План курсовой работы по теме "Продвинутый 3D renderer"

30.01.2022

Содержание

1 Аннотация

На сегодняшний день технологии отрисовки трёхмерных сцен используются во многих сферах нашей жизни: в 3d моделировании и анимации, в компьютерных играх и 3d/VR симуляциях. Первые статьи по данной теме были опубликованы ещё в 60х-70х годах 20 века, и с тех пор было проведено огромное количество исследований и работ в сфере 3d рендеринга.

Цель данной работы - изучить и реализовать ряд важнейших алгоритмов 3d рендеринга. На втором курсе я реализовал базовую версию 3d рендера, в которой все основные алгоритмы уже были имплементированы. Поэтому в этой курсовой работе планируется продолжить изучать данную сферу, и реализовать более сложные техники отрисовки.

2 Постановка задачи

2.1 Цели

Основная цель проекта - это изучение и реализация алгоритмов, использующихся в компьютерной графике, на которых основаны большинство программ 3d моделирования, 3d игр, 3d/VR симуляторов и других приложений, имеющих какое-либо отношение к трёхмерной графике.

2.2 Задачи

Основные задачи:

- Изучить и реализовать алгоритмы (ниже будет конкретный список)
- Изложить теоретические основы этих алгоритмов, описать детали их реализации, написать сопроводительную документацию к коду и протестировать его

2.3 Предварительный список техник, которые я планирую реализовать

1. Отображение нормалей (normal mapping)
2. "Параллакс" отображение (parallax mapping)
3. HDR
4. Bloom
5. Отложенное освещение и затенение (deferred shading) - в качестве эксперимента
6. SSAO
7. Физически корректный рендеринг (PBR)
8. Поддержка прозрачных объектов
9. Поддержка теней
10. Геометрические шейдеры (может быть)

11. Разные эффекты постобработки
12. Сглаживание (antialiasing)
13. Система частиц

2.4 Уже реализованные в прошлом году алгоритмы

1. Построение матрицы перехода из глобальной системы координат в пространство камеры
2. Построение матриц проекции на экран (перспективная и ортогональная проекции)
3. Удаление фрагментов треугольников, лежащих вне пирамиды зрения (view frustrum)
4. Проверка точек на глубину с помощью z-буффера (при отрисовке объектов должны отрисовываться только ближайшие объекты к камере, но не те, которые находятся за ними)
5. Отрисовка отрезков на экране (алгоритм Брезенхэма)
6. Отрисовка треугольников на экране
7. Перспективно правильная интерполяция параметров объектов при перспективной проекции

2.5 Репозиторий

Репозиторий проекта: [4]

3 Актуальность и значимость

Акутальности темы 3d рендеринга в целом я уже касался выше, поэтому давайте поговорим про актуальность конкретно моей работы.

Важной особенностью моей реализации является то, что в моём проекте не используются такие известные API для работы с компьютерной графикой как OpenGL, DirectX, Vulcan и др. Это означает, что весь рендеринг происходит исключительно на CPU (так называемый software rendering). Данный подход имеет ряд своих преимуществ и недостатков, по сравнению с распространенным GPU rendering-ом:

Преимущества:

- Работает на всех устройствах, независимо от наличия видеокарты. (в том числе микроконтроллерах и других встроенных системах)
- Также не нужно требовать от видеокарты пользователя поддержки конкретной версии API (например opengl 3.3)
- На всех устройствах одно и то же приложение работает одинаково, нет уязвимости к багам реализации API.

- С точки зрения программирования software renderer-а, есть полный контроль над реализацией. Это означает, что можно добавлять свои произвольные методы в пайплайн отрисовки, или изменять какие-то части пайплайна.

- С образовательной точки зрения, самостоятельная реализация 3d рендерера с нуля даёт намного более глубокое понимание работы алгоритмов 3d рендеринга, чем просто использование некоторой готовой библиотеки.

Недостатки:

- Главный недостаток - скорость. Видеокарты намного лучше справляются с множеством хорошо распараллеливаемых вычислений, чем процессоры. Они специально для этого и проектировались. Разница в скорости очень сильно зависит от конкретной видеокарты и конкретного приложения, но может составлять несколько порядков.

- С точки зрения программиста - нужно всё писать с нуля самому, а не использовать API (хотя также есть и уже написанные библиотеки).

Как можно видеть из данных особенностей, моё приложение вряд ли будет иметь шанс соперничать по скорости с библиотеками, использующими мощности видеокарт.

Тем не менее, в случаях, когда скорость не так важна, но необходима кроссплатформенность и отсутствие большого количества внешних зависимостей, я надеюсь, что мой 3d рендерер может быть полезен в качестве библиотеки отрисовки.

Также, данный проект возможно будет полезен для некоторых людей, так же как и я интересующихся 3d графикой, как пример реализации software 3d рендерера на более-менее современном стандарте c++17.

Ну и очевидно, работа также значима лично для меня, как возможность изучить многие алгоритмы 3d рендеринга и реализовать их на практике.

4 Существующие работы и решения

Не знаю, сколько точно существует работ по данной теме, но я уверен, что их количество измеряется в сотнях, если не в тысячах. Очевидно, что все аналоги я не смогу здесь привести и описать, но постараюсь хотя бы упомянуть самые популярные. Проекты я брал из списка на github [13].

1. ssloy/tinyrenderer [11] Отличный проект, весь код занимает примерно 500 строчек. Насколько я понимаю, реализовано не так много алгоритмов 3d рендеринга (например нет клиппинга). Вообще проект в основном носит образовательный характер, и в этом плане там очень хорошие уроки на вики странице.

2. zauonlok/renderer [12] Тоже интересный проект, написан на C89, реализовано довольно много вещей, но более продвинутые техники также не реализованы (которые я планирую реализовать).

3. kosua20/herebedragons [5] Не совсем корректно считать этот проект аналогом моего, так как это не software рендерер, а наоборот, реализация одной сцены с помощью разных графических API. Интересно то, что также есть реализации сцены на платформах, где никаких API для 3d графики нет (например для PICO-8 и Nintendo Game Boy

Advance). На таких платформах используются различные способы обойти аппаратные ограничения.

4. skywind3000/mini3d [8] Небольшой (700 строк) 3d рендерер на c. Почти нет никаких реализованных дополнительных/продвинутых алгоритмов.

5. ssloy/tinyraycaster [10] Как видно из названия, это рейкастер, а не полноценный 3d рендерер, поэтому тоже не очень корректный пример.

6. skywind3000/RenderHelp [9] Ещё один довольно простой 3d рендерер на C++.

7. Angelo1211/SoftwareRenderer [1] Один из самых интересных проектов из всего списка. Реализовано довольно много продвинутых алгоритмов. Тем не менее, есть несколько проблем с сглаживанием и "Муаровым эффектом" в некоторых сценах, а также есть нереализованные алгоритмы (из моего планируемого списка).

8. martinResearch/DEODR [7] Дифференцируемый 3d рендерер на c. Какая-то классная штука для ML. Не уверен, часто ли данную библиотеку используют в других областях.

Бонусные аналоги (не из списка выше, по крайней мере не из топа по звёздам)

1. bytecode77/fastpix3d [2] Пока сильно не разбирался как именно, но данный 3d рендерер очень хорошо оптимизирован, и действительно показывает хорошие показатели по производительности. Хотя в примерах используются не очень детализированные модели и текстуры в не очень высоком разрешении, производительность не может не впечатлять, учитывая, что это software рендерер. Каких-то очень продвинутых алгоритмов не реализовано, хотя есть поддержка освещения и теней.

2. Dawoodoz/DFPSR [3] Мне очень нравятся идеи, применённые в данном проекте. Библиотека предназначена для изометрических игр/сцен, и если использовать тот факт, что объекты всегда будут видны только с одного ракурса, то можно "запечь"(pre-render, bake) 3d модель в три текстуры - diffuse, normal, height, а дальше работать с моделью как с одним прямоугольником (причём ориентированным в пространстве камеры вдоль координатных осей). Это очень сильно снижает необходимое количество вычислений на процессоре и позволяет отрисовывать достаточно сложные изометрические сцены в реальном времени.

Как можно видеть, довольно мало проектов реализуют какие-то продвинутые алгоритмы 3d рендеринга, либо же я плохо искал. Это немного обнадёживает в плане того, что работа кажется довольно актуальной (так как аналогов не так много).

5 Предлагаемые подходы и методы

В основном используются достаточно классические алгоритмы, описанные во многих книгах и статьях, например [15], [14]. Некоторые алгоритмы, которые уже были реализованы в прошлом году, я указал в секции 2.4.

Я бы не сказал, что подходы, которые я использую, обладают большой новизной, но разумеется реализации даже одно и того же алгоритма разными людьми может часто выглядеть довольно по разному. Также, я стараюсь писать в более-менее современном стиле c++17, и каких-то очень похожих реализаций 3d рендерера я пока что не видел.

6 Ожидаемые результаты

Кроссплатформенное интерактивное приложение, в котором можно переключаться между сценами, редактировать сцены и отдельные объекты, менять различные параметры освещения и наглядно наблюдать разные алгоритмы отрисовки 3d моделей.

Также есть отдельное "ядро" 3d renderer-a, используя которое в качестве библиотеки можно написать свою произвольную программу, для которой необходима отрисовка 3d сцен.

Ещё ожидается, что весь код будет протестирован, задокументирован и содержать комментарии.

7 План работ

Примерный план:

- 30.01.2022 - план КР, normal mapping, parallax mapping, небольшие оптимизации 3d renderer-a (уже сделано)
- 01.03.2022 - HDR, Bloom, SSAO, PBR
- 01.04.2022 - Прозрачные объекты, постобработка, сглаживание, система частиц
- 01.05.2022 - Тени, отложенный рендеринг, может быть что-то ещё
- По мере работы - Оптимизация производительности и улучшение качества кода

8 Список источников

- learnopengl.com [6]
- Mathmatics for 3d game programming and computer graphics [15]
- A Mathematical Introduction with OpenGL [14]

[1] Angelo1211/softwarerenderer. URL: https://github.com/Angelo1211/SoftwareRenderer.

[2] bytecode77/fastpix3d. URL: https://github.com/bytecode77/fastpix3d.

[3] Dawoodoz/dfpsr. URL: https://github.com/Dawoodoz/DFPSR.

[4] github репозиторий проекта. URL: https://github.com/asmorodinov/3d-renderer-from-scratch/tree/dev.

[5] kosua20/herebedragons. URL: https://github.com/kosua20/herebedragons.

[6] learnopengl.com. URL: https://learnopengl.com/.

[7] martinresearch/deodr. URL: https://github.com/martinResearch/DEODR.

[8] skywind3000/mini3d. URL: https://github.com/skywind3000/mini3d.

[9] skywind3000/renderhelp. URL: https://github.com/skywind3000/RenderHelp.

[10] ssloy/tinyraycaster. URL: https://github.com/ssloy/tinyraycaster.

[11] ssloy/tinyrenderer. URL: https://github.com/ssloy/tinyrenderer.

[12] zauonlok/renderer. URL: https://github.com/zauonlok/renderer.

[13] Список software 3d renderer-ов на github. URL: https://github.com/topics/software-rendering.

[14] Samuel R. Buss. A Mathematical Introduction with OpenGL. Cambridge University Press, 2003.

[15] Eric Lengyel. Mathmatics for 3d game programming and computer graphics. Course Technology PTR, 2012.