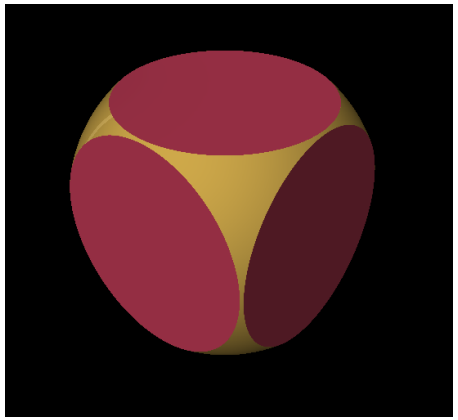
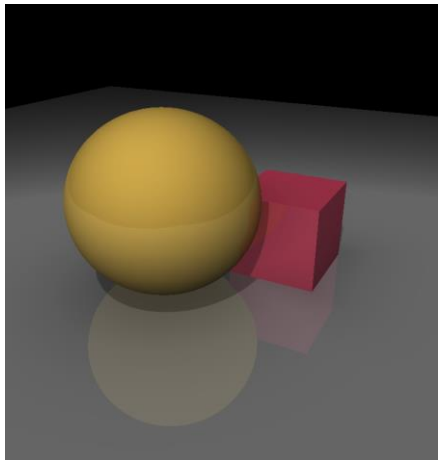
Разработка системы визуализации трехмерных сцен с помощью алгоритма трассировки лучей

ВШЭ СПб ПМИ, весна 2023





Команда:

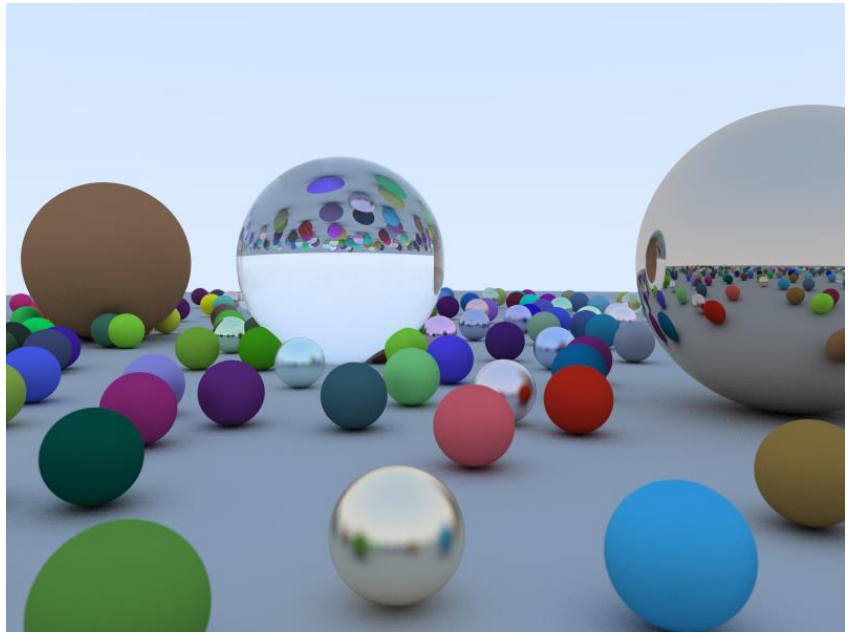
- Кураленок Святослав
- Власов Дмитрий

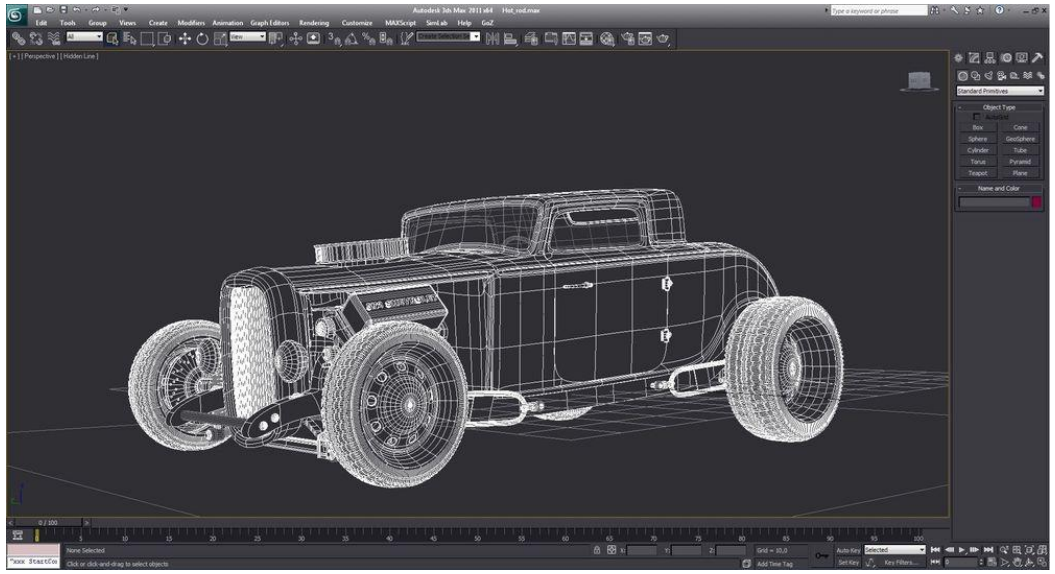
Ментор:

- Александр Викторович Еналдиев

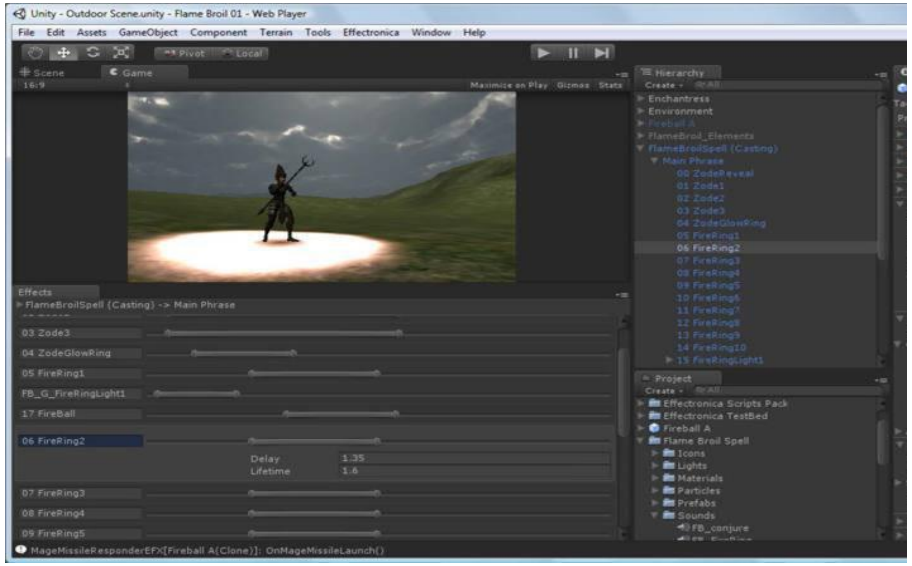
Обзор области:

Компьютерная графика - это область компьютерных наук, которая занимается отображением изображений с помощью компьютера. Она включает в себя 2D и 3D графику, анимацию, работу с текстом и эффектами, разработку игр и веб-дизайн.



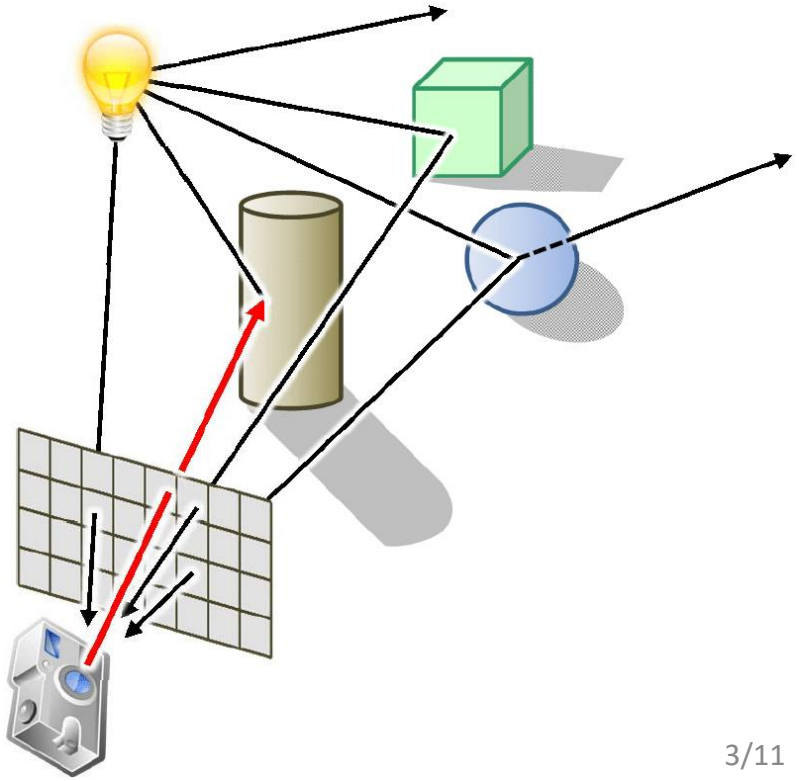


Описание проекта:



Графический движок — программное обеспечение, которое предоставляет инструменты для создания и отрисовки двухмерных или трехмерных сцен.

Одним из способов отрисовки сцен является **Raymarching** - способ, в котором цвет каждого пикселя определяется посредством запуска луча и анализа его взаимодействия с поверхностями объектов сцены.



Сравнение с аналогами:

	Наш проект	Unity	Unreal Engine
Отрисовка raymarching-ом	Да	Нет	Нет
Open source	Да	Нет	Нет
Простой пользовательский интерфейс	Да	Да	Нет
Большой набор инструментов	Нет	Да	Да

Используемые технологии:

Графическая библеотека:

OpenGL + glsl (язык программирования шейдеров – програм, исполняющихся на видеокарте). Почему не DirectX/Metal/Vulcan? Одна из осных целей нашего проекта – написание движка для работы с визуалом на всех основных платформах (Apple/Windows/Linux), OpenGL – старая библеотека, которая работает везде, DirectX – только на Windows, Metal – только на Apple, Vulcan – работает там где надо, но очень навороченный, в реалях нашей задачи и времени OpenGL подошел лучше.

Библеотека для работы с окном (его создание + обработка колбэков):

GLFW. Почему именно GLFW, а не например GLUT/GLAD/etc.? Вся работа с окном в нашем проекте: обработка колбэков по изменению его размеров и ввод (клавиатура + мышь) – такой функционал предоставляет абсолютно любая библеотека, так что взяли то с чем уже работали.

Архитектура проекта: структура

Клиент-программист создает сцену со следующими параметрами:

- В методе создания сцены: задает сцену
- В методе обновления сцены: задает поведение каждой фигуры и их взаимодействие с друг-другом

Движок:

- Предоставляет пользователю весь функционал вида: create, set, get + управление внутренностями всех созданных объектов
- Создает все заданные сцены и выполняет все инструкции, которые пользователь задал в методе обновления сцены и рисует все видимые объекты (примитивы/модели)
- Выполняет вышеуказанные пункт для всех созданных сцен (обычно работаем с одной сценой)

Архитектура проекта: дерево классов

Рендер: класс который создает окно, обрабатывает все колбэки, считает глобальные переменные и запускает обработку всех созданных сцен

Сцена: класс который является связующей точкой между движком и пользователем программистом, предоставляет весь функционал по взаимодействию с ресурсами рендера (создание, хранение, отрисовка, удаление)

Утилиты:

- **Камера**: матиматический объект, относительно которого происходит весь рендер
- **Вектора**: 3х-компонентные, 2х-копонентные
- **Матрицы**: 4х4

Ресурсы рендера:

Сложные ресурсы рендера:

- **Примитив**: ресурс содержащий внутри себя вершинный массив, шейдер, матрицу всех стандартных преобразований + методы настроек всех этих полей
- **Модель**: множество примитивов, загружается из файла формата .obj (на данный момент поддерживается только данный формат)
- **Универсальная сцена:** сцена, которая создается из базовых примитивов + эффектов. Сцена способна рисоваться несколькими способами: обычным рендером или с помощью raymarching-a

Базовые ресурсы рендера:

- **Вершинный буфер**: множество точек с заданными характеристиками, пример характеристик: позиция, цвет, текстурная координата, нормаль к точке
- **Буфер индексов**: массив целых чисел, для определения порядка рисования точек (существует парно к каждому буфферу вершин)
- **Вершинный массив**: пара из вершинного буфера и буфера индексов, как раз этот объект имеет внутри себя метод «нарисуй»
- **Буффер данных для шейдеров (SSBO)**: буффер, который хранит в себе массив данных заданого типа, до которого есть доступ с шейдеров
- **Шейдер**: микропрограмма исполняемая на видеокарте

Задачи Кураленка Святослава (рендер):

- Сделано:
  - Модуль создания окна и обработка ввода от клавтатуры
  - Модуль инитиализации графической библеотеки OpenGL
  - Реализация ресурсов рендера:
    - Буффера (вершинный/индексов/вершинный массив/данных для шейдеров) – создание, удаление
    - Шейдера – загрузка, компиляция, удаление
    - Примитив – создание, метод отрисовки, удаление + методы настройки (управление видимостью, задавание матрицы стандартных преобразований – паралельный перенос, поворот, гомотетия, тд)
    - Модель – загрузка моделей формата .obj + все что и для примитива
  - Утилиты
    - Камера – настройка ее базиса + получение матрицы проекции по данным конкретной камеры
    - Модуль математики – вектора (скалярное/векторное произведения и тд) + матрицы (умножение/задание матриц поворотов/параллельных переносов и тд)
  - Реализация класса сцены:
    - Методы для управления ресурсами
    - Шаблон: методы для пользователя в которых он пишет свой код (onCreate, onUpdate, деструктор)
- Планы
  - Добавить текстуры
  - Добавить Frame Buffer Object (FBO – объект OpenGL-я позволяющий рендерить в отдельные текстуры, а не сразу же на основной экран)
  - Вынести реализации программистом-пользователем в отдельные плагины

Задачи Власова Дмитрия (raymarching):

- Сделано:
  - Система универсальной сцены
    - Добавление примитивов с материалами
    - Создание из примитивов более сложных объектов - их композиций
    - Добавление к примитивам и их композициям трансформаций и эффектов
    - Изменение трансформаций и эффектов перед отрисовкой
    - Установка способа отрисовки сцены
  - Шейдер raymarching-а
    - Примитивы (куб, сфера, плоскость)
    - Возможность применения матриц
    - Композиции примитивов (объединение, пересечение, разность)
    - Эффекты скручивания и изгиба примитивов

Демонтрация проекта:

github.com/DmitriyVlasovDV1/CppRMRT