1. **Основные понятия, задачи и сферы применения компьютерной графики.**

Компьютерная графика — это область информатики, изучающая методы создания, обработки и представления графических данных с помощью компьютеров(вычислительных устройств). Она позволяет отобразить любые данные на устройстве вывода, используя различные методы визуализации. Компьютерная графика играет ключевую роль в современной мультимедийной программе, обеспечивая визуальное представление информации, будь то текстовые данные, изображения, анимации или трёхмерные модели.

История развития компьютерной графики

Компьютерная графика сформировалась как наука об аппаратном и программном обеспечении для создания изображений, начиная от простых чертежей до реалистичных образов естественных объектов. Её развитие прошло через несколько этапов, начиная с первых компьютеров, которые использовались для решения научных и производственных задач, и до современных высокотехнологичных решений, позволяющих создавать сложные графические изображения и анимации.

Основные направления компьютерной графики

Компьютерная графика включает в себя несколько основных направлений:

Распознавание образов (Computer Vision): Используется для получения описания изображения или его классификации. Например, при сортировке почты или распознавании лиц.

Обработка изображений (Image Processing): Занимается преобразованием изображений, включая улучшение качества, восстановление и специальные преобразования.

Машинная графика (Computer Graphics): Воспроизводит изображение из информации неизобразительной природы, например, визуализация экспериментальных данных или создание сцен в компьютерных играх.

Сферы применения компьютерной графики

Научная графика: Используется для наглядного представления результатов исследований, проведения вычислительных экспериментов.

Деловая графика: Предназначена для наглядного представления показателей работы учреждений, включая планы, отчётную документацию и статистику.

Конструкторская графика: Используется в архитектуре, инженерии и конструировании для создания проектных чертежей и моделей.

Иллюстративная графика: Обеспечивает создание графических материалов для обучения, презентаций и других целей.

Художественная и рекламная графика: Используется для создания рекламных материалов, мультфильмов, компьютерных игр и видеопрезентаций.

Основные понятия компьютерной графики:

Растровая графика - метод представления изображений, который использует матрицу пикселей для представления отдельных точек изображения. Каждый пиксель имеет свой цвет, который определяется через RGB или CMYK значения. Растровые изображения могут быть изменены в размерах, но при этом может возникнуть потеря качества.

Векторная графика - метод представления изображений, использующий для их создания совокупность векторов, определяемых математическими уравнениями. Векторные изображения не теряют качества при масштабировании, что делает их идеальными для логотипов, иллюстраций и других элементов дизайна, требующих высокой чёткости.

3D-моделирование - процесс создания трёхмерных объектов с использованием компьютерных программ. Эти объекты могут быть анимированы и освещены для создания реалистичного вида.

Анимация - процесс создания последовательности изображений или кадров, которые, когда они воспроизводятся быстро, создают иллюзию движения. Анимация используется в фильмах, видеоиграх и интерактивных приложениях.

Рендеринг - процесс генерации изображения или анимации с использованием компьютера. Рендеринг может включать в себя освещение, текстурирование, шейдеры и другие техники для достижения желаемого визуального результата.

Текстура — это растровое изображение, которое накладывается на поверхность полигональной модели для придания ей цвета, окраски или иллюзии рельефа.

Шейдеры - программы, написанные на языках шейдеров (например, GLSL), которые определяют, как свет и материалы будут взаимодействовать на поверхности 3D-объектов. Шейдеры используются для создания реалистичных текстур и освещения.

1. **Растровая и векторная графика. Основные характеристики растровых изображений.**

Растровая графика - метод представления изображений, который использует матрицу пикселей для представления отдельных точек изображения. Каждый пиксель имеет свой цвет, который определяется через RGB или CMYK значения. Растровые изображения могут быть изменены в размерах, но при этом может возникнуть потеря качества.

Векторная графика - метод представления изображений, использующий для их создания совокупность векторов, определяемых математическими уравнениями. Векторные изображения не теряют качества при масштабировании, что делает их идеальными для логотипов, иллюстраций и других элементов дизайна, требующих высокой чёткости.

Основные характеристики растрового изображения:

Размер изображения в пикселях — выражается в виде количества пикселей по ширине и по высоте (800 × 600, 1024 × 768, 1600 × 1200 пикселей и т. д.) или же в виде общего количества пикселей.

Глубина цвета  — количество [бит](https://ru.wikipedia.org/wiki/%D0%91%D0%B8%D1%82" \o "Бит), используемое для хранения и представления [цвета](https://ru.wikipedia.org/wiki/%D0%A6%D0%B2%D0%B5%D1%82" \o "Цвет) при [кодировании](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%B4" \o "Код), либо одного [пикселя](https://ru.wikipedia.org/wiki/%D0%9F%D0%B8%D0%BA%D1%81%D0%B5%D0%BB%D1%8C" \o "Пиксель) [растровой графики](https://ru.wikipedia.org/wiki/%D0%A0%D0%B0%D1%81%D1%82%D1%80%D0%BE%D0%B2%D0%B0%D1%8F_%D0%B3%D1%80%D0%B0%D1%84%D0%B8%D0%BA%D0%B0" \o "Растровая графика), количество цветов при этом определяется как 2 в степени количества бит(например для глубины цвета в 1 байт возможно кодирование 2^8 = 256 цветов).

[Цветовая модель](https://ru.wikipedia.org/wiki/%D0%A6%D0%B2%D0%B5%D1%82%D0%BE%D0%B2%D0%B0%D1%8F_%D0%BC%D0%BE%D0%B4%D0%B5%D0%BB%D1%8C" \o "Цветовая модель) — описание цвета за счет его представления в качестве соотношения цветовых компонент определённого набора( [RGB](https://ru.wikipedia.org/wiki/RGB" \o "RGB), [CMYK](https://ru.wikipedia.org/wiki/CMYK" \o "CMYK), [XYZ](https://ru.wikipedia.org/wiki/%D0%A6%D0%B2%D0%B5%D1%82%D0%BE%D0%B2%D0%B0%D1%8F_%D0%BC%D0%BE%D0%B4%D0%B5%D0%BB%D1%8C" \l "%D0%A6%D0%B2%D0%B5%D1%82%D0%BE%D0%B2%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D1%81%D1%82%D1%80%D0%B0%D0%BD%D1%81%D1%82%D0%B2%D0%BE_CIE_XYZ" \o "Цветовая модель), [YCbCr](https://ru.wikipedia.org/wiki/YCbCr" \o "YCbCr) и др.).

[Разрешение изображения](https://ru.wikipedia.org/wiki/%D0%A0%D0%B0%D0%B7%D1%80%D0%B5%D1%88%D0%B5%D0%BD%D0%B8%D0%B5_(%D0%BA%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80%D0%BD%D0%B0%D1%8F_%D0%B3%D1%80%D0%B0%D1%84%D0%B8%D0%BA%D0%B0)" \o "Разрешение (компьютерная графика)) — величина, определяющая количество точек (элементов растрового изображения) на единицу площади (или единицу длины), наиболее часто измеряется в dpi(число точек на дюйм).

1. **Свойства и применение скалярного произведения векторов в компьютерной графике.**

Скалярным произведением двух векторов*а* и *b* называется ЧИСЛО, равное произведению длин этих векторов на косинус угла между ними:

Свойства скалярного произведения:

1) IMG_256 – **коммутативный** закон скалярного произведения.

2) IMG_257 – **дистрибутивный** закон скалярного произведения. Попросту, можно раскрывать скобки.

3) IMG_258 – **ассоциативный** закон скалярного произведения. Константу можно вынести из скалярного произведения.

Применение скалярного произведения в компьютерной графике:

Определение угла между векторами: Скалярное произведение позволяет вычислить косинус угла между двумя векторами. Это полезно при определении направления движения объектов в анимации, например, при перемещении персонажей или объектов в играх.

Проекция векторов: Проекция одного вектора на другой может быть выполнена с помощью скалярного произведения. Это важно для определения проекции вектора скорости на ось движения, что позволяет анимировать движение объектов в соответствии с определёнными направлениями.

Матрицы преобразования: скалярное произведение может использоваться для преобразования вершины полигона в другое пространство, например в пространство наблюдателя.

Формирование освещения: для определения интенсивности освещения в отдельных вершинах используется скалярное произведение, для нормализованных векторов значение интенсивности также нормировано(от 0 до 1).

Оптимизация алгоритмов: Скалярное произведение используется в оптимизаторах для ускорения процесса отрисовки, например за счёт исключения из обработки невидимых поверхностей(backface culling), скалярное произведение можно использовать для определения угла между вектором взгляда и нормалью к полигону, чтобы определить его видимость на экране.

1. **Свойства и применение векторного произведения векторов в компьютерной графике.**

Векторным произведением векторов a и b называется ВЕКТОР n, длина которого численно равна площади параллелограмма, построенного на данных векторах; вектор n ортогонален векторам a и b, вектор n направлен таким образом, чтобы тройка векторов a b n была правой.

Свойства векторного произведения векторов:

1) Векторное произведение вектора на самого себя даёт в результате нулевой вектор.

2) IMG_257 – порядок умножения имеет значения,свойство называют **антикоммутативностью**.

3) IMG_258 – **ассоциативные** законы векторного произведения. Константы беспроблемно выносятся за пределы векторного произведения.

4) IMG_259 – **дистрибутивные** законы векторного произведения. С раскрытием скобок тоже нет проблем.

Применение векторного произведения в компьютерной графике:

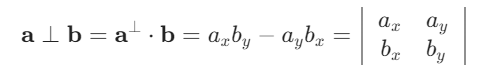
Определение нормали к поверхности: Векторное произведение двух векторов, касающихся поверхности, позволяет найти вектор, перпендикулярный этой поверхности. Это критически важно для правильного освещения и отражения света в 3D-рендеринге, поскольку нормали определяют, как свет должен отражаться от поверхности объекта.

Матрицы преобразования: векторное произведение используется для формирования компонент матрицы для преобразования вершины в другое пространство, в пространство наблюдателя.

Текстурирование: векторное произведение может использоваться для корректного наложения текстур на объект, например при наложении карты нормалей в касательном пространстве, оно используется в TBN-матрице, используемой для поворота векторов карты.

1. **Свойства и применение перпендикулярного скалярного произведения векторов в компьютерной графике.**

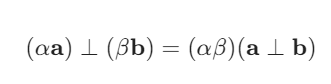
Перпендикулярным скалярным произведением(perp dot product) векторов a и b называется скалярное произведение вектора, перпендикулярного вектору а на вектор b.



Свойства перпендикулярного скалярного произведения:

1. Перпендикулярное скалярное произведение можно представить как произведение длин векторов на синус угла между ними.



1. Перпендикулярное скалярное произведение вектора на самого себя равно 0.
2. Свойство ассоциативности(вынос константного множителя за скобки):
3. Свойство дистрибутивности(раскрытие скобок)



1. Свойство антикоммутативности(от перестановки векторов меняется знак результата)



Применение скалярного произведения в компьютерной графике:

Аппроксимация нормалей в полигоне: перпендикулярное скалярное произведение можно использовать для нахождения барицентрических координат полигона, которые используются уже для аппроксимации нормали в точке на основе нормалей в вершинах полигона.

Расчёт точки пересечения прямых на плоскости: перпендикулярное скалярное произведение может быть полезно при нахождении точки пересечения двух прямых, лежащих на плоскости. Оно может предоставить информацию о том, являются ли прямые параллельными или совпадают.

Кратчайшее направление доворота векторов: перпендикулярное скалярное произведение может использоваться для определения кратчайшего направления доворота одного вектора до другого(например, если оно больше 0, то кратчайший доворот вектора осуществляется против часовой стрелки). Данную информацию можно использовать в оптимизационных целях: заменять расчёт векторного произведения и его направления на перпендикулярное скалярное, требующее меньшее количество расчётов.

1. **Системы координат на плоскости и в пространстве. Прямоугольная система координат.**

**Прямоугольная (декартова) система координат** — прямолинейная система координат с взаимно перпендикулярными координатными осями. Наиболее часто используемая система координат.

В зависимости от количества измерений(2D, 3D) данная система координат состоит из 2 (X,Y) или 3(X,Y,Z) осей.

Точка со значениями каждой компоненты системы равными 0 называется началом координат.

Все прямоугольные системы координат в трёхмерном пространстве делятся на два класса — *правые* и *левые*. Обычно по умолчанию стараются использовать правые координатные системы, а при их графическом изображении ещё и располагают их, если можно, в одном из нескольких обычных (традиционных) положений.

Определить, к какому классу относится какая-либо конкретно взятая система координат, можно, используя правило правой руки. (положительное направление осей выбирают так, чтобы при повороте оси �� против часовой стрелки на 90° её положительное направление совпало с положительным направлением оси��, если этот поворот наблюдать со стороны положительного направления оси ��).

1. **Системы координат на плоскости и в пространстве. Полярная система координат.**

**Полярная система координат** — двумерная [система координат](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0_%D0%BA%D0%BE%D0%BE%D1%80%D0%B4%D0%B8%D0%BD%D0%B0%D1%82" \o "Система координат), в которой каждая точка на плоскости определяется двумя числами — полярным углом и полярным радиусом. Полярная система координат особенно полезна в случаях, когда отношения между точками проще изобразить в виде радиусов и углов.

Полярная система координат задаётся единичным координатным вектором, который называют ***полярной осью***. Точка, из которой выходит этот вектор, называется ***началом координат, или полюсом(О)***. Любая точка на плоскости определяется двумя полярными координатами: ***радиальной(r)*** и ***угловой(ϕ)***. Радиальная координата соответствует расстоянию от точки до начала координат. Угловая координата также называется ***полярным углом и*** равна углу, на который нужно повернуть против часовой стрелки полярную ось для того, чтобы попасть в эту точку.

Связь между полярной и декартовой системами координат(слева выражение компонент ПСК в ДСК, справа - компонент ДСК в ПСК:

1. **Системы координат на плоскости и в пространстве. Цилиндрическая система координат.**

Цилиндрическая система координат -  трёхмерная система координат, являющаяся обобщением полярной системы координат посредством добавления третьей координаты, которая задаёт смещение произвольной точки вдоль оси 0z относительно координатной плоскости 0xy. Положение точки M в цилиндрической системе координат определяется тройкой чисел  ρ,  φ  и  z.

Радиальная координата соответствует расстоянию от точки до начала координат. Угловая координата также называется ***полярным углом*** и равна углу, на который нужно повернуть против часовой стрелки полярную ось для того, чтобы попасть в эту точку. Дополнительная координата z(аппликата) задает расстояние точки до плоскости 0xy.

1. **Системы координат на плоскости и в пространстве. Сферическая система координат.**

**Сферическая система координат** — трёхмерная [система координат](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0_%D0%BA%D0%BE%D0%BE%D1%80%D0%B4%D0%B8%D0%BD%D0%B0%D1%82" \o "Система координат), в которой каждая точка пространства определяется тремя числами: расстоянием до начала координат **(r)**, зенитным**(θ)** и азимутальным**()**углами.

В сферической системе координатные линии, проходящие через любую точку  M  пространства, пересекаются под прямым углом. Такие системы координат называются ортогональными, к ним относится и сферическая соответственно.

1. **Системы координат на плоскости и в пространстве. Барицентрические координаты.**

Барицентрические координаты - это способ описания местоположения точки в пространстве с помощью чисел, которые вместе дают нам уникальное место этой точки. Для этого нужно выбрать несколько точек в пространстве, которые будут служить нашим "маркерами" или "базой". Эти маркеры помогают нам определить, где находится наша точка, используя специальные числовые значения, называемые барицентрическими координатами.

Свойства барицентрических координат:

Барицентрические координаты остаются неизменными при любых афинных преобразованиях: они не изменяются, когда мы меняем перспективу или масштабируем изображение, сохраняя геометрию неизменной.

Каждая точка внутри многоугольника может быть представлена через барицентрические координаты, которые всегда положительны и вместе дают 1. Это позволяет точно указывать позицию любой точки внутри многоугольника. Данное свойство не выполняется при условии, что точка находится за пределами многоугольника.

Если одна из барицентрических координат становится нулевой, это означает, что точка находится на той стороне многоугольника, которая противоположна вершине, соответствующей этой координате.

Для точки внутри многоугольника можно найти значение одной неизвестной компоненты барицентрических координат путём вычитания суммы других компонент из 1.

Используя площади треугольников, можно определить барицентрические координаты для любой точки внутри треугольника. Это даёт нам ещё один способ представления позиции точки внутри треугольника, основанный на геометрии треугольников вокруг этой точки.

1. **Системы координат на плоскости и в пространстве. Однородные координаты.**

**Однородные координаты** ― система координат, используемая в проективной геометрии. Однородным координатам в n-мерном пространстве задаётся n+1 компонентами.

Особенность однородных координат в том, что если мы умножим все четыре числа на одно и то же число (не равно нулю), то проекция полученной точки на исходную систему координат останется такой же. Это значит, что мы можем легко масштабировать или перемещать точки в пространстве, просто изменяя коэффициент, на который умножаем все координаты.(например проекция точек (8,8,4) и (4,4,2) на декартову плоскость будет одинаковой - (2,2)).

Этот метод особенно полезен в компьютерной графике, потому что он позволяет нам легко выполнять различные трансформации и операции над точками и линиями. Например, мы можем легко перевести точки из одной системы координат в другую, или осуществлять работу с перспективой при отображении объекта.

1. **Геометрический конвейер преобразования координат. Преобразование из пространства модели в мировое пространство. Вывод мировой матрицы.**

Процесс преобразования координат происходит в следующей последовательности:

1. Координаты пространства модели преобразовываются в координаты мирового пространства.
2. Координаты мирового пространства преобразовываются в координаты видового пространства(пространства наблюдателя).
3. Координаты видового пространства преобразовываются в координаты пространства проекции.
4. Координаты пространства проекции преобразовываются в координаты пространства окна просмотра.
5. Полученные координаты используются в процессе растеризации.

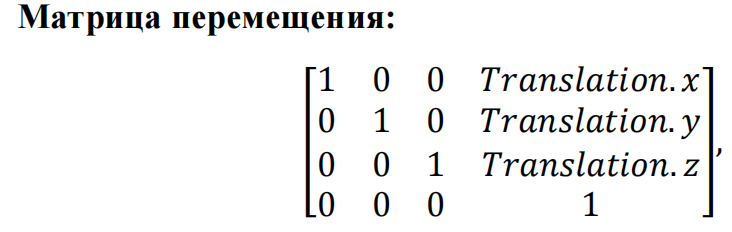
Преобразование из пространства модели в мировое:

Так как прочитанные из файла координаты вершин находятся в про-

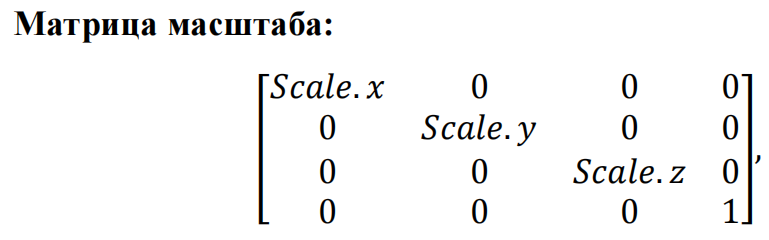
странстве модели, перед отрисовкой их необходимо преобразовать в мировые

координаты. Для этого используются матричные преобразования. Мировое пространство задает расположение объектов на сцене относительно друг друга за счёт масштабирования, поворотов и перемещения.

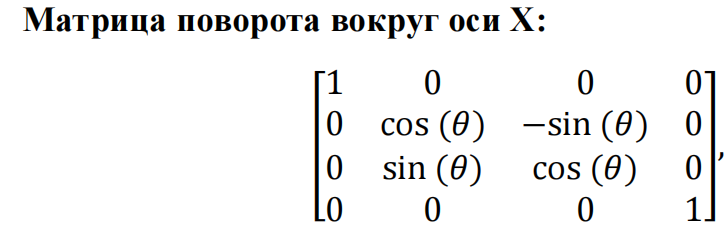
Здесь представлены матриц, используемые для преобразования векторов-столбцов(для векторов-строк необходимы транспонированные версии + вместо умножения матрицы на столбец необходимо умножение строки на матрицу).

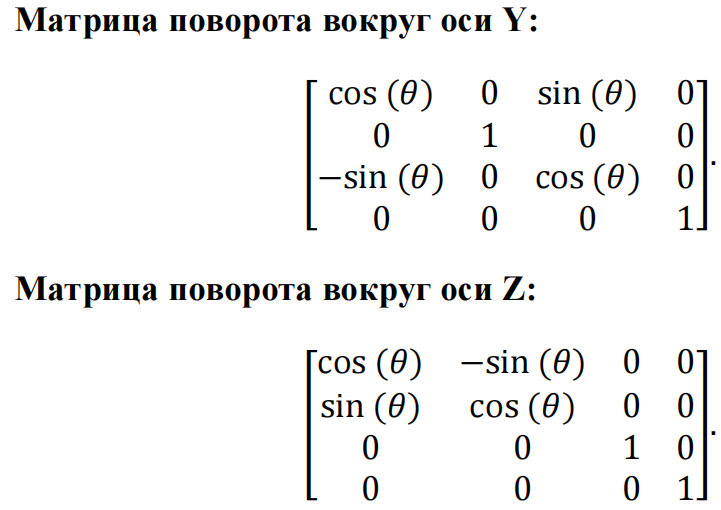


Данная матрица работает таким образом, что при умножении значение координаты корректируется на величину соответствующего сдвига за счет умножения этой величины на w и прибавление этого значения к исходному(которое сохраняется благодаря 1 по диагонали матрицы). Однородная координата w неизменна при использовании этой матрицы.



Данная матрица работает таким образом, что значения координат вершины масштабируются за счёт умножения, таким образом расстояние каждой вершины до «центра» объекта увеличивается, при этом направление прямых сохраняется. Масштабирование должно выполняться до перемещения, иначе перемещение будет учтено при масштабировании, что может сдвигать объект в месте с ним.





Данные матрицы используются для поворота объекта вокруг определённой оси, для объяснения используемой тригонометрии лучше начертить поворот вершины на бумаге и вывести формулы для преобразования исходной вершины в повёрнутую.

1. **Геометрический конвейер преобразования координат. Преобразование из мирового пространства в видовое пространство. Вывод видовой матрицы.**

Процесс преобразования координат происходит в следующей последовательности:

1. Координаты пространства модели преобразовываются в координаты мирового пространства.
2. Координаты мирового пространства преобразовываются в координаты видового пространства(пространства наблюдателя).
3. Координаты видового пространства преобразовываются в координаты пространства проекции.
4. Координаты пространства проекции преобразовываются в координаты пространства окна просмотра.
5. Полученные координаты используются в процессе растеризации.

Преобразование из мирового пространства в видовое:

Пространство наблюдателя — это вспомогательное пространство, идея состоит в том, чтобы расположить все объекты относительно камеры, предполагая проецирование всех

их вершин на экран камеры, который может быть произвольно ориентирован в пространстве. Если необходимо поместить камеру в мировое пространство, нужно использовать матрицу преобразования, которая находится там, где находится камера, и ориентирована так, что ось Z смотрит на цель камеры. Обратное преобразование, применяемое ко всем объектам в мировом пространстве, переместит весь мир в пространство наблюдателя.

Для построения матрицы должны быть заданы следующие величины:

− позиция камеры в мировом пространстве (eye);

− позиция цели, на которую направлена камера (target);

− вектор, направленный вертикально вверх с точки зрения камеры (up).

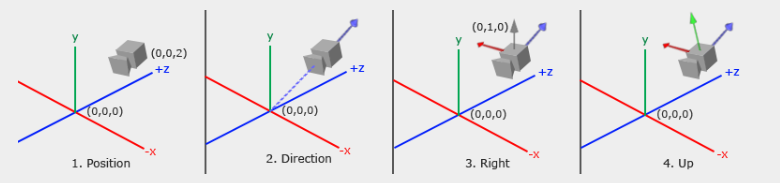
Далее необходимо вычислить базисные векторы камеры:

ZAxis = normalize(eye − target)

XAxis = normalize(ZAxis × up)

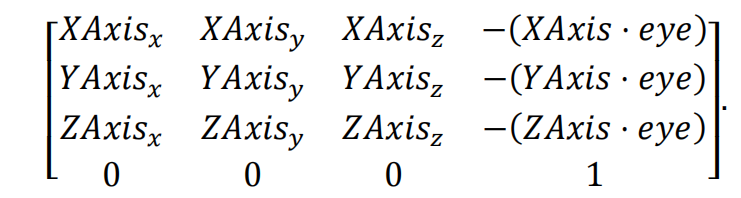
YAxis = up.

Построение осей выглядит примерно так:



Таким образом мы получаем тройку векторов, где ось Z смотрит в камеру, Y смотрит вертикально вверх, X перпендикулярна обеим осям. Для вычисления оси X вектор up можно направить вертикально вверх относительно мирового пространства,так как при векторном произведении ось X в любом случае окажется перпендикулярна Z, а ось Y может быть найдена как векторное произведение осей X,Z.

Матрица трансформации из мирового в видовое пространство имеет следующий вид:



Компоненты осей пространства камеры преобразовывают вершины в новое пространство, скалярное произведение используется для корректировки преобразования с учётом позиции камеры в мировом пространстве, отрицательное значение берётся для перемещения объекта в обратную сторону от направления перемещения камеры.

1. **Геометрический конвейер преобразования координат. Преобразование из видового пространства в пространство проекции. Вывод матрицы ортографической проекции.**

Процесс преобразования координат происходит в следующей последовательности:

1. Координаты пространства модели преобразовываются в координаты мирового пространства.
2. Координаты мирового пространства преобразовываются в координаты видового пространства(пространства наблюдателя).
3. Координаты видового пространства преобразовываются в координаты пространства проекции.
4. Координаты пространства проекции преобразовываются в координаты пространства окна просмотра.
5. Полученные координаты используются в процессе растеризации.

Сцена теперь находится в пространстве, которое удобно проецировать. Все, что теперь нужно сделать, это спроецировать его на воображаемый экран камеры. Прежде чем сделать изображение плоским, необходимо перейти в другое, последнее пространство, пространство проекции. Это пространство является кубоидом, размеры которого находятся между значениями -1 и 1 для осей X и Y и между значениями 0 и 1 для оси Z. Это пространство очень

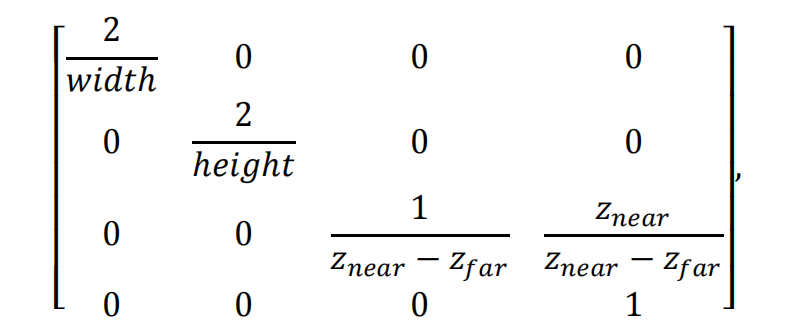
удобно для отсечения (все, что находится за пределами диапазона от -1 до 1 и от 0 до 1, находится за пределами области просмотра камеры) и упрощает операцию проецирования (нужно просто отбросить значение 𝑧, чтобы получить

плоское изображение).

Чтобы перейти из пространства наблюдателя в пространство проецирования, нужна другая матрица — матрица преобразования из пространства наблюдателя в пространство проекции. Значения этой матрицы зависят от того, какой тип проекции необходимо выполнить. Двумя наиболее часто ис-

пользуемыми проекциями являются ортографическая проекция и перспективная проекция.Чтобы сделать ортографическую проекцию, нужно определить размер области, которую может видеть камера. Обычно он задаётся значениями ширины и высоты для оси X и Y и значениями ближнего и дальнего 𝑧 для оси Z.

Учитывая эти значения, можно создать матрицу преобразования, которая преобразует область прямоугольника в кубоид. Следующая матрица преобразует векторы из пространства наблюдателя в пространство ортографической проекции и предполагает правостороннюю систему координат:



Данная матрица работает следующим образом: координаты x y находятся относительно точки (0,0). Для этого достаточно умножить значение компоненты на половину длины/ширины кубоида пространства проекции, так как половина значений отрицательные, а половина положительные для значений обеих осей. Для z все немного сложнее, поскольку необходимо учитывать глубину для однородных z. Вывод строки для вычисления z проще описать на бумаге, но в основе лежит решение системы уравнений, сформированных на основе значения Znear и Zfar, с учётом того, что коэффициент Wnear = 1.

1. **Геометрический конвейер преобразования координат. Преобразование из видового пространства в пространство проекции. Вывод матриц перспективной проекции.**

Процесс преобразования координат происходит в следующей последовательности:

1. Координаты пространства модели преобразовываются в координаты мирового пространства.
2. Координаты мирового пространства преобразовываются в координаты видового пространства(пространства наблюдателя).
3. Координаты видового пространства преобразовываются в координаты пространства проекции.
4. Координаты пространства проекции преобразовываются в координаты пространства окна просмотра.
5. Полученные координаты используются в процессе растеризации.

Сцена теперь находится в пространстве, которое удобно проецировать. Все, что теперь нужно сделать, это спроецировать его на воображаемый экран камеры. Прежде чем сделать изображение плоским, необходимо перейти в другое, последнее пространство, пространство проекции. Это пространство является кубоидом, размеры которого находятся между значениями -1 и 1 для осей X и Y и между значениями 0 и 1 для оси Z. Это пространство очень

удобно для отсечения (все, что находится за пределами диапазона от -1 до 1 и от 0 до 1, находится за пределами области просмотра камеры) и упрощает операцию проецирования (нужно просто отбросить значение z, чтобы получить

плоское изображение).

Чтобы перейти из пространства наблюдателя в пространство проецирования, нужна другая матрица — матрица преобразования из пространства наблюдателя в пространство проекции. Значения этой матрицы зависят от того, какой тип проекции необходимо выполнить. Двумя наиболее часто ис-

пользуемыми проекциями являются ортографическая проекция и перспективная проекция.

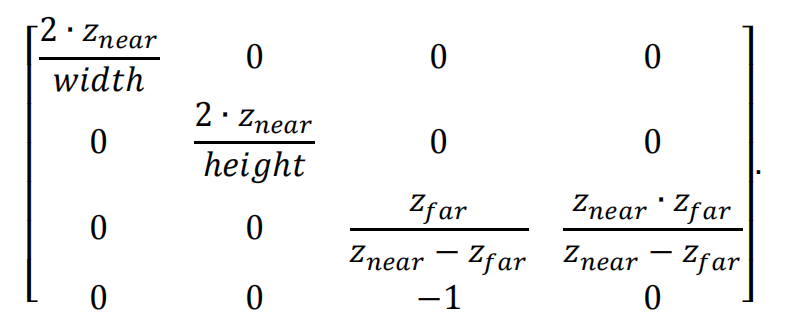
Разберём перспективную проекцию. Идея похожа на

орфографическую проекцию, но на этот раз область просмотра является усечённой пирамидой, и поэтому преобразования немного сложнее. Матричного умножения в этом случае недостаточно, потому что после умножения на матрицу результат не находится в одном и том же пространстве

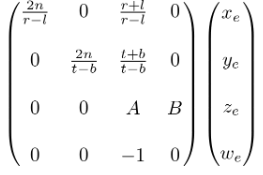
проекции (что означает, что w-компонент не равен 1 для каждой вершины).

Чтобы завершить преобразование, нужно разделить каждую компоненту вектора на компонент w.

Матрица перспективной проекции имеет следующий вид:



При изображении плоскостей оси Z и любой другой по свойствам подоных треугольников получаем, что для пространства проекции с x y необходимы указанные преобразования + деление на Z в дальнейшем. Для формирования нового значения Z необходимо определить порядок его формирования: пусть есть некоторые коэффициенты A и B, определяющие корректировку значения Z на основе него и значения коэффициента W.



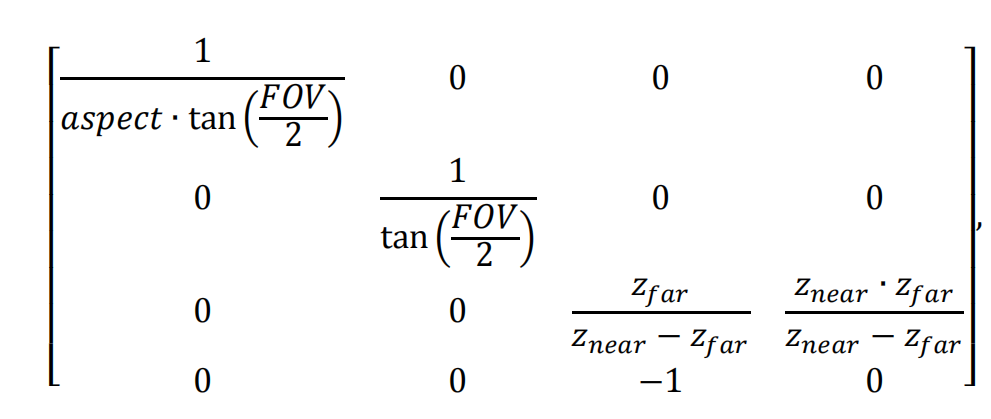
Скорректированное Z имеет примерно следующий вид:



При этом We = 1 так как оно находится в мировом пространстве.

Таким образом можно составить систему уравнений на основе Znear и Zfar, и получить значения А и В решением системы.

Также существует другая форма матрицы перспективной проекции, использующая угол обзора камеры:

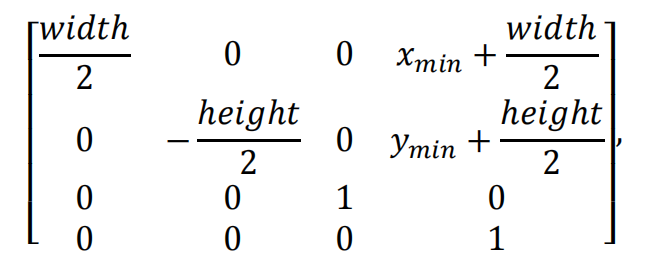


Компоненты матрицы выводятся аналогично с первым её вариантом, за исключением того, что коэффициенты коррекции координат x и y выводятся через угол обзора камеры.

1. **Геометрический конвейер преобразования координат. Преобразование из пространства проекции в экранное пространство. Вывод матрицы окна просмотра.**

Окно просмотра — это область в окне приложения, в которой отображается визуализируемая сцена. Оно может занимать всю область окна приложения или его часть.

Матрица преобразования из пространства проекции в пространство окна просмотра выглядит следующим образом:



Данная матрица работает следующим образом: размеры проекции адаптируются под размеры окна просмотра, нормализованные координаты соотносятся с размерами окна просмотра, в центр окна просмотра перемещается начало координат, экранная ось Y направлена вниз, поэтому для пространства проекции её она переворачивается

1. **Растеризация отрезков. Алгоритм ЦДА.**

Растеризация отрезков — это процесс преобразования геометрических объектов, таких как линии, отрезки, круги и другие фигуры, в растровые изображения. Растровые изображения состоят из пикселей, каждый из которых имеет свой собственный цвет и позицию на экране. Цель растеризации — определить, какой цвет и позицию должен иметь каждый пиксель, чтобы наиболее точно воспроизвести исходный геометрический объект.

Процесс растеризации отрезков включает в себя несколько основных этапов:

Определение цвета каждого пикселя: На этом этапе алгоритм решает, какой цвет должен иметь каждый пиксель вдоль отрезка. Это может быть выполнено различными способами, включая интерполяцию цвета вдоль отрезка или использование предопределённых правил для определения цвета.

Преобразование геометрического объекта в растровое изображение: После того как цвет каждого пикселя был определён, алгоритм строит растровое изображение, помещая каждый пиксель в соответствующее место на экране.

Иногда оба этапа выполняются параллельно, в этом случае цвет каждого пикселя определяется в процессе преобразования объекта в растровое изображение.

Для преобразования геометрического объекта в изображение используются различные алгоритмы. Одним из таких является алгоритм ЦДА-линий(DDA-линий)(ЦДА - это цифровой дифференциальный анализатор). Данный алгоритм используется для построения линии между двумя концами отрезка и работает в вещественных числах. Работу алгоритма можно описать следующим образом:

Пусть отрезок задан вещественными координатами концов **(x1,y1)**; **(x2,y2)**. Растровыми (целочисленными) координатами концевых точек становятся округлённые значения исходных координат:

**xstart=round(x1)**, **ystart=round(x2)**;

**xend=round(x2)**, **yend=round(y2)**.

Большее из двух чисел — **(xend - xstart)** или **(yend - ystart)** — по абсолютной величине принимается за количество шагов L цикла растеризации, увеличенное на 1.

В начале цикла вспомогательным вещественным переменным x и u присваиваются исходные координаты начала отрезка: **x=x1**; **y=y1**. На каждом шаге цикла эти вещественные переменные получают приращения **(xend - xstart)/L**; **(yend - ystart)/L**. Растровые же координаты, продуцируемые на каждом шаге, являются результатом округления соответствующих вещественных значений **x** и **y**.

Применение вычислений с вещественными числами и лишь однократное использование округления для окончательного получения значения растровой координаты делают данный алгоритм очень точным в отношении координат точек на экране, но в то же время очень медленным.

Также необходимо учесть направление возрастания координат точек. Таким образом, необходимо переставить значения х и у точек начала и конца таким образом, чтобы значения х и у возрастали от точки начала отрезка до точки конца отрезка.

1. **Растеризация отрезков. Алгоритм Брезенхема.**

Растеризация отрезков — это процесс преобразования геометрических объектов, таких как линии, отрезки, круги и другие фигуры, в растровые изображения. Растровые изображения состоят из пикселей, каждый из которых имеет свой собственный цвет и позицию на экране. Цель растеризации — определить, какой цвет и позицию должен иметь каждый пиксель, чтобы наиболее точно воспроизвести исходный геометрический объект.

Процесс растеризации отрезков включает в себя несколько основных этапов:

Определение цвета каждого пикселя: На этом этапе алгоритм решает, какой цвет должен иметь каждый пиксель вдоль отрезка. Это может быть выполнено различными способами, включая интерполяцию цвета вдоль отрезка или использование предопределённых правил для определения цвета.

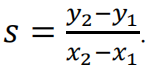
Преобразование геометрического объекта в растровое изображение: После того как цвет каждого пикселя был определён, алгоритм строит растровое изображение, помещая каждый пиксель в соответствующее место на экране.

Иногда оба этапа выполняются параллельно, в этом случае цвет каждого пикселя определяется в процессе преобразования объекта в растровое изображение.

Для преобразования геометрического объекта в изображение используются различные алгоритмы. Одним из таких алгоритмов является алгоритм Брезенхема. Данный [алгоритм](https://ru.wikipedia.org/wiki/%D0%90%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC" \o ") определяет какие точки [экрана нужно закрасить, чтобы получить близкое приближение прямой линии между двумя заданными точками](https://ru.wikipedia.org/wiki/%D0%90%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC%D1%8B_%D0%BF%D0%BE%D1%81%D1%82%D1%80%D0%BE%D0%B5%D0%BD%D0%B8%D1%8F_%D0%BE%D1%82%D1%80%D0%B5%D0%B7%D0%BA%D0%B0" \o "Алгоритмы построения отрезка). Его работу можно описать следующим образом:

На каждом шаге

цикла растеризации происходит одно из двух: значение y остаётся неизменным либо увеличивается на 1. Выбор производится на основе значения ошибки, которое означает вертикальное расстояние между текущим значением y и точным значением функции для текущего х. Всякий раз, когда мы увеличиваем х, мы увеличиваем значение ошибки на величину наклона:



Если ошибка превысила 0.5, линия стала ближе к следующему

у, поэтому нужно увеличить текущее значение y на единицу, а значение ошибки уменьшить на 1.

В данном алгоритме значение ошибки, величина наклона s, а также константа 0.5 являются числами с плавающей запятой. Можно ускорить вычисления,для этого необходимо перейти к целым числам. Это можно сделать следующим образом:

1)Домножить все используемые вещественные величины на

Тогда на каждом шаге цикла значение ошибки будет увеличиваться на а при увеличении у на 1 значение ошибки будет уменьшаться на .

2)Чтобы избавиться от константы 0.5, необходимо умножить обе части неравенства проверки коррекции ошибки на 2, тогда вместо

*error* > ( / 2 получится 2 × *error* > ().

Также необходимо учесть, угол наклона линии и направление возрастания координат точек. Таким образом, необходимо переставить значения х и у точек начала и конца таким образом, чтобы х имел больше шагов изменения значения, чем у, а также чтобы значения х и у возрастали от точки начала отрезка до точки конца отрезка.

1. **Кривые и сплайны Безье. Алгоритм де Кастельжо.**

Сплайн Безье — это кривая, определяемая двумя наборами точек: две из них это конечные точки и контрольные точки. Кривая начинается с первой из конечных точек и заканчивается во второй конечной точке. Кривая не проходит через контрольные точки, но контрольные точки действуют как магниты, вытягивая кривую в определённых направлениях и влияя на то, как кривая изгибается.

Существуют следующие виды кривых Безье:

### **Линейные кривые**



Кривая представляет собой отрезок прямой линии.

**Квадратичные кривые**



Кривая представляет собой параболу.

**Кубические кривые**



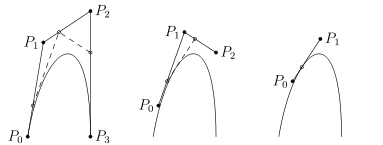
Линия берёт начало из первой конечной точки, направляясь к первой контрольной точке и заканчивается во второй конечной точке, подходя к ней со стороны второй контрольной точки. То есть, кривая не проходит через Длина отрезка между первыми конечной и контрольной точками определяет, как скоро кривая повернёт ко второй контрольной точке.

**Алгоритм де Кастельжо** является алгоритмом для разделения кривой Безье на две части по произвольному значению параметра t, то делает его мощным инструментом для работы с кривыми Безье в компьютерной графике.

Основная идея алгоритма заключается в том, что для определения точки на кривой Безье по параметру (t) можно использовать рекурсивный процесс.

Для начала, алгоритм проверяет, является ли текущая точка конечной точкой кривой. Если это так, то возвращается сама точка. В противном случае, алгоритм рекурсивно вызывается для каждой из двух половин кривой, соответствующих двум соседним контрольным точкам, и затем используется формула для интерполяции между этими двумя точками, чтобы получить итоговую точку на кривой.

Графически алгоритм де Кастельжо можно представить в следующем виде:



Достоинством алгоритма де Кастельжо является его высокая вычислительная устойчивость. Это означает, что даже при больших значениях параметра (t) алгоритм может быстро и точно вычислить точку на кривой Безье.

1. **Растеризация треугольников. Алгоритм сканирующей линии.**

Растеризация треугольников

Растеризация треугольника — это процесс определения, какие пиксели треугольника должны быть окрашены в растровом изображении с последующим его заполнением.

Для растеризации треугольников широко используется алгоритм сканирующей линии(scanline), который работает следующим образом:

1. Вершины треугольника располагают таким образом, чтобы начальная вершина была самой верхней(с наименьшим значением у), из оставшихся выбирается средняя и нижняя.
2. От у0 до у1 двигаем горизонтальную линию, заполняя с ее помощью пространство между точками граней треугольника.
3. Теперь двигаем линию от у1 до у2, при этом сторона (у0;у2) продолжает использоваться, при движении зарисовываем пространство внутри треугольника между точками его граней.

В качестве алгоритмов для заполнения пространства внутри треугольника можно использовать алгоритмы растеризации отрезков: ЦДА-линии или алгоритм Брезенхема.

1. **Растеризация треугольников. Алгоритм, основанный на барицентрических координатах.**

Растеризация треугольников

Растеризация треугольника — это процесс определения, какие пиксели треугольника должны быть окрашены в растровом изображении с последующим его заполнением.

Для растеризации треугольников широко используется алгоритм сканирующей линии(scanline), однако существует альтернативный вариант алгоритма, использующий барицентрические координаты для растеризации.

Барицентрические координаты однозначно описывают положение точки в пространстве относительно некоторой базы, являющейся набором опорных точек. В данном случае базой будут являться вершины треугольника, а условием принадлежности точки треугольнику будет нахождение значения все барицентрических координат в пределах от 0 до 1.

Сам алгоритм работает следующим образом:

1)В цикле перемещаемся по координатной плоскости (по пикселям) построчно слева направо.

2) Вычисляются барицентрические координаты текущей точки относительно вершин текущего треугольника.

3) Выполняется проверка условия принадлежности точки треугольнику.

4) Если условие верно, то точка принадлежит треугольнику и её необходимо рисовать на экране, дополнительно при использовании цветов значению её цвета присваивается цвет, полученный при интерполяции цвет вершин этого треугольника, после этого точка выводится на экран. При некорректности условия возвращаемся к шагу 1.

Данный алгоритм можно оптимизировать, если рассчитать приращения при изменении координат точки для каждой компоненты барицентрических координат. Тогда после расчёта барицентрических координат начальной точки, можно просто корректировать значения координат на основе приращения, что будет быстрее расчёта этих же координат с нуля для каждой точки.

1. **Алгоритмы отбраковки нелицевых граней трёхмерных объектов.**

Алгоритмы отбраковки нелицевых граней трёхмерных объектов играют ключевую роль в оптимизации процесса рендеринга и визуализации сцен.

Они позволяют исключить из процесса рендеринга те части объектов, которые не видны из точки наблюдения, что существенно увеличивает производительность.

Одним из широко используемых алгоритмов можно назвать алгоритм отбраковки задних граней(backface culling).

Данный алгоритм можно реализовать 2-мя способами:

В первом способе необходимо найти скалярное произведение вектора взгляда и нормали к поверхности полигона, если скалярное произведение отрицательное или равно 0, данный полигон является видимым и его необходимо отрисовывать, делать этого в противном случае не нужно.

Во втором способе необходимо определить расположение вершин треугольника в пространстве экрана. Для этого выберем грань треугольника, которая соединяется с концом наиболее длинной грани треугольника, если кратчайший ее доворот до оставшейся грани осуществляется против часовой стрелки, то треугольник повернут к нам лицевой стороной, иначе нелицевой. Для определения расположения можно использовать перпендикулярное скалярное произведение, треугольник повернут лицевой если оно положительное, иначе он повернут нелицевой стороной.

1. **Алгоритм удаления невидимых поверхностей с использованием Z-буфера.**

Алгоритм удаления невидимых поверхностей с использованием z-буфера — это один из простейших и наиболее эффективных методов в компьютерной графике для оптимизации процесса рендеринга 3D-сцен.

Этот алгоритм работает непосредственно в пространстве изображения и достаточно прост для понимания и использования.

Данный алгоритм работает следующим образом:

Алгоритм основан на идее хранения информации о глубине (z-значении) каждого пикселя в отдельном буфере (z-буфере). Когда объекты рендерятся на экране, их z-значения сравниваются с уже имеющимися значениями в z-буфере. Если новый объект находится дальше от камеры (имеет большее z-значение), то он считается невидимым и пропускается.

Если объект ближе к камере (имеет меньшее z-значение), то его информация записывается в z-буфере, заменяя старые данные.

Данный алгоритм прост в реализации и понимании, имеет линейную вычислительную сложность: не требуется предварительной сортировки объектов по глубине, что сокращает время обработки, в целом он подходит для сцен любой сложности и позволяет легко визуализировать пересечения сложных поверхностей.

Данный алгоритм можно считать практически не имеющим недостатков. Его недостатки заключаются в потреблении памяти из-за необходимости хранения z-значений для каждого пикселя и лестничного эффекте(z-fighting). Первая проблема на сегодняшний день уже не актуальна, несколько килобайт или даже мегабайт на буфер не является запредельным объёмом памяти, лестничный эффект сохраняется, однако существует несколько решений этой проблемы:

1. Отдалить один из объектов, тогда разница в глубине увеличится и количество артефактов уменьшится вплоть до нуля.
2. Увеличить расстояние Znear от камеры, это приведёт к более резкому изменению значения глубины и соответственно значения z будут различаться сильнее и их легче будет сравнивать.
3. **Модели затенения трёхмерных объектов. Плоское затенение.**

Модель затенения трёхмерных объектов реализована во фрагментном шейдере и определяет как осуществляется формирование цвета в отдельных пикселях для имитации освещения.

Наиболее простой моделью затенения является модель плоского затенения.

Идея алгоритма плоского затенения довольна простая. Сперва цвет вычисляется в каждой вершине треугольника, затем полученные значения усредняются и весь треугольник закрашивается в полученный цвет.

Данная модель обладает высокой скоростью работы, но на визуализированной модели чётко заметны переходы между гранями.

Даже существенное увеличение количества треугольников не позволяет скрыть резкие переходы между ними.

1. **Модели затенения трёхмерных объектов. Затенение Гуро.**

Модель затенения трёхмерных объектов реализована во фрагментном шейдере и определяет как осуществляется формирование цвета в отдельных пикселях для имитации освещения.

Наиболее простой моделью затенения является модель плоского затенения, однако она даёт крайне полигональный объект в конечном итоге. Для получения более реалистичного объекта при хорошей вычислительной стоимости можно использовать модель затенения по Гуро.

В этой модели цвет не усредняется для грани, а линейно интерполируется между вершинами.

С другой стороны, вычислительная стоимость затенения по Гуро остаётся приемлемой, т.к. дорогостоящий расчёт освещения по-прежнему осуществляется в вершинах, а линейную интерполяцию можно хорошо оптимизировать.

К сожалению, модель затенения по Гуро не безупречна. Блики на освещаемой поверхности при невысоком уровне детализации будут смазаны или могут вовсе "потеряться", если находятся в центре полигона.

1. **Модели затенения трёхмерных объектов. Затенение Фонга.**

Модель затенения трёхмерных объектов реализована во фрагментном шейдере и определяет как осуществляется формирование цвета в отдельных пикселях для имитации освещения.

Наиболее простой моделью затенения является модель плоского затенения, однако она даёт крайне полигональный объект в конечном итоге. Для получения более реалистичного объекта при хорошей вычислительной стоимости можно использовать модель затенения по Гуро.

Однако, затенение по Гуро также имеет недостатки по имитации освещения, особенно в отношении бликов. Для получения наиболее визуально-приятной имитации света используется затенение по Фонгу.

В этой модели между вершинами интерполируется не цвет, а нормаль.

Цвет, в свою очередь, рассчитывается для каждого пикселя в отдельности.

При использовании затенения по Фонгу изображение получается гораздо более качественным, чем при использовании других моделей затенения, и исчезает проблема с бликами. Но данный алгоритм требует гораздо больше вычислительных ресурсов даже по сравнению с затенением по Гуро.

1. **Модели освещения трёхмерных объектов. Освещение Ламберта.**

Модель освещения трёхмерных объектов реализована в вершинном шейдере и определяет способ расчёта освещения для его имитации.

Наиболее простой моделью освещения является модель освещения Ламберта. Идея данной модели заключается в том, что свет равномерно распределяется по объекту в зависимости от направления свечения. Таким образом чем ближе точка объекта в точке пересечения вектора света с ним, тем более интенсивно она освещена и тем более светлой является.

Для имитации освещения по модели Ламберта рассчитывается нормаль от поверхности полигона, которая при скалярном перемножении с вектором света даёт интенсивность освещения данного полигона. Чем больше скалярное произведение, тем интенсивнее освещается полигон объекта.

1. **Модели освещения трёхмерных объектов. Освещение Фонга.**

Модель освещения трёхмерных объектов реализована в вершинном шейдере и определяет способ расчёта освещения для его имитации.

Наиболее простой моделью освещения является модель освещения Ламберта, однако для более реалистичного освещения больше подходит модель освещения Фонга. Идея модели освещения Фонга заключается в следующем:

Пусть заданы точечный источник света, расположенный в некоторой точке, поверхность, которая будет освещаться и наблюдатель. Будем считать, что наблюдатель тоже точечный.

Каждая точка поверхности имеет свои координаты и в ней определена нормаль к поверхности. Её освещённость складывается из трёх компонент: фоновое освещение (ambient), рассеянный свет (diffuse) и бликовая составляющая (specular). Свойства источника определяют мощность излучения для каждой из этих компонент, а свойства материала поверхности определяют её способность воспринимать каждый вид освещения. Разберём отдельные компоненты освещения:

Фоновое освещение это постоянная в каждой точке величина надбавки к освещению. Фоновая составляющая освещённости не зависит от пространственных координат освещаемой точки и источника. Поэтому при моделировании освещения, в большинстве случае, не имеет смысла брать более одного фонового источника света. Часто просто задаётся некое глобальное фоновое освещение всей сцены.

Рассеянный свет при попадании на поверхность рассеивается равномерно во все стороны. При расчёте такого освещения учитывается только ориентация поверхности (нормаль) и направление на источник света(по сути модель освещения Ламберта).

Зеркальный свет при попадании на поверхность подчиняется следующему закону: “Падающий и отражённый лучи лежат в одной плоскости с нормалью к отражающей поверхности в точке падения, и эта нормаль делит угол между лучами на две равные части”. Отражённая составляющая освещённости в точке зависит от того, насколько близки направления на наблюдателя и отражённого луча.

Именно зеркальное отражение представляет наибольший интерес, но в то же время его расчёт требует больших вычислительных затрат. При фиксированном положении поверхности относительно источников света фоновая и рассеянные составляющие освещения могут быть просчитаны единожды для всей сцены, т.к. их значение не зависит от направления взгляда. С зеркальной составляющей этот фокус не сработает и придётся пересчитывать её каждый раз, когда взгляд меняет своё направление.

1. **Модели освещения трёхмерных объектов. Освещение Блинна-Фонга.**

Модель освещения трёхмерных объектов реализована в вершинном шейдере и определяет способ расчёта освещения для его имитации.

Наиболее приближенной к реальному освещению(без учета PBR, physically based rendering) является модель освещения Блинна-Фонга.

Идея модели освещения Блинна-Фонга заключается в следующем:

Пусть заданы точечный источник света, расположенный в некоторой точке, поверхность, которая будет освещаться и наблюдатель. Будем считать, что наблюдатель тоже точечный.

Каждая точка поверхности имеет свои координаты и в ней определена нормаль к поверхности. Её освещённость складывается из трёх компонент: фоновое освещение (ambient), рассеянный свет (diffuse) и бликовая составляющая (specular). Свойства источника определяют мощность излучения для каждой из этих компонент, а свойства материала поверхности определяют её способность воспринимать каждый вид освещения. Разберём отдельные компоненты освещения:

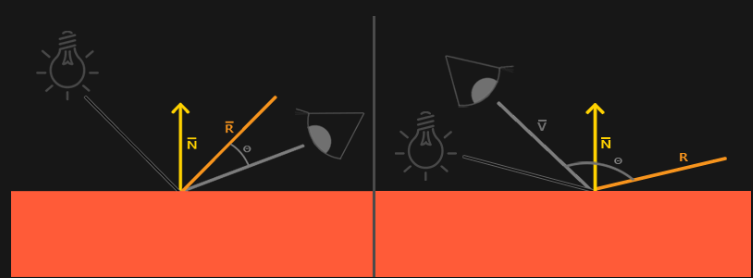
Фоновое освещение это постоянная в каждой точке величина надбавки к освещению. Фоновая составляющая освещённости не зависит от пространственных координат освещаемой точки и источника. Поэтому при моделировании освещения, в большинстве случае, не имеет смысла брать более одного фонового источника света. Часто просто задаётся некое глобальное фоновое освещение всей сцены.

Рассеянный свет при попадании на поверхность рассеивается равномерно во все стороны. При расчёте такого освещения учитывается только ориентация поверхности (нормаль) и направление на источник света(по сути модель освещения Ламберта).

Зеркальный свет при попадании на поверхность подчиняется следующему закону: “Падающий и отражённый лучи лежат в одной плоскости с нормалью к отражающей поверхности в точке падения, и эта нормаль делит угол между лучами на две равные части”. Отражённая составляющая освещённости в точке зависит от того, насколько близки направления на наблюдателя и отражённого луча.

Именно зеркальное отражение представляет наибольший интерес, но в то же время его расчёт требует больших вычислительных затрат. При фиксированном положении поверхности относительно источников света фоновая и рассеянные составляющие освещения могут быть просчитаны единожды для всей сцены, т.к. их значение не зависит от направления взгляда. С зеркальной составляющей этот фокус не сработает и придётся пересчитывать её каждый раз, когда взгляд меняет своё направление.

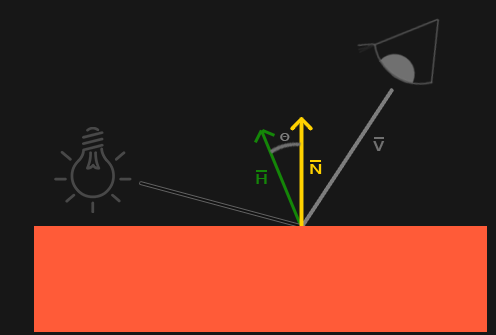
Проблема модели освещения Фонга заключается именно в вычислении зеркальной компоненты. Данную проблему хорошо можно продемонстрировать на следующем примере:



Слева мы видим привычную нам картину Фонговского отражения с θ(углом между вектором взгляда и отражённым лучом света) менее 90 градусов. На правом рисунке угол θ между направлениями зрения и отражения больше 90 градусов, в результате чего вклад зеркального освещения аннулируется. Обычно эта проблема очень незначительна, так как вектор обзора зачастую заметно удален от вектора отражения. Но при малых значениях силы зеркального блеска радиус области отражения становится достаточно большим, и даёт заметный вклад в общую картину. Используя модель Фонга мы аннулируем этот вклад при углах больше 90 градусов. На следующей картинке более заметна проблема модели освещения Фонга(слева):



Модель освещения Блинна-Фонга является дополнением к модели Фонга. Она использует немного иной подход к расчёту зеркальной компоненты, что позволяет решить проблему. Вместо того, чтобы полагаться на вектор отражения, мы используем так называемый медианный вектор (halfway vector), который представляет из себя единичный вектор точно посередине между направлением обзора и направлением света. Чем ближе этот вектор к нормали поверхности, тем больше будет вклад зеркальной компоненты. Это можно увидеть на картинке:



Когда направление обзора полностью совпадает с (теперь уже мнимым) вектором отражения, медианный вектор совпадает с нормалью к поверхности. Таким образом, чем ближе направление обзора к направлению отражения, тем сильнее становится зеркальный блеск.

Очевидно, что вне зависимости от направления, с которого смотрит наблюдатель, угол между медианным вектором и нормалью к поверхности никогда не превысит 90 градусов (если, конечно, источник света не находится ниже поверхности). Благодаря этому картина выглядит более визуально правдоподобной(сравнение есть на позапрошлой картинке), особенно при низких значениях силы зеркального блеска.

1. **Модели освещения трёхмерных объектов. Основные типы источников освещения.**

Модель освещения трёхмерных объектов реализована в вершинном шейдере и определяет способ расчёта освещения для его имитации.

Существует 3 основных типа источника освещения: источник направленного света, источник точечного света и spot light источник(прожектор).

Разберём чуть подробнее каждый из них:

**Источник направленного света**

Источник направленного света находится в бесконечно-удалённой точке. В этом случае допустимо считать, что все лучи света от него распространяются параллельно и для всех точек можно использовать один и тот же вектор направления освещения. Хорошим примером такого источника света является Солнце.

Для источников направленного света использование модели Блинна-Фонга оказывается весьма выгодным. В предположении о постоянстве вектора взгляда, вектор полупути может быть вычислен единожды для источника и, далее использоваться на протяжении всех вычислений. В этом случае расчёт зеркальной компоненты освещённости будет почти так же прост, как для рассеянной составляющей.

**Точечный источник света**

В отличие от источников направленного света, точечные источники находятся в определённой точке пространства с конечными координатами, и свет от них распространяется равномерно по всем направлениям. При расчёте освещённости в точке будет учитываться направление на такой источник.

Также для точечных источников учитывается эффект поглощения света, когда интенсивность излучения уменьшается с расстоянием.

**Прожекторы (spot light sources)**

Такие источники света являются простой имитацией реальных прожекторов, которые используются, к примеру, в театре для создания мощного луча света в определённом направлении. Свет от такого источника распространяется лишь в определённом направлении и получается конус света. Обеспечивается это с помощью наложения ограничений на вектор направления \vec{L} от освещаемой точки до источника света. У прожекторов, по сравнению с точечными источниками света добавляются два новых параметра: направление центрального освещения, половинный угол отсечения φ и коэффициент поглощения. Угол отсечения φ Є (0; 90°). Если φ = 90°, то прожектор превращается в точечный источник света.

1. **Текстурирование трёхмерных объектов. Наложение текстур с перспективной коррекцией. Методы повторения текстур.**

Текстурирование объектов в компьютерной графике — это процесс добавления деталей на поверхность объекта, что позволяет ему выглядеть более реалистично и интересно. Основная идея текстурирования схожа с применением обоев к стенам: сложный узор наносится на плоскую бумагу, а затем эта бумага накладывается на поверхность стены. В компьютерной графике такой же принцип применяется к объектам: узоры или изображения наносятся на их поверхности, улучшая визуальное восприятие и добавляя глубину и детальность.

Для применения текстуры к объекту необходимо использовать координаты текстуры (также известные как UV-координаты). Эти координаты представляют собой систему координат, в которой каждая вершина объекта имеет пару чисел, указывающих на местоположение в текстуре. Координата U горизонтальная и возрастает слева направо, V - вертикальная, возрастает снизу вверх(в экранной плоскости обычно наоборот, стоит это учитывать).

Текстура объекта обычно задается набором текстурных карт. Текстурная карта — это двухмерное изображение, которое накладывается на полигональную сетку модели для придания ей больше выразительности, добавления цвета или создания иллюзии реалистичности. Существует несколько основных используемых текстурных карт:

1. Диффузная карта: задает цвет каждого пикселя на поверхности объекта.
2. Карта нормалей: задает нормаль к поверхности в точках объекта.

Нормаль хранится в файле в виде цвета, нормализованная компонента которого изменяется от 0 до 1, для перехода к нормализованной компненте нормали, изменяющейся от -1 до 1 необходимо выполнить 2 действия : умножить компоненту на 2(получим диапазон от 0 до 2) и вычесть 1(получим искомый диапазон от -1 до 1). Выполнив действия в обратном порядке, можно трансформировать нормаль в цвет.

Существует две разновидности карт нормалей: карты нормалей, векторы которой находятся в пространстве модели, и карты

нормалей, у которой все векторы находятся в пространстве плоскости поли-

гона (в касательном пространстве). В касательном пространстве все нормали смотрят в положительном направлении оси Z, ко-

торая равна (0, 0, 1). Карта нормалей в касательном пространстве

очень подходит для создания анимации, т.к. для каждого состояния объекта потребовалась бы отдельная карта нормалей в пространстве модели.

Однако существует одна проблема, которая сильно ограничивает ис-

пользование карт нормалей в касательном пространстве. Требуется преобразовать нормали таким образом, чтобы полученные векторы смотрели в том же

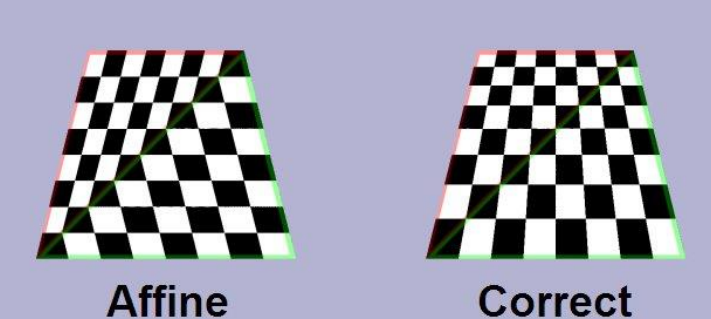
направлении, что и плоскость полигона (т.е. нужно перейти из касательного пространства в мировое пространство). Другое решение заключается в том, чтобы перенести освещение и камеру в касательное пространство, а векторы нормалей оставить неизменными. Таким образом, можно использовать одну и

ту же карту нормалей, независимо от ориентации объекта.

1. Зеркальная карта: задает способность отдельных точек объекта отражать свет. В большинстве случаев данная текстура является черно-белой модификацией диффузной карты. Каждый тексель зеркальной карты содержит коэффициент зеркального освещения для соответствующего фрагмента поверхности.

**Перспективная коррекция текстур**

Аффинное наложение текстур линейно интерполирует текстурные координаты по поверхности и является самым быстрым методом текстурирования. Поскольку данный способ не учитывает информацию о глубине фрагмента, когда поверхность расположена под углом к наблюдателю, можно заметить искажение текстуры. Выглядеть оно будет примерно так:



Чтобы выполнить перспективную коррекцию текстурных координат u и v, причем что z - компонент глубины зависит от точки зрения зрителя, мы можем воспользоваться тем фактом, что значения 1/z, u/z и v/z линейны в пространстве экрана по всей текстурируемой поверхности. В отличие от этого, исходные значения z, u и v до деления не являются линейными в пространстве экрана. Поэтому мы можем линейно интерполировать эти взаимно обратные величины по поверхности, вычисляя скорректированные значения в каждом пикселе, чтобы в результате получить текстурное отображение, корректное с точки зрения перспективы.

Для этого мы сначала вычисляем взаимные коэффициенты в каждой вершине нашей геометрии (3 точки для треугольника). Для вершины n мы имеем un / zn,vn / zn,1 / zn. Затем мы линейно интерполируем эти взаимности между n вершинами (например, используя барицентрические координаты), в результате чего получаем интерполированные значения по всей поверхности. В заданной точке это дает интерполированные значения ui, vi и zReciprocal\_i=1 / zi. Обратите внимание, что эти ui,vi пока нельзя использовать в качестве координат текстуры, так как деление на z изменило их систему координат.

Для коррекции обратно в пространство u,v мы сначала вычисляем скорректированный z, снова взяв обратную величину z\_correct=1 / zReciprocal\_i=1 / 1 / zi = zi. Затем мы используем это для коррекции ui,vi : u\_correct=ui/zi и v\_correct=vi / zi.

**Методы повторения текстур**

Существует множество методов повторения текстур. Наиболее примитивным решение можно считать простое копирование текстуры в пределах текстурной карты. Проблема такого метода заключается в ощутимом расходе памяти, особенно для текстур с высоким разрешением и большим числом повторений.

Решением данной проблемы является использование простого метода повторения текстур, основанного на взятии остатка от деления для текструных координат. Для текстурных координат вне диапазона от 0 до 1 необхлодимо брать остаток от деления по следующей формуле:

, при условии что а положительное.

, при условии что а отрицательное.

Таким образом текстурная координата для координаты -0.5 будет -0.5 % 1 = 1 + (- 0.5) = 0.5. При этом текстура будет повторяться для координат -1.5, -2.5 и так далее. Поскольку данный подход применим к текстуре в целом, его можно использовать для эффективного повторения текстур.

1. **Текстурирование трёхмерных объектов. Наложение текстур с перспективной**

**коррекцией. Точечная и билинейная фильтрация текстур.**

Текстурирование объектов в компьютерной графике — это процесс добавления деталей на поверхность объекта, что позволяет ему выглядеть более реалистично и интересно. Основная идея текстурирования схожа с применением обоев к стенам: сложный узор наносится на плоскую бумагу, а затем эта бумага накладывается на поверхность стены. В компьютерной графике такой же принцип применяется к объектам: узоры или изображения наносятся на их поверхности, улучшая визуальное восприятие и добавляя глубину и детальность.

Для применения текстуры к объекту необходимо использовать координаты текстуры (также известные как UV-координаты). Эти координаты представляют собой систему координат, в которой каждая вершина объекта имеет пару чисел, указывающих на местоположение в текстуре. Координата U горизонтальная и возрастает слева направо, V - вертикальная, возрастает снизу вверх(в экранной плоскости обычно наоборот, стоит это учитывать).

Текстура объекта обычно задается набором текстурных карт. Текстурная карта — это двухмерное изображение, которое накладывается на полигональную сетку модели для придания ей больше выразительности, добавления цвета или создания иллюзии реалистичности. Существует несколько основных используемых текстурных карт:

1. Диффузная карта: задает цвет каждого пикселя на поверхности объекта.
2. Карта нормалей: задает нормаль к поверхности в точках объекта.

Нормаль хранится в файле в виде цвета, нормализованная компонента которого изменяется от 0 до 1, для перехода к нормализованной компненте нормали, изменяющейся от -1 до 1 необходимо выполнить 2 действия : умножить компоненту на 2(получим диапазон от 0 до 2) и вычесть 1(получим искомый диапазон от -1 до 1). Выполнив действия в обратном порядке, можно трансформировать нормаль в цвет.

Существует две разновидности карт нормалей: карты нормалей, векторы которой находятся в пространстве модели, и карты

нормалей, у которой все векторы находятся в пространстве плоскости поли-

гона (в касательном пространстве). В касательном пространстве все нормали смотрят в положительном направлении оси Z, ко-

торая равна (0, 0, 1). Карта нормалей в касательном пространстве

очень подходит для создания анимации, т.к. для каждого состояния объекта потребовалась бы отдельная карта нормалей в пространстве модели.

Однако существует одна проблема, которая сильно ограничивает ис-

пользование карт нормалей в касательном пространстве. Требуется преобразовать нормали таким образом, чтобы полученные векторы смотрели в том же

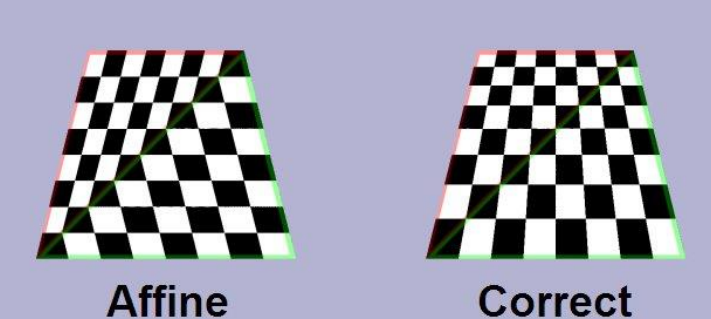
направлении, что и плоскость полигона (т.е. нужно перейти из касательного пространства в мировое пространство). Другое решение заключается в том, чтобы перенести освещение и камеру в касательное пространство, а векторы нормалей оставить неизменными. Таким образом, можно использовать одну и

ту же карту нормалей, независимо от ориентации объекта.

1. Зеркальная карта: задает способность отдельных точек объекта отражать свет. В большинстве случаев данная текстура является черно-белой модификацией диффузной карты. Каждый тексель зеркальной карты содержит коэффициент зеркального освещения для соответствующего фрагмента поверхности.

**Перспективная коррекция текстур**

Аффинное наложение текстур линейно интерполирует текстурные координаты по поверхности и является самым быстрым методом текстурирования. Поскольку данный способ не учитывает информацию о глубине фрагмента, когда поверхность расположена под углом к наблюдателю, можно заметить искажение текстуры. Выглядеть оно будет примерно так:



Чтобы выполнить перспективную коррекцию текстурных координат u и v, причем что z - компонент глубины зависит от точки зрения зрителя, мы можем воспользоваться тем фактом, что значения 1/z, u/z и v/z линейны в пространстве экрана по всей текстурируемой поверхности. В отличие от этого, исходные значения z, u и v до деления не являются линейными в пространстве экрана. Поэтому мы можем линейно интерполировать эти взаимно обратные величины по поверхности, вычисляя скорректированные значения в каждом пикселе, чтобы в результате получить текстурное отображение, корректное с точки зрения перспективы.

Для этого мы сначала вычисляем взаимные коэффициенты в каждой вершине нашей геометрии (3 точки для треугольника). Для вершины n мы имеем un / zn,vn / zn,1 / zn. Затем мы линейно интерполируем эти взаимности между n вершинами (например, используя барицентрические координаты), в результате чего получаем интерполированные значения по всей поверхности. В заданной точке это дает интерполированные значения ui, vi и zReciprocal\_i=1 / zi. Обратите внимание, что эти ui,vi пока нельзя использовать в качестве координат текстуры, так как деление на z изменило их систему координат.

Для коррекции обратно в пространство u,v мы сначала вычисляем скорректированный z, снова взяв обратную величину z\_correct=1 / zReciprocal\_i=1 / 1 / zi = zi. Затем мы используем это для коррекции ui,vi : u\_correct=ui/zi и v\_correct=vi / zi.

**Точечная и билинейная фильтрация текстур**

В процессе текстурирования каждому пикселю экрана ставится в соответствие координата внутри текстуры (не обязательно целочисленная). В зависимости от ситуации (степени масштабирования текстуры, угла наклона и т.д.), одному пикселю экрана может соответствовать несколько пикселей текстуры или же наоборот - один пиксель текстуры растянется на несколько экранных пикселей. В результате получившееся изображение может быть смазанным, ступенчатым или иметь другие артефакты. Для решения подобных проблем визуализации используются методы текстурной фильтрации.

**Точечная фильтрация**

Точечная фильтрация является простейшим методом фильтрации, при котором экранный пиксель получает значение цвета текселя, ближайшего к его центру.

Данный метод приводит к большому количеству артефактов - «блочности» текстуры при увеличении и мерцанию при минимизации.

**Билинейная фильтрация**

При билинейной фильтрации отбираются четыре ближайших к центру пикселя текселя и их цвета объединяются средневзвешенным значением в зависимости от расстояния. Это устраняет «блочность», наблюдаемую при увеличении, поскольку теперь существует плавный градиент изменения цвета от одного текселя к другому, а не резкий скачок, когда центр пикселя пересекает границу текселя.

Билинейная фильтрация для фильтрации при увеличении - обычное дело. При минификации она часто используется в связке с MIP-текстурами, хотя ее можно использовать и без них, при слишком сильной минификации она будет страдать теми же проблемами мерцания, что и фильтрация ближайших соседей. Однако при скромных коэффициентах минификации его можно использовать в качестве вычислительно недорогого решения для фильтрации текстур.

**33. Текстурирование трёхмерных объектов. Наложение текстур с перспективной**

**коррекцией. MIP-текстурирование и трилинейная фильтрация текстур.**

Текстурирование объектов в компьютерной графике — это процесс добавления деталей на поверхность объекта, что позволяет ему выглядеть более реалистично и интересно. Основная идея текстурирования схожа с применением обоев к стенам: сложный узор наносится на плоскую бумагу, а затем эта бумага накладывается на поверхность стены. В компьютерной графике такой же принцип применяется к объектам: узоры или изображения наносятся на их поверхности, улучшая визуальное восприятие и добавляя глубину и детальность.

Для применения текстуры к объекту необходимо использовать координаты текстуры (также известные как UV-координаты). Эти координаты представляют собой систему координат, в которой каждая вершина объекта имеет пару чисел, указывающих на местоположение в текстуре. Координата U горизонтальная и возрастает слева направо, V - вертикальная, возрастает снизу вверх(в экранной плоскости обычно наоборот, стоит это учитывать).

Текстура объекта обычно задается набором текстурных карт. Текстурная карта — это двухмерное изображение, которое накладывается на полигональную сетку модели для придания ей больше выразительности, добавления цвета или создания иллюзии реалистичности. Существует несколько основных используемых текстурных карт:

1. Диффузная карта: задает цвет каждого пикселя на поверхности объекта.
2. Карта нормалей: задает нормаль к поверхности в точках объекта.

Нормаль хранится в файле в виде цвета, нормализованная компонента которого изменяется от 0 до 1, для перехода к нормализованной компненте нормали, изменяющейся от -1 до 1 необходимо выполнить 2 действия : умножить компоненту на 2(получим диапазон от 0 до 2) и вычесть 1(получим искомый диапазон от -1 до 1). Выполнив действия в обратном порядке, можно трансформировать нормаль в цвет.

Существует две разновидности карт нормалей: карты нормалей, векторы которой находятся в пространстве модели, и карты

нормалей, у которой все векторы находятся в пространстве плоскости поли-

гона (в касательном пространстве). В касательном пространстве все нормали смотрят в положительном направлении оси Z, ко-

торая равна (0, 0, 1). Карта нормалей в касательном пространстве

очень подходит для создания анимации, т.к. для каждого состояния объекта потребовалась бы отдельная карта нормалей в пространстве модели.

Однако существует одна проблема, которая сильно ограничивает ис-

пользование карт нормалей в касательном пространстве. Требуется преобразовать нормали таким образом, чтобы полученные векторы смотрели в том же

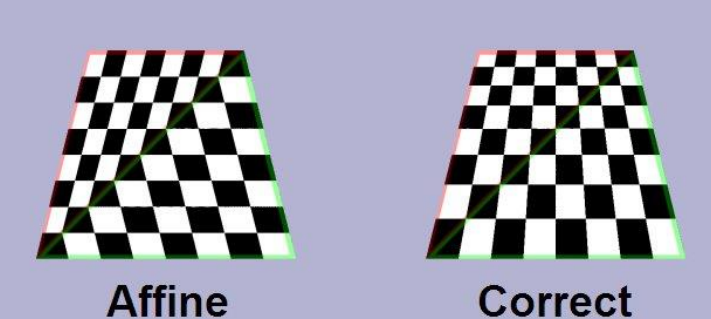
направлении, что и плоскость полигона (т.е. нужно перейти из касательного пространства в мировое пространство). Другое решение заключается в том, чтобы перенести освещение и камеру в касательное пространство, а векторы нормалей оставить неизменными. Таким образом, можно использовать одну и

ту же карту нормалей, независимо от ориентации объекта.

1. Зеркальная карта: задает способность отдельных точек объекта отражать свет. В большинстве случаев данная текстура является черно-белой модификацией диффузной карты. Каждый тексель зеркальной карты содержит коэффициент зеркального освещения для соответствующего фрагмента поверхности.

**Перспективная коррекция текстур**

Аффинное наложение текстур линейно интерполирует текстурные координаты по поверхности и является самым быстрым методом текстурирования. Поскольку данный способ не учитывает информацию о глубине фрагмента, когда поверхность расположена под углом к наблюдателю, можно заметить искажение текстуры. Выглядеть оно будет примерно так:



Чтобы выполнить перспективную коррекцию текстурных координат u и v, причем что z - компонент глубины зависит от точки зрения зрителя, мы можем воспользоваться тем фактом, что значения 1/z, u/z и v/z линейны в пространстве экрана по всей текстурируемой поверхности. В отличие от этого, исходные значения z, u и v до деления не являются линейными в пространстве экрана. Поэтому мы можем линейно интерполировать эти взаимно обратные величины по поверхности, вычисляя скорректированные значения в каждом пикселе, чтобы в результате получить текстурное отображение, корректное с точки зрения перспективы.

Для этого мы сначала вычисляем взаимные коэффициенты в каждой вершине нашей геометрии (3 точки для треугольника). Для вершины n мы имеем un / zn,vn / zn,1 / zn. Затем мы линейно интерполируем эти взаимности между n вершинами (например, используя барицентрические координаты), в результате чего получаем интерполированные значения по всей поверхности. В заданной точке это дает интерполированные значения ui, vi и zReciprocal\_i=1 / zi. Обратите внимание, что эти ui,vi пока нельзя использовать в качестве координат текстуры, так как деление на z изменило их систему координат.

Для коррекции обратно в пространство u,v мы сначала вычисляем скорректированный z, снова взяв обратную величину z\_correct=1 / zReciprocal\_i=1 / 1 / zi = zi. Затем мы используем это для коррекции ui,vi : u\_correct=ui/zi и v\_correct=vi / zi.

**MIP-текстурирование**

MIP-текстурирование — это техника в компьютерной графике, которая позволяет улучшить качество рендеринга текстур на объектах, находящихся на разных расстояниях от камеры. Термин "mipmap" происходит от латинского выражения "multum in parvo", что переводится как "много в одном месте".

Как работает MIP-текстурирование

MIP**-**текстурирование использует набор версий одной и той же текстуры, каждая из которых имеет меньшее разрешение по сравнению с предыдущей. Например, первая версия текстуры (уровня 0) имеет исходное разрешение, вторая версия (уровня 1) — половину размера первой, третья версия (уровня 2) — четверть первого размера, и так далее.

При рендеринге объектов в 3D-сцене выбирается наиболее подходящий уровень MIP-текстуры в зависимости от расстояния объекта от камеры. Ближайшие объекты рендерятся с использованием версий с высоким разрешением, а более далёкие объекты — с использованием версий низкого разрешения. Это позволяет достичь более плавного перехода деталей и уменьшить артефакты, такие как пикселизация на близком расстоянии и мерцание на далёком расстоянии, которые могут возникнуть при рендеринге текстур.

При использовании MIP-текстур происходит улучшение качества рендеринга, более плавный переход деталей и уменьшение артефактов, а также оптимизация производительности необходимо выполнять меньшее количество операций с уменьшенными версиями текстур, что ускоряет процесс рендеринга.

Недостатком такого текстурирования можно считать увеличение размера текстур: MIP-текстуры увеличивают размер текстуры примерно на 33%, что может быть ощутимо для текстур с высоким разрешением.

**Трилинейная фильтрация**

Трилинейная фильтрация - это метод фильтрации текстур, основанный на интерполяции результатов билинейной фильтрации для MIP-текстур. Она может быть использована для устранения распространённого артефакта, наблюдаемого в билинейно отфильтрованных изображениях: резкого и очень заметного изменения качества на границах, где рендерер переключается с одного уровня MIP-текстуры на другой.

Трилинейная фильтрация решает эту проблему, выполняя поиск текстуры и билинейную фильтрацию на двух ближайших уровнях MIP-текстуры (один более высокого, а другой более низкого качества), а затем линейно интерполируя результаты.

Это приводит к плавному снижению качества текстуры по мере увеличения расстояния от зрителя, а не к серии внезапных падений. Ближе к уровню 0 доступен только один уровень MIP-текстуры, и алгоритм возвращается к билинейной фильтрации.

**34. Текстурирование трёхмерных объектов. Наложение текстур с перспективной**

**коррекцией. MIP-текстурирование и анизотропная фильтрация текстур.**

**33. Текстурирование трёхмерных объектов. Наложение текстур с перспективной**

**коррекцией. MIP-текстурирование и трилинейная фильтрация текстур.**

Текстурирование объектов в компьютерной графике — это процесс добавления деталей на поверхность объекта, что позволяет ему выглядеть более реалистично и интересно. Основная идея текстурирования схожа с применением обоев к стенам: сложный узор наносится на плоскую бумагу, а затем эта бумага накладывается на поверхность стены. В компьютерной графике такой же принцип применяется к объектам: узоры или изображения наносятся на их поверхности, улучшая визуальное восприятие и добавляя глубину и детальность.

Для применения текстуры к объекту необходимо использовать координаты текстуры (также известные как UV-координаты). Эти координаты представляют собой систему координат, в которой каждая вершина объекта имеет пару чисел, указывающих на местоположение в текстуре. Координата U горизонтальная и возрастает слева направо, V - вертикальная, возрастает снизу вверх(в экранной плоскости обычно наоборот, стоит это учитывать).

Текстура объекта обычно задается набором текстурных карт. Текстурная карта — это двухмерное изображение, которое накладывается на полигональную сетку модели для придания ей больше выразительности, добавления цвета или создания иллюзии реалистичности. Существует несколько основных используемых текстурных карт:

1. Диффузная карта: задает цвет каждого пикселя на поверхности объекта.
2. Карта нормалей: задает нормаль к поверхности в точках объекта.

Нормаль хранится в файле в виде цвета, нормализованная компонента которого изменяется от 0 до 1, для перехода к нормализованной компненте нормали, изменяющейся от -1 до 1 необходимо выполнить 2 действия : умножить компоненту на 2(получим диапазон от 0 до 2) и вычесть 1(получим искомый диапазон от -1 до 1). Выполнив действия в обратном порядке, можно трансформировать нормаль в цвет.

Существует две разновидности карт нормалей: карты нормалей, векторы которой находятся в пространстве модели, и карты

нормалей, у которой все векторы находятся в пространстве плоскости поли-

гона (в касательном пространстве). В касательном пространстве все нормали смотрят в положительном направлении оси Z, ко-

торая равна (0, 0, 1). Карта нормалей в касательном пространстве

очень подходит для создания анимации, т.к. для каждого состояния объекта потребовалась бы отдельная карта нормалей в пространстве модели.

Однако существует одна проблема, которая сильно ограничивает ис-

пользование карт нормалей в касательном пространстве. Требуется преобразовать нормали таким образом, чтобы полученные векторы смотрели в том же

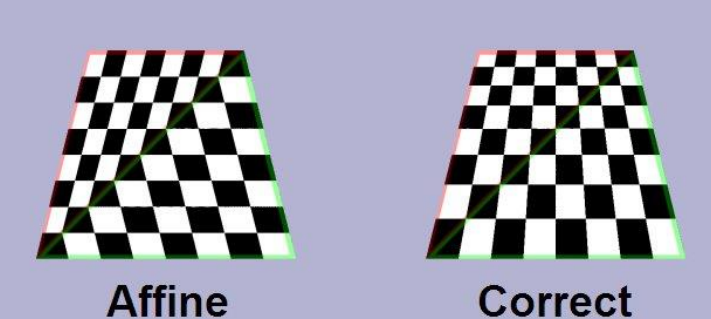
направлении, что и плоскость полигона (т.е. нужно перейти из касательного пространства в мировое пространство). Другое решение заключается в том, чтобы перенести освещение и камеру в касательное пространство, а векторы нормалей оставить неизменными. Таким образом, можно использовать одну и

ту же карту нормалей, независимо от ориентации объекта.

1. Зеркальная карта: задает способность отдельных точек объекта отражать свет. В большинстве случаев данная текстура является черно-белой модификацией диффузной карты. Каждый тексель зеркальной карты содержит коэффициент зеркального освещения для соответствующего фрагмента поверхности.

**Перспективная коррекция текстур**

Аффинное наложение текстур линейно интерполирует текстурные координаты по поверхности и является самым быстрым методом текстурирования. Поскольку данный способ не учитывает информацию о глубине фрагмента, когда поверхность расположена под углом к наблюдателю, можно заметить искажение текстуры. Выглядеть оно будет примерно так:



Чтобы выполнить перспективную коррекцию текстурных координат u и v, причем что z - компонент глубины зависит от точки зрения зрителя, мы можем воспользоваться тем фактом, что значения 1/z, u/z и v/z линейны в пространстве экрана по всей текстурируемой поверхности. В отличие от этого, исходные значения z, u и v до деления не являются линейными в пространстве экрана. Поэтому мы можем линейно интерполировать эти взаимно обратные величины по поверхности, вычисляя скорректированные значения в каждом пикселе, чтобы в результате получить текстурное отображение, корректное с точки зрения перспективы.

Для этого мы сначала вычисляем взаимные коэффициенты в каждой вершине нашей геометрии (3 точки для треугольника). Для вершины n мы имеем un / zn,vn / zn,1 / zn. Затем мы линейно интерполируем эти взаимности между n вершинами (например, используя барицентрические координаты), в результате чего получаем интерполированные значения по всей поверхности. В заданной точке это дает интерполированные значения ui, vi и zReciprocal\_i=1 / zi. Обратите внимание, что эти ui,vi пока нельзя использовать в качестве координат текстуры, так как деление на z изменило их систему координат.

Для коррекции обратно в пространство u,v мы сначала вычисляем скорректированный z, снова взяв обратную величину z\_correct=1 / zReciprocal\_i=1 / 1 / zi = zi. Затем мы используем это для коррекции ui,vi : u\_correct=ui/zi и v\_correct=vi / zi.

**MIP-текстурирование**

MIP-текстурирование — это техника в компьютерной графике, которая позволяет улучшить качество рендеринга текстур на объектах, находящихся на разных расстояниях от камеры. Термин "mipmap" происходит от латинского выражения "multum in parvo", что переводится как "много в одном месте".

Как работает MIP-текстурирование

MIP**-**текстурирование использует набор версий одной и той же текстуры, каждая из которых имеет меньшее разрешение по сравнению с предыдущей. Например, первая версия текстуры (уровня 0) имеет исходное разрешение, вторая версия (уровня 1) — половину размера первой, третья версия (уровня 2) — четверть первого размера, и так далее.

При рендеринге объектов в 3D-сцене выбирается наиболее подходящий уровень MIP-текстуры в зависимости от расстояния объекта от камеры. Ближайшие объекты рендерятся с использованием версий с высоким разрешением, а более далёкие объекты — с использованием версий низкого разрешения. Это позволяет достичь более плавного перехода деталей и уменьшить артефакты, такие как пикселизация на близком расстоянии и мерцание на далёком расстоянии, которые могут возникнуть при рендеринге текстур.

При использовании MIP-текстур происходит улучшение качества рендеринга, более плавный переход деталей и уменьшение артефактов, а также оптимизация производительности необходимо выполнять меньшее количество операций с уменьшенными версиями текстур, что ускоряет процесс рендеринга.

Недостатком такого текстурирования можно считать увеличение размера текстур: MIP-текстуры увеличивают размер текстуры примерно на 33%, что может быть ощутимо для текстур с высоким разрешением.

**Анизотропная фильтрация**

Анизотропная фильтрация - это метод фильтрации текстур на поверхностях, находящихся под косыми углами обзора по отношению к камере, когда проекция текстуры (а не полигона или другого примитива, на который она отрисована) оказывается неортогональной.

Рассмотрим принцип работы анизотропной фильтрации:

Если нам нужно текстурировать горизонтальную плоскость, которая находится под косым углом к камере, то традиционный MIP-маппинг даст недостаточное горизонтальное разрешение из-за снижения частоты изображения по вертикальной оси. Это связано с тем, что при MIP-маппинге каждый уровень пропорционален, поэтому текстура 256 × 256 уменьшается до изображения 128 × 128, затем 64 × 64 и так далее и разрешение уменьшается вдвое по каждой оси одновременно. Поскольку разрешение в целом изменяется только по одной из осей, уменьшение второй неизбежно приводит к снижению разрешения текстуры по ней и соответственно изображение выглядит размыто.

В процессе рендеринга могут применяться различные степени или коэффициенты анизотропной фильтрации.

Эта степень относится к максимальному коэффициенту, поддерживаемому в фильтрации. Например, анизотропная фильтрация 4:1 будет продолжать повышать резкость более наклонных текстур, выходя за пределы диапазона резкости 2:1(то есть вместо усреднения цвета в пикселе за счет сетки 2:2 из ближайших пикселей будет использоваться сетка 4:4).

На практике это означает, что в ситуациях с сильно наклонёнными текстурами фильтр 4:1 будет вдвое резче, чем фильтр 2:1. Однако для большей части сцены фильтр 4:1 не потребуется; только более наклонные и, как правило, более удаленные пиксели будут нуждаться в более резкой фильтрации. Это означает, что при увеличении степени анизотропной фильтрации в два раза снижается отдача в плане видимого качества: все меньше и меньше пикселей затрагивается при рендеринге, и результаты становятся менее очевидными для зрителя.

Если сравнить результаты визуализации сцены с анизотропной фильтрацией 8:1 и сцены с фильтрацией 16:1, то в сцене с более высокой степенью анизотропной фильтрации только несколько сильно наклонённых пикселей, в основном на более удалённой геометрии, будут отображать заметно более чёткие текстуры, а частотная информация на этих нескольких пикселях с фильтрацией 16:1 будет вдвое больше, чем при фильтрации 8:1.

В общем случае алгоритм анизотропной фильтрации определяет степень искажения пикселя относительно его изначального положения и в зависимости от степени искажения подбирается соответствующий размер сетки пикселей для выборки, чей цвет усредняется и присваивается рассматриваемому пикселю экрана.