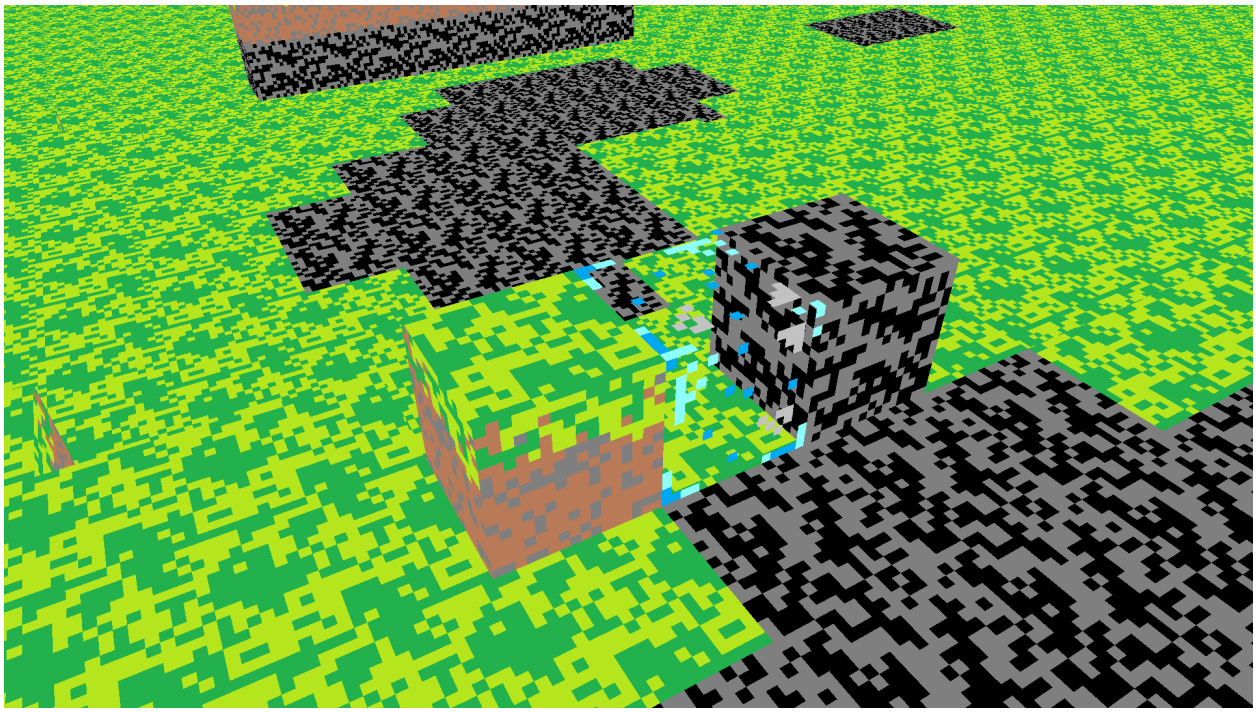

Рис. 1: Кадр из представленной игры.

3.1. Основные возможности приложения

Приложение позволяет летать по процедурно генерируемому миру. При этом отображаются только чанки, находящиеся рядом с пользователем. При перемещении происходит загрузка (или генерация, если данного чанка еще нет на диске) чанков, к которым приблизился игрок, и выгрузка на диск тех чанков, от которых отдалился. Пользователь может устанавливать блоки, выбирая их тип, и удалять любые блоки. Для установки нового блока нужно при помощи мыши навестись центром экрана на место, куда нужно поставить новый куб. При нажатии правой кнопки мыши блок будет установлен в дальнее пустое место,

[1] Исходный код: https://github.com/F5DXWsqPme/CubeGame-public (accessed: 17.05.2021)

которое пересекает луч из камеры через центр экрана до первого пересечения с каким-либо блоком. Нельзя установить блок так, чтобы он не соприкасался с существующими блоками. Тип устанавливаемого блока можно менять нажатиями на кнопки клавиатуры «Q» и «E». Для удаления произвольного куба нужно навестись центром экрана на нужный блок и нажать левую кнопку мыши. Операции с блоками производятся только в кубе размером 11x11x11 с центром в камере. На данный момент поддерживаются только три типа блоков: камень, трава и стекло.

3.2. Особенности реализации

Мир игры разделен на кластеры размером 16 в длину и ширину (чанки). Блоки могут быть установлены на высотах от 0 до 255. На отрисовку поступают только те стороны блоков, которые не граничат с непрозрачными блоками внутри чанка. Загрузка и выгрузка чанков осуществляется в параллельном потоке, обеспечивая плавную работу приложения.

Для каждого загруженного чанка создается свой вторичный командный буфер с инструкциями для отрисовки, который после этого вызывается из первичных буферов. Таким образом, при изменении геометрии чанка нужно перезаписывать только один вторичный командный буфер и первичные буферы.

Текстуры загружаются в виде предварительно созданного текстурного атласа. Он состоит из одного изображения, состоящего из текстур, и файла XML с описанием структуры изображения (в каком месте находится та или иная текстура).

При выгрузке чанков данные записываются на диск для сохранения построек пользователя. Так как зачастую данные содержат много повторов, они очень хорошо сжимаются словарными алгоритмами сжатия [13]. По этой причине на диск сохраняются сжатые данные.

3.3. Архитектура решения

В данном разделе приведено описание архитектуры реализованного приложения. На рисунке 2 представлена диаграмма компонентов проекта. Для простоты использования объектов Vulkan была реализована библиотека оберток над основными объектами данного программного интерфейса (vulkan_wrappers на схеме проекта). Основная цель данных оберток заключается в автоматическом освобождении подконтрольных объектов Vulkan. Вспомогательные классы содержатся в компоненте utils. Управлением игровой логикой занимается код из компонента game_objects. Система визуализации обозначена на диаграмме как render.

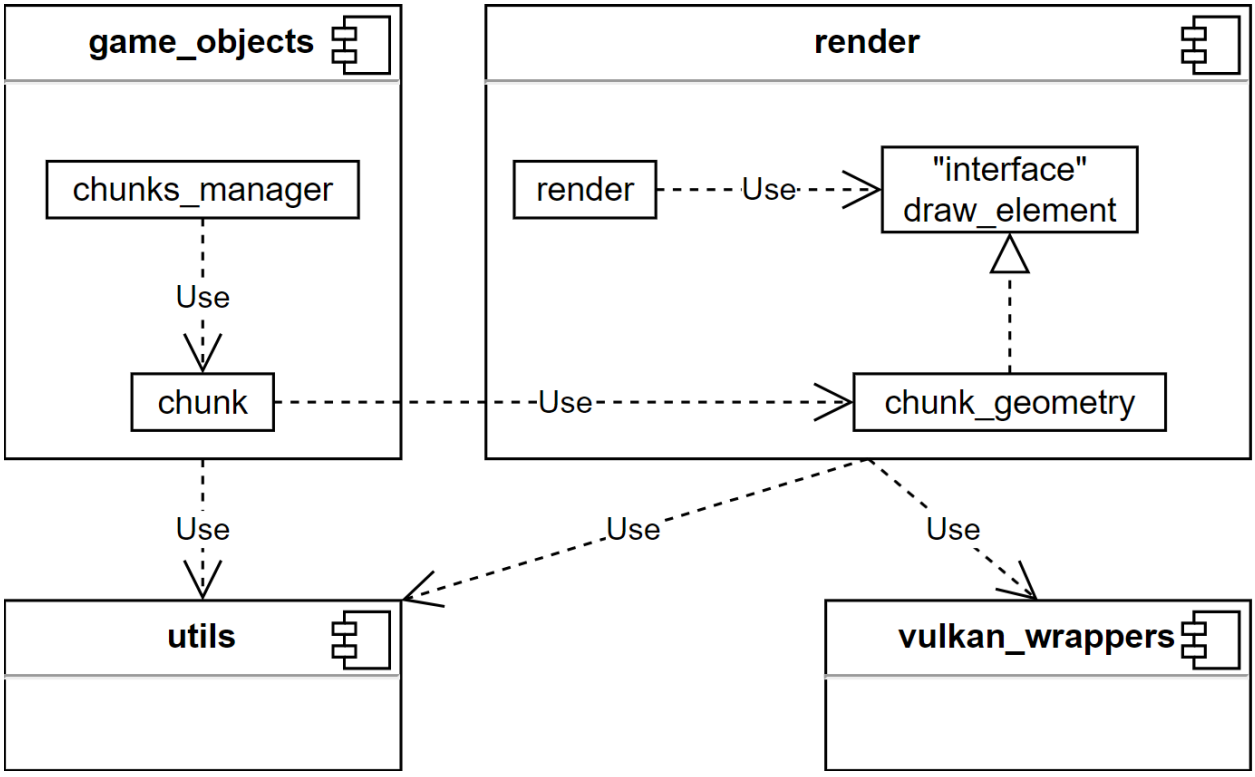

Рис. 2: Диаграмма компонентов.

3.4. Производительность

Для измерения производительности и сравнения с Minecraft была измерена частота кадров с одинаковой дальностью прорисовки (16 чанков). Было произведено 10 замеров.

- Minecraft.
  - Частота кадров: 89.3 Гц.
  - Среднеквадратичное отклонение: 2.9 Гц.
- Представляемое решение.
  - Частота кадров: 365.4 Гц.
  - Среднеквадратичное отклонение: 3.3 Гц.

4. Заключение

В итоге достигнуты следующие результаты.

- Проведен обзор наиболее популярной игры про строительство из блоков (Minecraft).
- Разработана архитектура приложения.
- Реализована игра, позволяющая пользователям создавать постройки из блоков.

В результате разработана игра для строительства построек из кубов.

Ссылка на исходный код — https://github.com/F5DXWsqPme/CubeGame-public (accessed: 17.05.2021)

Список литературы

[1] Boost C++ Libraries. — URL: https://www.boost.org/ (online; accessed: 26.05.2021).

[2] The CImg Library. — URL: https://cimg.eu/ (online; accessed: 02.06.2021).

[3] Eigen. — URL: https://eigen.tuxfamily.org/index.php?title=Main_Page (online; accessed: 02.06.2021).

[4] Entry Point. — URL: https://en.wikipedia.org/wiki/Entry_point (online; accessed: 02.06.2021).

[5] FastNoiseLite. — URL: https://github.com/Auburn/FastNoiseLite (online; accessed: 26.05.2021).

[6] The FreeImage Project. — URL: https://freeimage.sourceforge.io/ (online; accessed: 02.06.2021).

[7] GLFW. — URL: https://www.glfw.org/ (online; accessed: 26.05.2021).

[8] GLM. — URL: https://github.com/g-truc/glm (online; accessed: 26.05.2021).

[9] Metal. — URL: https://developer.apple.com/metal/ (online; accessed: 11.12.2020).

[10] NVIDIA Nsight Graphics. — URL: https://developer.nvidia.com/nsight-graphics (online; accessed: 15.05.2021).

[11] Simple DirectMedia Layer - Homepage. — URL: https://www.libsdl.org// (online; accessed: 02.06.2021).

[12] Volk. — URL: https://github.com/zeux/volk (online; accessed: 26.05.2021).

[13] Wikipedia. Dictionary coder // Wikipedia, the free encyclopedia. — URL: https://en.wikipedia.org/wiki/Dictionary_coder (online; accessed: 15.05.2021).

[14] Wikipedia. DirectX // Wikipedia, the free encyclopedia. — URL: https://ru.wikipedia.org/wiki/DirectX (online; accessed: 11.12.2020).

[15] Wikipedia. Game engine // Wikipedia, the free encyclopedia. — URL: https://en.wikipedia.org/wiki/Game_engine (online; accessed: 15.05.2021).

[16] Wikipedia. Minecraft // Wikipedia, the free encyclopedia. — URL: https://en.wikipedia.org/wiki/Minecraft (online; accessed: 15.05.2021).

[17] Wikipedia. Mipmap // Wikipedia, the free encyclopedia. — URL: https://en.wikipedia.org/wiki/Mipmap (online; accessed: 26.05.2021).

[18] Wikipedia. Moiré pattern // Wikipedia, the free encyclopedia. — URL: https://en.wikipedia.org/wiki/Moire_pattern (online; accessed: 26.05.2021).

[19] Wikipedia. Open world // Wikipedia, the free encyclopedia. — URL: https://en.wikipedia.org/wiki/Open_world (online; accessed: 15.05.2021).

[20] Wikipedia. OpenGL // Wikipedia, the free encyclopedia. — URL: https://en.wikipedia.org/wiki/OpenGL (online; accessed: 15.05.2021).

[21] Wikipedia. Texture atlas // Wikipedia, the free encyclopedia. — URL: https://en.wikipedia.org/wiki/Texture_atlas (online; accessed: 15.05.2021).

[22] Wikipedia. Voxel // Wikipedia, the free encyclopedia. — URL: https://en.wikipedia.org/wiki/Voxel (online; accessed: 15.05.2021).

[23] Wikipedia. zlib // Wikipedia, the free encyclopedia. — URL: https://en.wikipedia.org/wiki/Zlib (online; accessed: 15.05.2021).

[24] stb. — URL: https://github.com/nothings/stb (online; accessed: 26.05.2021).

[25] vkCmdExecuteCommands(3) Manual Page. — URL: https://www.khronos.org/registry/vulkan/specs/1.2-extensions/man/html/vkCmdExecuteCommands.html (online; accessed: 15.05.2021).

[26] vkGetDeviceProcAddr(3). — URL: https://www.khronos.org/registry/vulkan/specs/1.2-extensions/man/html/vkGetDeviceProcAddr.html (online; accessed: 03.06.2021).

[27] Г. Селлерс. Vulkan. Руководство разработчика. — ДМК Пресс, 2017. — ISBN: 978-1568814247.