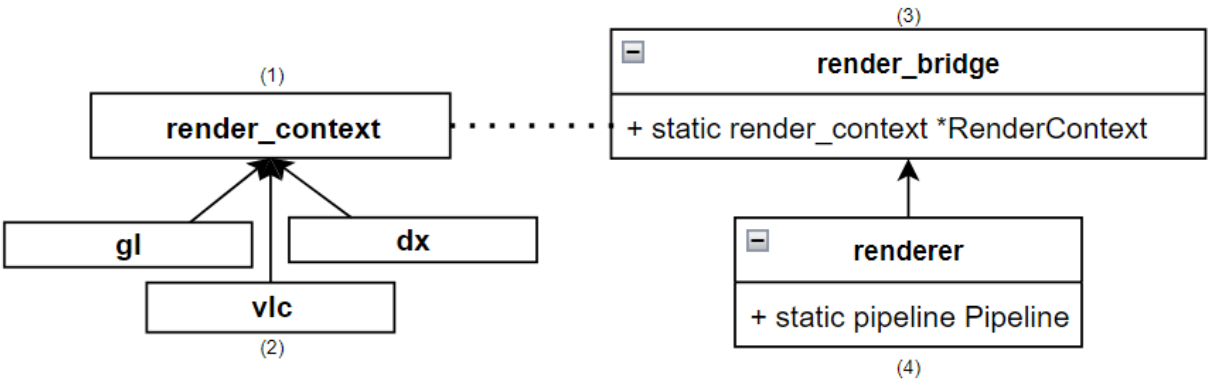
Описание всех модулей библиотеки sculpto.

1. *Модуль создания системных процессов, окон для приложения, обработки пользовательского ввода через HID.* Модуль абстрагирован от взаимодействия с API операционной системы напрямую и делает это через абстрактные интерфейсы. Для каждой платформы реализованы классы, наследуемые абстрактные интерфейсы и реализующие необходимый функционал с помощью API конкретной платформы – для Windows использован WinAPI, для macOS и Linux дополнительная библиотека GLFW. Во время инициализации приложения инстанцируются конкретными классами по паттерну абстрактной фабрики. Данный подход и дает поддержку кросс-платформенности.
2. *Несколько небольших подсистем приложения*: logger, timer (для подсчета FPS, времени между кадром, общего времени работы программы), определение платформы пользователя.
3. *Модуль для обработки и создания event'ов приложения*. Пример event'ов – изменение размеров окна/viewport'ов, пользовательский ввод (нажатие клавиш клавиатуры и мыши, движение мыши) и т.д.
4. *Модуль математики*. Модуль включает в себя объекты для работы и действиями над двумерными, трехмерными, четырехмерными векторами, матрицами третьего и четвертого порядка, углами Эйлера для задания вращения объектов. Модуль необходим, так как при рендере трехмерной сцены в двумерное изображение проводится большое количество преобразований координат.
5. *Модуль графического движка* (Render Engine) на подобии с первым модулем абстрагирован от низкоуровневых RenderAPI, обращение к ним происходит с помощью абстрактного контекста и абстрактных интерфейсов рендер примитивов (рендер примитив – любой объект, используемый для вывода изображения на экран, чаще всего хранится в видеопамяти; пример рендер примитивов – вершинный буфер, буфер индексов вершин, шейдерная программа, текстура). Это дает возможность использовать писать код для создания изображения не зависящий от конкретного RenderAPI, и использовать любой по необходимости (например, в зависимости от платформы или текущих задач) – DirectX/OpenGL/Vulkan/Metal. Для того что бы использовать конкретный RenderAPI необходимо создать конкретный контекст и рендер ресурсы, наследующие абстрактные интерфейсы и реализовать весь необходимый функционал, используя этот RenderAPI (на данный момент есть реализация интерфейсов только под OpenGL, в ближайшем будущем планируется добавить DirectX).



Диаграмма, иллюстрирующая общую архитектуру Render Engine'а.
(1) – абстрактный интерфейс Render Context'а
(2) – классы, имплементирующие интерфейс Render Context'а
(3) – статический класс, мост между нижним и верхним уровнями Render Engine'а. По сути, singleton обвязка над Render Context'ом, позволяющая создавать собственные Render Pipeline'ы.
(4) – основной класс верхнего уровня Render Engine'а, используется для сбора рендер ресурсов, их рендера через Render Pipeline.

6.

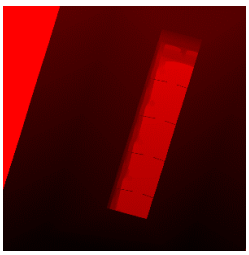
Рендер движок во время рендера использует так называемые рендер ресурсы. Рендер ресурс – абстрактное понятие, по сути, набор определенных рендер примитивов, облегчает использование рендер движка конечным пользователем. Реализованы следующие ресурсы:

- Материалы – ряд классов, объединяющий шейдерную программу, буфер, содержащий информацию о объекте, который использует материал (например параметры освещения) и ряд текстур. Все материалы наследуются от базового класса.
- Mesh – класс, объединяющий несколько вершинных и индексных буферов, а также ряд материалов. С помощью Assets Manager'а, который является часть библиотеки можно загрузить 3D модель и из не создать mesh со всеми необходимыми материалами.
- Камера – класс, объединяющий информацию о положении в пространстве "зрителя" (вектор позиции, точка, в которую он смотрит, вектор "вверх" от его положения), информацию о проецировании изображения на экран (тип проекции -ортогональная или перспективная, в зависимости от типа параметры ortho-бокса или frustum-конуса), матрицы View, Projection, View * Projection, построенные по положению зрителя и параметрам проецирования, а так же несколько Frame Buffer'ов для вывода изображения, которое "снимает" камера, на экран.

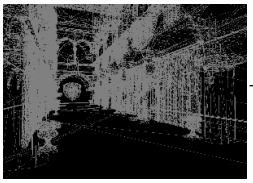
Организация процессов вывода изображения на экран происходит через Render Pipeline. Render Pipeline, изначально включенный в систему, позволяет создавать фотореалистичные изображения и использует следующие техники:

- Затенение объектов, с помощью модели Блина-Фонга.
- Instancing. Техника позволяет с минимальными потерями производительности рендерить множество объектов с идентичной геометрией.
- Normal mapping. Техника позволяет с минимальными потерями производительности увеличивать реалистичность объекта. Обычные нормали поверхности объекта заменяются на нормали, просчитываемые дополнительно. Из изображения Bump Map'а получаются нормали ориентированные в положительном направлении оси-z, затем для каждого треугольника строится матрица для их трансформации в касательное пространство.
- Shadow Mapping - алгоритм создания теней объектов, использующий так называемую карту теней для каждого источника света. Этот алгоритм в значительной степени увеличивает производительность по сравнению с другими алгоритмами создания теней, например: алгоритмом использующем проецирование каждого объекта друг на друга, алгоритмом построение теневой маски объекта и проективное наложение её на другие объекты, рядом алгоритмов, использующих трассировку лучей (последний использован в Path Tracer'е, созданном с помощью моей системы, который описан далее в этом документе).
- Техника отложенного освещение объектов, значительно увеличивающая производительность при рендере любых сложных сцен (сцен со сложной геометрией – большим количеством треугольников и множеством источников света).
- Эффект Bloom (реалистичное свечение очень ярких объектов сцены), использующий алгоритм двухпроходного Гауссового размытия для увеличения производительности.
- HDR Image (при отсутствии поддержки монитором изображение на этапе пост-обработке с помощью алгоритма Exposure Tone Mapping преобразуется в изображение со стандартным диапазоном цветов).

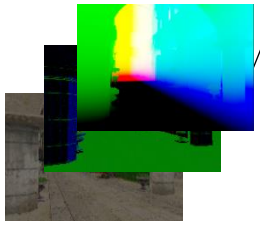
Диаграмма, демонстрирующая все проходы (Render Passes) Default Render Pipeline’а
(на примере рендера сцены с моделью Crytek Sponza и несколькими источниками света).



Этап заполнения буферов глубины всех источников света, создающих тени.



Этап растеризации. Трехмерная геометрия проецируется на двумерную плоскость.



Этап, на котором формируется геометрический буфер. Заполняются color attachment’ы – мировая позиция фрагмента, нормаль, диффузионный, спекулярный вклады, блеск объекта.

Этап отложенного освещения. Рассчитывается влияние всех источников освящения.

Два этапа:
- Применение эффекта Bloom. Яркие цвета, выделенные на предыдущем этапе, размываются, затем совмещаются с итоговым изображением.
- Экспозиционный Tone Mapping. HDR изображение переводится в изображение с цветовым диапазоном [0;1].

Созданный графический движок очень гибок и позволяет создавать пользователю собственные Render Pipeline’ы, новые типы материалов, шейдерные программы.

7. *Модуль динамических сцен*. Сцена – специальное хранилище абстрактных объектов сцены. Для создания этого модуля мной использован паттерн проектирования Entity Component System, который предполагает, что каждый объект – это просто хранилище компонентов, которые имеют необходимую функциональность. Например, на сцене может быть объект, которые имеет компоненты: имя – “Уличный фонарь”, соответствующую 3D модель, специальный скрипт, который контролирует включение/выключение фонаря в зависимости от времени суток, а также компонент прожекторного источника света, который освещает другие объекты, попадающие в его “конус”. В процессе использования сцены из хранилища могут быть получены объекты, имеющие один определенный компонент или определенную группу компонентов (хранилище организованно таким образом, что этот процесс происходит быстро и не ресурсоёмок). Таким образом при рендере сцены выбираются все объекты, которые используются рендер движком (3D модели, источники света) и передаются в Render Engine для вывода на экран. Так же реализована возможность сериализации и десериализации сцен, использован формат хранения данных JSON, для простоты ручного редактирования файлов сцены.
8. *Модуль графического юзер интерфейса*, включающий в себя API для вывода виджетов на экран, создания пользователем собственных виджетов, а также несколько стандартных виджетов:
    - виджет для управления параметрами Render Engine’а
    - profiler виджет, вывод статистики о быстродействии приложения
    - виджет для просмотра объектов сцены
    - виджет для редактирования объектов сцены и их компонентов
    - панель для отображения отрендеренного изображения (viewport).