Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный университет информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

Кафедра программного обеспечения информационных технологий

Дисциплина: Операционные системы и системное программирование

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовой работе

на тему

3D редактор с использованием технологии DirectX

БГУИР КР 1-40 01 01 235 ПЗ

Студент гр.951004: Сорока П. Л

Руководитель: Асс. Деменковец Д.В.

Минск 2021

Содержание

[ВВЕДЕНИЕ 4](#_Toc89969796)

[1 АНАЛИЗ ПРОТОТИПОВ, ЛИТЕРАТУРНЫХ ИСТОЧНИКОВ И ФОРМИРОВАНИЕ ТРЕБОВАНИЙ К ПРОЕКТИРУЕМОМУ ПРОГРАММНОМУ СРЕДСТВУ 5](#_Toc89969797)

[1.1 Анализ прототипов 5](#_Toc89969798)

[1.2 Формирования требований к ПС 7](#_Toc89969799)

[2 АНАЛИЗ ТРЕБОВАНИЙ К ПРОГРАММНОМУ СРЕДСТВУ И РАЗРАБОТКА ФУНКЦИОНАЛЬНЫХ ТРЕБОВАНИЙ 8](#_Toc89969800)

[2.1 Функциональные требования 8](#_Toc89969801)

[3 ПРОЕКТИРОВАНИЕ ПРОГРАММНОГО СРЕДСТВА 9](#_Toc89969802)

[3.1 Системы координат 9](#_Toc89969803)

[3.2 Наложение текстуры на модель 11](#_Toc89969804)

[3.3 Графический конвейер DirectX 11](#_Toc89969805)

[3.4 Проектирование системы 15](#_Toc89969806)

[4 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ 16](#_Toc89969807)

[5 ЗАКЛЮЧЕНИЕ 17](#_Toc89969808)

[СПИСОК ЛИТЕРАТУРНЫХ ИСТОЧНИКОВ 18](#_Toc89969809)

# ВВЕДЕНИЕ

В век информационных технологий, когда ни одна компания не может обойтись без программного обеспечения для более быстрого и качественного ведения бизнеса и увеличения прибыли в стороне не остается и сфера развлечений. Фильмы, игры, сериалы – множество из них основано на 3D графике. И, когда программы становятся всё более и более сложными и многофункциональными, качество графики также постоянно увеличивается. Аватар, Игра престолов, Cyberpunk, Ведьмак – эти шедевры не были бы возможны без современных технологий для создания графики. Они увлекают и объединяют миллионы людей по всему миру. А там где увлечение – там и большие деньги, поэтому отрасль не стоит на месте и постоянно развивается. Однако, всё приходит постепенно и необходимо с чего-то начинать.

В этом курсовом проекте будет создан относительно простой игровой движок с использованием технологии DirectX – это одна из ключевых базовых технологий для разработки игр, которая применяется практически во всех игровых движках.

В первом разделе будут рассмотрены профессиональные инструменты, которыми пользуются разработчики. На основе рассмотренных программ будут сформированы требования, который позволят воссоздать программное средство для воссоздания различных 3D объектов, текстурирования, настройки источников света.

В третьем разделе будет рассмотрен DirectX 3D, последовательность действий, необходимая для того, чтобы рисовать 3D объекты, шейдеры, буферы. Также будет спроектирована система, которая позволить автоматизировать процесс рисования, перемещение камеры. Будет создан графический интерфейс пользователя, основанный на сторонней библиотеке ImGui – простой и удобной в использовании относительно нативного интерфейса WINApi.

В 4 разделе будет протестировано программное средство, его соответствие функциональным требованиям. В 5 разделе же будет показан финальный вариант ПС, предоставлено руководство пользователя.

В заключении будет проанализирована проделанная работа и сделаны выводы о программном средстве.

# АНАЛИЗ ПРОТОТИПОВ, ЛИТЕРАТУРНЫХ ИСТОЧНИКОВ И ФОРМИРОВАНИЕ ТРЕБОВАНИЙ К ПРОЕКТИРУЕМОМУ ПРОГРАММНОМУ СРЕДСТВУ

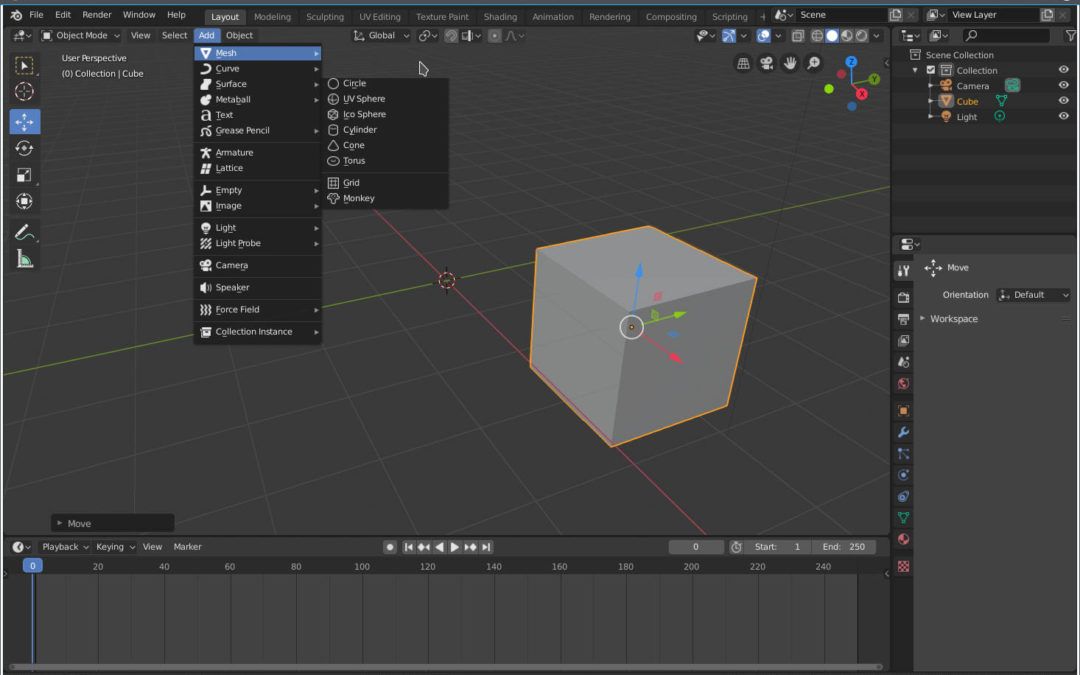
В наш век стремительного развития информационных технологий широко распространена 3D графика и существует множество инструментов для работы с ними: от моделирования различных строений до созданий красочных и масштабных игр или фильмов. Для каждой из задач было создано множество инструментов, ориентированных специально для их решения. В этом разделе будет рассмотрено несколько из них.

## Анализ прототипов

**1.1.1 Blender**

**Blender** — профессиональное свободное и открытое программное обеспечение для создания трёхмерной компьютерной графики, включающее в себя средства моделирования, скульптинга, анимации, симуляции, рендеринга, постобработки и монтажа видео со звуком, компоновки с помощью «узлов» (Node Compositing), а также создания 2D-анимаций. В настоящее время пользуется большой популярностью среди бесплатных 3D-редакторов в связи с его быстрым стабильным развитием и технической поддержкой. [1]

*Рисунок 1.1*

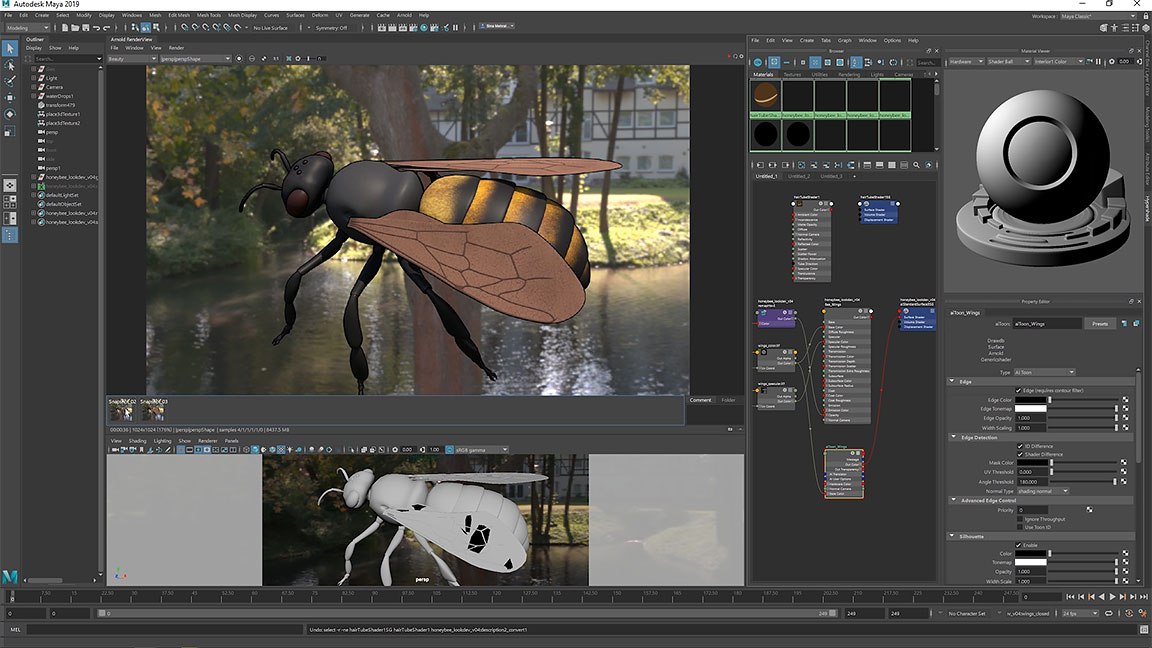


Как следует из описания – это очень мощный инструмент, который в умелых руках позволяет решать множество задач и предоставляет богатый список инструментов для работы с 3D графикой. Также стоит выделить то, что программа является бесплатной, что обеспечивает ей огромную популярность среди пользователей, а открытый исходный код позволяет создавать множество различных дополнений для программы. Однако из-за столь широкого разнообразия инструментов неопытный пользователь легко запутается в данной программе, поэтому необходимо потратить много времени для подробного изучения каждого из них для начала работы.

**1.1.2 Autodesk Maya**

**Autodesk Maya** — редактор трёхмерной графики, доступный на Windows, macOS и Linux. Maya обладает широкой функциональностью 3D-анимации, моделирования и визуализации. Программу используют для создания анимации, сред, графики движения, виртуальной реальности и персонажей. Широко применяется в кинематографии, телевидении и игровой индустрии. Изначально разработан Alias Systems Corporation, а затем выкуплен и поддерживается в настоящее время Autodesk, Inc. [2]

*Рисунок 1.2*



В программе есть все, что необходимо для создания трехмерной графики. Maya позволяет пройти все этапы создания 3D — от моделирования и анимации до текстурирования, композитинга и послойного рендеринга. Этот трехмерный редактор может моделировать физику твердых и мягких тел, просчитывать поведение ткани, эмулировать текучие эффекты, позволяет детально настраивать прическу персонажей, создавать сухой и мокрый мех, анимировать волосы и т. д. Визитной карточкой программы является модуль PaintEffects, который дает возможность рисовать виртуальной кистью такие трехмерные объекты, как цветы, трава, объемные узоры и прочее. Программа довольно сложна в освоении, что компенсируется большим количеством уроков по данному редактору. Кроме того, Maya очень удобна в работе.

Этот инструмент подойдет профессионалам, которые уже обладают большим багажом знаний и опытом. Но за предложенный список инструментов придется заплатить – 1700$ в год – внушительная цифра, но если это позволит повысить продуктивность и создавать более качественный продукт, то это достойная цена, которая в последующем окупить себя.

## Формирования требований к ПС

На

# АНАЛИЗ ТРЕБОВАНИЙ К ПРОГРАММНОМУ СРЕДСТВУ И РАЗРАБОТКА ФУНКЦИОНАЛЬНЫХ ТРЕБОВАНИЙ

## Функциональные требования

* Возможность добавления примитивов на экран
  + Пирамид
  + Кубов
  + Шаров
  + Конусов
  + Призм
* Наложение на примитивы текстур из файла
* Возможность добавления источника света, который будет изменять яркость фигур в зависимости от положения.
* Возможность загрузки 3D объектов из файлов *.obj*
* Возможность перемещения фигур в пространстве
* Возможность изменения положения камеры
* Графический интерфейс, который позволит:
  + Перемещать источника света
  + Осуществлять выбор фигуры рисования для последующего его перемещения
  + Загружать 3D объекты из файла
  + Задавать текстуру объекта
  + Перемещать камеру
* Возможность скрыть интерфейс
* Графический интерфейс должен быть разделен на несколько отдельных окон для группировки различных настроек.
* Возможность загрузки шейдеров из файла
* Ошибки и исключения отображаются в отдельном окне, в котором представлена следующая информация:
  + Имя файла
  + Строка и столбец, где произошла ошибка или исключение
  + Текст с описанием ошибки

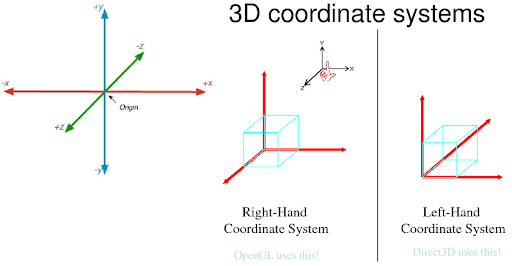
# ПРОЕКТИРОВАНИЕ ПРОГРАММНОГО СРЕДСТВА

## Системы координат

Отображение объектов требует преобразования точек из одной системы координат в другую систему координат для корректного отображения фигур на экран. В программировании 3D графики было придумано множество систем координат для вычисления конечных координат точек на экране.

**3.1.1 Мировая система координат**

*Рисунок 3.1 – Мировая система координат*

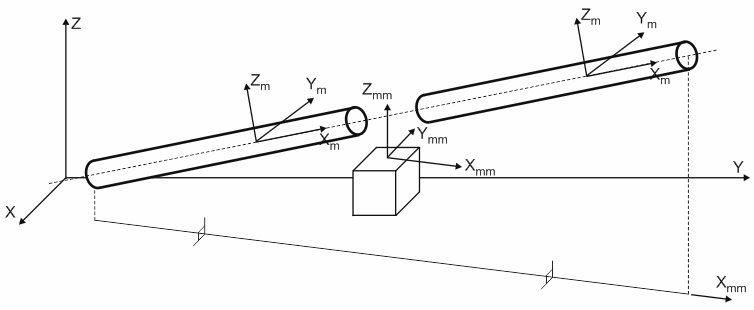
****

На рисунке 3.1 представлена мировая система координат. Это система в которой цент находится по середине экрана и от него идет 3 оси – oX, oY, Oz. Направление осей меняется в зависимости от предпочтений разработчика. В основном используют 2 вида – система правой руки, то есть ось X направлена вправо, ось Y – вверх и Z – в направлении к камере и система левой руки – ось Z направлена от камеры. В библиотеке Direct3D используется система координат левой руки, а в OpenGL – правой.

**3.1.2 Локальная система координат**

Также для каждого объекта можно ввести локальную систему координат. Она создана для удобства, так как позволяет не зависеть от мировой системы координат. Для каждого объекта будет существовать своя система и точки объектов будут выстраиваться вокруг нее. В последующем с помощью различных матриц трансформации можно преобразовать локальную систему в мировую.

*Рисунок 3.2 – Локальная система координат*

****

*Матрица масштабирования:*

*Scale matrix =*

Умножая точки из локальной системы координат на данную матрицу можно получить увеличенный или уменьшенный объект. Sx – масштабирование по оси X, Sy – по оси Y, Sz – по оси Z.

*Матрица перемещения:*

Матрица перемещения позволяет переместить точки объекта на tx единиц по оси x, ty единиц по оси y и tz единиц по ози z.

*Матрица вращения:*

Где u = (ux, uy, uz) – вектор оси вращения и – угол вращения

**3.1.3 Система координат просмотра (View space)**

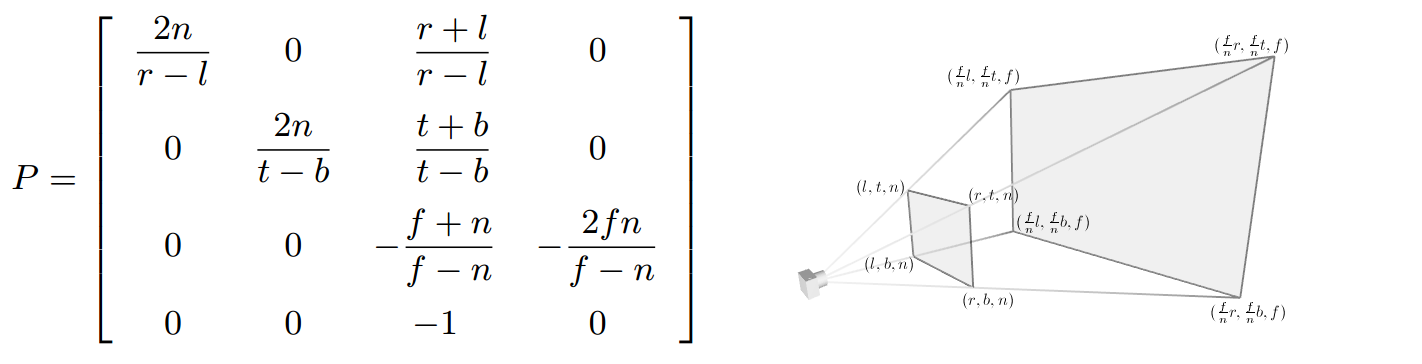
После того, как координаты объекта были преобразованы в мировые, необходимо определить положение объекта относительно камеры. Чтобы упросить задачу вычисления координат объектов относительно камеры можно считать, что камера находится в центре системы координат, и, вместо того, чтобы передвигать камеру, можно передвинуть мир вокруг камеры. То есть если разработчику необходимо передвинуть камеру на 3 единицы вправо, нужно передвинуть мир на 3 единицы влево. То есть использовать те же матрицы трансформации, что и для перевода из локальной системы координат в мировую, но с обратными значениями.

**3.1.4 Система координат экрана**

Система координат экрана представляет собой декартову систему координат, в которой центр отсчета (0;0) находится в левом верхнем углу экрана, ось Y направлена вниз, ось X – вправо.

После перехода в системы координат камеры необходимо преобразовать 3-х мерное пространство в двухмерное. Для этого используется матрица проекции.

*Рисунок 3.3 - Матрица проекции*



**3.1.5 Итог**

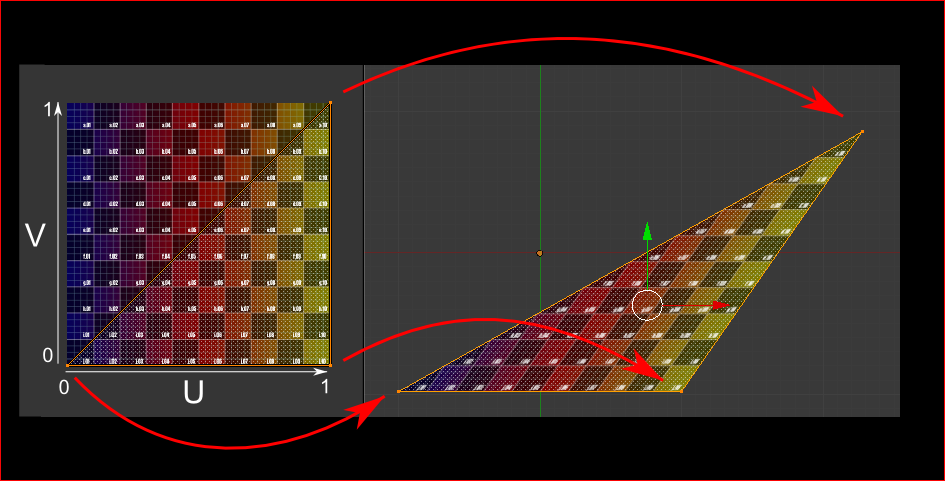
Для того, чтобы правильно отобразить объект на экран монитора, необходимо сделать множество трансформации и переводов из одной системы координат в другую. Обычно итоговую матрицу, которая преобразует координаты вершины в конечные координаты, готовые для отображения, называют MVP матрицей – Model-View-Projection. Каждый ее множитель ответственный за перевод из одной системы в другу.

DirectX 3D имеет в себе встроенную библиотеку DirectXMath.h для работы с матрицами, векторами и их преобразованиями – поворотами, перемещением, скалированием, проецированием и множество других.

## Наложение текстуры на модель

Для наложения 2х мерной текстуры на 3х-мерную модель используется система UV координат. Для каждой точки в примитиве указывается координаты U и V – значения от 0 до 1. Они представляют собой нормализованный вид координат X и Y на текстуре, которую необходимо наложить на примитив. Пример можно увидеть на рисунке 3.4

*Рисунок 3.4*



Для того, чтобы загрузить текстуру в графический конвейер и наложить ее на 3D объект пиксельному шейдеру необходимо предоставить:

1. Пару нормализованных UV координат текстуры
2. Описание текстуры, которое содержит высоту, ширину, формат пикселей, массив пикселей и различные системные параметры. Для этого используется интерфейс ID3D11ShaderResourceView
3. Модель поведения при условии, что текстура не полностью накладывается на объект – повторить текстуру, продлить до конца примитива крайние пиксели или оставить пустое пространство. ID3D11SamplerState – интерфейс, который предоставляет это поведение

## Динамическое освещение

Для создания иллюзии освещения необходимо два объекта: источник света и объект освещения.

**3.3.1 Типы источников света**

Существуют различные типы источников света. Они представлены в таблице 3.1.

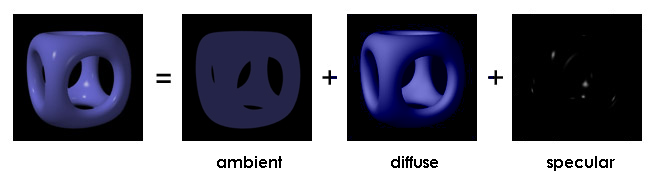
*Таблица 3.1*

| Тип | Описание |
| --- | --- |
| Точечный источник света (Point light source) | Излучает свет из одной точки по всем возможным направлениям. С увеличением расстояния до объекта освещения уменьшается интенсивность освещения. Примером такого источника света является электрическая лампа. |
| Направленный источник света (Directional light source) | Направленный источник света (или удаленный источник света) освещает всё сцену для рисования в одном направлении. В отличие от точечного источника света интенсивность света не меняется с удалением объекта от него, так как считается, что источник находится на очень большом расстоянии от объектов. Примером такого источника света может являться солнечный свет |
| Прожекторный источник света (Spotlight light source) | Этот источник света излучает конусообразный свет. Интенсивность света в этом случае зависит не только от расстояния до объекта, но и от удаления объекта от цента основания конуса. Примером такого источника света может быть фонарик. |
| Окружающее освещение (Ambient light source) | Окружающий источник света подсвечивает объекты даже когда никаких других источников света не предоставлено. Интенсивность света не зависит от направления, расстояния или других объектов. Это значит, что на всей сцене рисования эффект освещения будет однообразным. Данный источник света необходим для того, чтобы объекты были видны даже в полной темноте. |

**3.3.2 Взаимодействие источника света с объектом**

В компьютерной графике освещение, обычно, состоит из нескольких компонентов. Итоговый эффект определяется комбинацией взаимодействия объекта со всеми этими компонентами. Обычно выделяют три основных компонента – диффузионный (diffuse), окружающий (ambient) и зеркальный (specular). Пример такого освещения можно увидеть на рисунке 3.5

*Рисунок 3.5*



**Диффузия**

Диффузионный свет (или диффузионное отражение) является основным способом освещения объекта. Значение яркости поверхности, освещенной этим способом, является функцией от параметров поверхности объекта и угла исходящего света. Данный тип взаимодействия с объектом является основным фактором итоговых яркости и сформированном цвете поверхности.

**Окружающий тип взаимодействия**

Так как окружающий источник света является ненаправленным, он взаимодействует с объектом одинаково по всей его поверхности. Интенсивность освещения главным образом определяется силой окружающего источника света и свойствами материала поверхности.

**Зеркальный тип взаимодействия**

Данный компонент придает объектам блеск и выделение. Он не является “зеркальным” в прямом смысле этого слова, так как не показывает отражение других объектов на поверхности. Вместо этого он создает яркие точки на объекте, яркость которых зависит от параметров источника света и коэффициента отражения (specular reflection coefficient) поверхности.

**3.3.3 Модели освещения**

Модели освещения используются для имитации световых эффектов в визуализированных средах, где свет основывается на физике света. Без моделей освещения воспроизведение световых эффектов в естественном виде потребует большой вычислительной мощности, которой будет недостаточно на среднестатистическом компьютере. Модель освещения предназначена для вычисления цвета каждого пикселя или количества света, отраженного от различных поверхностей. Существует две основные модели освещения: объектно-ориентированное освещение и глобальное освещение. Они отличаются тем, что объектно-ориентированное освещение рассматривает каждый объект индивидуально, тогда как глобальная модель освещение показывает, как свет взаимодействует между объектами. В настоящее время исследователи разрабатывают методы глобального освещения, чтобы более точно воспроизвести, как свет взаимодействует с окружающей средой.

**Объектно-ориентированная модель Фонг**

В работе будет использована объектно-ориентированная модель Фонг, которая была спроектирована исследователем компьютерной графики Буи Тонг Фонгом. Данная модель предполагает, что интенсивность каждого пикселя является суммой интенсивностей диффузионного, зеркального и окружающего освещения. Модель учитывает расположение наблюдателя в пространстве, чтобы определить интенсивность зеркального света используя угол отражения света от объекта. Косинус угла высчитывается и возводится в степень, которая определяется дизайнером. С помощью этого дизайнер может определить, насколько сильно будет выделен объект. Параметр силы, с которой будет подсвечен объект, называется интенсивность блеска (Shininess value). Интенсивность блеска определяется жесткостью поверхности, где, например, зеркало будет иметь значение, стремящееся к бесконечности, а самая грубая поверхность. Данная модель обеспечивает более реалистичный вид объекта с отблеском, основанным на перспективе наблюдателя. [4]

## Графический конвейер DirectX

Весь процесс рисования в DirectX 3D проходит через определенную последовательность этапов, которая называется графический конвейер (Graphics Pipeline). Он изображен на рисунке 3.1. Графический конвейер Direct3D предназначен для создания графики для игр, действие в которых разворачивается в реальном времени. Данные поступают с устройств ввода на устройства вывода, проходя через все настраиваемые или программируемые этапы. Все этапы можно настроить с помощью API Direct3D. Этапы с использованием общих ядер шейдеров (прямоугольные блоки со скругленными краями) можно программировать с помощью языка программирования HLSL. Это обеспечивает высочайшую гибкость и адаптивность конвейера.

Наиболее распространены этапы шейдера вершин (VS) и пиксельного шейдера (PS). Они являются обязательными для простейшей программы. [3]

* + 1. **Этап сборки входных данных**

Этап сборки входных данных (IA) предоставляет конвейеру данные о примитивах и выравнивании, такие как треугольники, линии и точки, включая идентификаторы семантики, чтобы сделать шейдеры более эффективными, включая обработку только тех примитивов, которые еще не были обработаны.

*Входные данные*

Данные примитивов (треугольники, линии или точки) из заполненных пользователем буферов в памяти. А также, возможно, данные о выравнивании в памяти. Для каждого треугольника есть данные трех вершин и, возможно, еще трех смежных вершин.

*Выходные данные*

Примитивы с системными значениями (такими как id примитива, id экземпляра или id вершины).

Чтобы задать формат входных данных используется класс *DirectX::ID3D11InputLayout*. Для его создания необходим байткод шейдера вершин, размер в байтах байткода, описание входных элементов, которые представляются структурой D3D11\_INPUT\_ELEMENT\_DESC (ее формат представлен в таблице 3.2) и количество входных элементов.

*Таблица 3.2 – Описание структуры D3D11\_INPUT\_ELEMENT\_DESC*

| Поле | Тип | Описание |
| --- | --- | --- |
| SemanticName | LPCSTR | Имя HLSL семантики, которая ассоциируется с входным элементом данных шейдера вершин |
| SematicIndex | UINT | Индекс семантики, который необходим, если существуют повторяющиеся имена семантики |
| Format | DXGI\_ FORMAT | Тип данных. Например R8G8B8A8 представляет собой набор из 4 8-байтовых чисел |
| InputSlot | UINT | Целочисленное значение, которое представляет собой слот для сборщика входных данных (должен иметь значение от 0 до 15) |
| AlignedByte  Offset | UINT | Смещение данных в памяти |
| InstanceSlot  Class | D3D11\_ INPUT\_ CLASSIFI CATION | Определяет класс данных для слота – данные для одной вершины или данные для примитива. |
| InstanceData  StepRate | UINT | Количество вершин, которые описывают один примитив. Необходимо указывать, если InstanceSlotClass указан как класс примитива. |

Чтобы назначить сборщик входных данных используется функция контекста *ID3D11DeviceContext::IASetInputLayout*.

**3.4.2 Этап шейдера вершин**

Этап шейдера вершин (VS) обрабатывает вершины, обычно выполняя такие операции, как трансформация, текстурирование и освещение. Шейдер вершин принимает одну входную вершину и создает одну выходную вершину. Отдельные операции по вершинам, такие как трансформация, текстурирование и освещение для каждой вершинам.

*Входные данные*

Одна вершина с созданными системой значениями VertexID и InstanceID. Каждая входная вершина шейдера вершин может состоять из 16 32-разрядных векторов (до 4 компонентов в каждом).

*Выходные данныe*

Одна вершина. Каждая выходная вершина может состоять из 16 32-разрядных четырехкомпонентных векторов.

В программе представляется классом *ID3D11VertexShader.* Для создания экземпляра необходимо передать указатель на буфер с байткодом скомпилированного HLSL шейдера вершин, который можно получить с помощью функции D3DReadFileToBlob и размер буфера. Назначить конвейеру данный шейдер можно с использованием функции котекста *ID3D11DeviceContext::VSSetShader*.

**3.4.3 Этап шейдера поверхностей**

Этап шейдера поверхностей (HS) — один из этапов тесселяции, на котором одна поверхность модели фактически разбивается на множество треугольников. Шейдер поверхностей вызывается единожды для каждого участка и преобразует входные контрольные точки, определяющие поверхность низкого порядка, в контрольные точки, формирующие участок. Он также выполняет некоторые вычисления для каждого участка, чтобы предоставить данные для этапа тесселяции (TS) и шейдера доменов (DS).

*Входные данные*

От 1 до 32 входных контрольных точек, которые вместе определяют поверхность низкого порядка.

*Выходные данные*

От 1 до 32 выходных контрольных точек, которые вместе образуют участок. Шейдер поверхностей объявляет состояние для этапа тесселяции (TS), включая количество контрольных точек, тип поверхности участка и нужный тип секционирования при тесселяции.

**3.2.4 Этап тесселяции**

На этапе тесселяции (TS) создается шаблон выборки домена, представляющего участок геометрии, и генерируется набор небольших объектов (треугольников, точек или линий), которые объединяются в этот участок.

*Входные данные*

Тесселяция выполняется единожды для каждого участка с использованием факторов тесселяции (указывают тщательность тесселяции домена) и типа разделения (указывает алгоритм разделения участка на части), переданных с предыдущего этапа шейдера поверхностей.

*Выходные данные*

Тесселяция выводит координаты uv (дополнительно — w) и топологию поверхности для этапа шейдера доменов.

**3.2.5 Этап шейдера доменов**

На этапе шейдера доменов (DS) вычисляется положение вершины составной точки в выходном участке. Вычисляется положение вершины, соответствующее каждой выборке домена. Шейдер доменов запускается единожды на каждую выходную точку этапа тесселяции и имеет доступ только для чтения к выходному участку шейдера поверхности и константам выходного участка, а также к координатам UV выходных данных этапа тесселяции.

*Входные данные*

Шейдер доменов использует выходные контрольные точки этапа шейдера поверхности (HS). Выходные данные шейдера доменов включают контрольные точки, данные констант участков и факторы тесселяции (факторы тесселяции могут включать значения, используемые тесселятором с фиксированными функциями, а также необработанные значения — перед округлением с помощью целочисленной тесселяции, — например для упрощения геоморфизма). Шейдер доменов вызывается единожды на каждую выходную координату этапа тесселяции (TS).

*Выходные данные*

Этап шейдера доменов (DS) выводит положение вершины составной точки в выходном участке.

**3.2.6 Этап шейдера геометрии**

На этапе шейдера геометрии (GS) обрабатываются все примитивы: треугольники, линии и точки, а также соседние вершины. Он поддерживает усложнение и упрощение геометрии. Он полезен для таких алгоритмов, как расширение спрайтов точек, динамические системы частиц, создание меха/"плавников", создание объемных теней, однопроходная отрисовка с использованием кубической карты, замена материалов по примитивам и настройка материалов по примитивам, включая создание барицентрических координат в качестве данных примитива, чтобы пиксельный шейдер мог выполнить настраиваемую интерполяцию атрибутов.

*Входные данные*

В отличие от шейдеров вершин, работающих с одной вершиной, входными данными шейдера геометрии являются вершины полного примитива (три вершины для треугольников, две вершины для линий или одна вершина для точек).

*Выходные данные*

Этап шейдера геометрии может выводить несколько вершин, формирующих единую выбранную топологию. Доступны следующие выходные топологии шейдера геометрии: tristrip, linestrip и pointlist. Количество сгенерированных примитивов может свободно меняться при каждом вызове шейдера геометрии, но максимальное количество генерируемых вершин необходимо объявить статически. Длины полос, выдаваемые из вызова шейдера геометрии, могут быть произвольными, а новые ленты можно создавать с помощью функции HLSL RestartStrpip.

**3.2.7 Этап потокового вывода**

Этап потокового вывода (SO) позволяет непрерывно выводить данные вершин из предыдущей активной стадии в один или несколько буферов в памяти. Поток данных, выведенный в память, можно снова вернуть в конвейер как входные данные или обратно считать из ЦП.

*Входные данные*

Данные вершин из предыдущего этапа конвейера.

*Выходные данные*

Этап потокового вывода (SO) непрерывно выводит данных вершин из предыдущего активного этапа, например этапа геометрического шейдера (GS), в один или несколько буферов в памяти. Если этап шейдера геометрии (GS) неактивен, а этап потокового вывода (SO) активен, оно непрерывно выводит данные вершин из этапа шейдера доменов (DS) в буферы в памяти (а если этап шейдера доменов также неактивен, то из этапа шейдера вершин (VS)).

**3.2.8 Этап растеризации**

Этап растеризации (RS) обрезает примитивы, которых нет в представлении, готовит их для этапа пиксельного шейдера (PS) и определяет, как вызывать эти шейдеры. Векторные данные (состоящие из фигур или примитивов) преобразуются в растровое изображение (состоящее из пикселей) для отображения трехмерной графики в режиме реального времени.

*Входные данные*

Предполагается, что вершины (x, y, z, w), поступающие на этап растеризации, находятся однородном пространстве обрезки. В этом пространстве координат точки оси X указывают вправо, точки Y — вверх, а точки Z — по направлению от камеры.

*Выходные данные*

Фактические пиксели для отрисовки. Включает некоторые атрибуты вершин для интерполяции построителем текстуры.

**3.2.9 Пиксельный шейдер**

На этапе построителя текстуры (PS) принимаются интерполированные данные для примитива и генерируются данные для пикселей, такие как цвет.

*Входные данные*

Если конвейер настроен без шейдера геометрии, шейдер текстуры ограничивается 16 32-разрядными и 4-компонентными входными данными. В противном случае шейдер пикселей может принимать до 32 32-разрядных, 4-компонентных входных данных. Входные данные шейдера пикселей включают атрибуты вершин (которые можно интерполировать с коррекцией перспективы или без нее) или могут рассматриваться как константы примитивов. Входные данные шейдера пикселей интерполируются на основе атрибутов вершин растеризуемого примитива в зависимости от объявленного режима интерполяции. Если примитив обрезается до растеризации, режим интерполяции также учитывается в процессе обрезки.

*Выходные данные*

Пиксельный шейдер может вывести до 8 32-разрядных 4-компонентных цветов (если пиксель отклоняется, цвет не предоставляется). Компоненты регистрация выходных данных шейдера пикселей следует объявить перед использованием. У каждого регистра может быть отдельная маска вывода записи.

**3.2.10 Этап слияния вывода**

На этапе выходной слияния (OM) объединяются различные типы выходных данных (значения шейдера пикселей, сведения о глубине и трафарете) с содержимым целевого объекта отрисовки и буферами глубины и шаблона для создания конечного результата конвейера.

*Входные данные*

Входные данные средства слияния вывода — это состояние конвейера, пиксельные данные, сгенерированные построителями текстур, содержимое целевых объектов отрисовки и буферов глубины и трафаретов.

*Выходные данные*

Окончательный цвет отрисованных пикселей.

## Проектирование системы

Система классов, которая представляет п

# РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

# ЗАКЛЮЧЕНИЕ

# СПИСОК ЛИТЕРАТУРНЫХ ИСТОЧНИКОВ

[1] <https://ru.wikipedia.org/wiki/Blender>

[2] <https://ru.wikipedia.org/wiki/Autodesk_Maya>

[3] <https://docs.microsoft.com/ru-ru/windows/uwp/graphics-concepts/graphics-pipeline>

[4] Перевод http://www.bcchang.com/immersive/ygbasics/lighting.html