# Введение

В условиях современного уровня технического прогресса становится очевидными польза и превалирование методов визуализации данных, использующих компьютерные технологии. Область применения таких методов также развивается и проникает даже в области, не связанные напрямую с информационными технологиями, такие как искусство, архитектура, кино, и многочисленные сферы развлечения и организации досуга. Способами графического представления данных с помощью компьютерных технологий занимается такая дисциплина, как компьютерная графика.

*Компьютерная графика* (машинная, цифровая графика) – область деятельности, в которой ЭВМ используются в качестве инструмента для создания реалистичных изображений, а также для обработки визуальной информации, полученной из реального мира, например, фото, видео [1]. Компьютерная графика справляется с задачей построения трехмерных моделей, в которых будут представлены тени, прозрачность и фактура, учитывающие эффекты отражения и преломления цвета, а также многие другие их статические и динамические характеристики.

Целью данной практики является разработка аналитической части проекта по дисциплине «Компьютерная графика» на тему «создание трехмерной модели музея «Мир Науки» города Ванкувер».

Для достижения поставленной цели требуется решить следующие задачи:

1) Описать структуру трехмерной модели, а именно объекты, из которых она состоит.

2) Рассмотреть существующие алгоритмы трехмерной графики для визуализации реалистичных изображений.

3) Выбрать наиболее подходящий алгоритм для обозначенной выше темы и объяснить выбор.

# 1. Аналитический раздел

## 1.1 Постановка задачи

В соответствии с обозначенным техническим заданием, в рамках данного проекта необходимо создать программный продукт, обеспечивающий визуализацию трехмерной модели здания – музея «Мир Науки» города Ванкувер, Канада. Модель статическая и должна состоять из нескольких объектов, имеющих зеркальную поверхность. Также необходимо реализовать возможность масштабирования сцены. Полученная визуализация должна быть максимально реалистичным изображением здания музея на рис. 1.1 в качестве его схематичной модели.



Рисунок 1.1 – Музей «Мир Науки» г. Ванкувер, Канада.

Модель состоит из следующих основных объектов:

1. Нижняя часть здания – правильная призма, основание которой параллельно оси OXY, а боковые грани – перпендикулярны ей.
2. Верхняя часть здания – геодезический купол.

## 1.2 Анализ алгоритмов удаления невидимых линий и поверхностей

Алгоритмы удаления невидимых линий и поверхностей служат для определения линий ребер, поверхностей или объемов, видимых или невидимых наблюдателю, находящемуся в заданной точке пространства. Универсального решения этой задачи не существует, так как для каждой отдельной цели выдвигаются разные приоритеты в области используемых ресурсов. Например, для некоторых задач, таких как моделирование процессов в реальном времени, требуется высокая скорость выполнения, в то время как при создании реалистичных объектов или сцен этим фактором можно пренебречь. Иными словами, скорость работы алгоритма обратно пропорциональна детальности его результата. Также важно отметить, что алгоритмы удаления невидимых линий и поверхностей делятся на два типа по способу выбора системы координат, в которой они работают: алгоритмы, работающие в объектном пространстве (используется физическая система координат, в которой описаны эти объекты, за счет которой точность результата ограничивается лишь точностью вычислений), и алгоритмы, работающие в пространстве изображения (используется система координат экрана, на котором объекты визуализируются, что ставит результат в зависимость от разрешающей способности экрана и затрудняет такие задачи, как масштабирование) [1]. Рассмотрим некоторые алгоритмы подробнее.

## 1.2.1 Алгоритм Робертса

Алгоритм Робертса представляет собой первое известное решение задачи об удалении невидимых линий. Этот метод работает в объектном пространстве и строго с выпуклыми телами (если тело таковым не является, его следует разбить на выпуклые части). Алгоритм прежде всего удаляет из каждого тела те ребра или грани, которые экранируются самим телом. Затем каждое из видимых ребер каждого тела сравнивается с каждым из оставшихся тел для определения того, какая его часть или части, если таковые есть, экранируются этими телами. Теоретически вычислительная трудоемкость алгоритма Робертса растет, как квадрат числа объектов. Однако математические методы, используемые в этом алгоритме, являются простыми и точными [1].

Работа Алгоритм Робертса проходит в несколько этапа:

1) Подготовка исходных данных.

2) Удаление линий, экранируемых самим телом.

3) Удаление линий, экранируемых другими телами.

4) Удаление линий пересечения тел, экранируемых самими телами, связанными отношением протыкания и другими телами.

1. Подготовка исходных данных.

Для каждого тела сцены необходимо сформировать матрицу тела. Обозначают: V. Размерность: 4 \* n, где n - количество граней. Каждый столбец матрицы - это коэффициенты уравнения плоскости, проходящей через очередную грань тела.

Уравнение плоскости:

ax + by + cz + d = 0 (1.2)

Или в матричном виде:

[x y z 1] (1.3)

Тогда матрица V будет иметь вид:

(1.4)

где n – количество граней.

Имея информацию о координатах 3 неколлинеарных точек, при­ надлежащих плоскости, можно найти коэффициенты этой плоскости. Делим уравнение на d, при этом свободный член будет равен единице.

(1.5)

Или в матричном виде:

= (1.6)

Решая полученное уравнение (1.6), получим значения коэффициентов уравнения плоскости: a, b, c. Сформированную матрицу тела V проверяют на корректность.

2. Удаление линий, экранируемых самим телом.

Для решения этой задачи нужно указать место расположения наблюдателя и направление его взгляда.

E = |00-10| (1.7)

С одной стороны (1.7) - вектор взгляда наблюдателя, с другой сторонs - вектор однородных координат точки, расположенной в минус бесконечности по оси z. Отрицательные компоненты вектора, полученного в результате произведения вектора взгляда E и матрицы тела V, позволят определить невидимые грани.

3. Удаление линий, экранируемых другими телами.

Наблюдатель находится на положительном направлении оси z (в +∞). На этом этапе используются координаты точки, в которой находится наблюдатель. Для определения невидимых точек ребра следует построить луч, соединяющий точку наблюдения с точкой на ребре тела. Если луч на своем пути в качестве преграды встречает данное тело, рассматриваемая точка будет невидимой.

Если тело является преградой, то луч должен пройти через тело и, следовательно, луч находится по положительную сторону от каждой грани этого тела.

4. Удаление линий пересечения тел, экранируемых самими телами, связанными отношением протыкания и другими телами.

В случае, когда тела связаны отношением взаимного протыкания, образуются новые ребра, соответствующие линиям пересечения этих тел. Соединяя точки протыкания между собой, получаем новые ребра. Затем следует повторить этапы 1-2 для полученных новых ребер.

Достоинством этого алгоритма является простота и четкость используемых математических методов. Недостатки заключаются в большой вычислительной трудоемкости и невозможности воспроизводства визуальных эффектов без дополнительных методов, что делает его менее преферентным для реализации реалистичных объектов или сцен, и хотя его более поздние реализации позволили достичь почти линейной зависимости трудоемкости от числа объектов, они достаточно сложны.

## 

## 1.2.2 Алгоритм Варнока

В противовес алгоритму Робертса, Варнок в 1968 г. предложил алгоритм, работающий не в объектном пространстве, а в пространстве изображения. Он также нацелен на изображение многогранников, а главная идея его основана на гипотезе о способе обработки информации, содержащейся в сцене, глазом и мозгом человека. Большая часть времени и труда затрачивается на области с высоким информационным содержимым. В алгоритме Варнока и его вариантах делается попытка воспользоваться тем, что большие области изображения однородны. Такое свойство называют когерентностью, имея в виду, что смежные области (пиксели) вдоль обеих осей X и Y имеют тенденцию к однородности. Конкретная реализация алгоритма Варнока зависит от метода разбиения окна и от деталей критерия, используемого для того, чтобы решить, является ли содержимое окна достаточно простым [1]. В оригинальной версии алгоритма каждое окно разбивалось на четыре одинаковых подокна до тех пор, пока не был получен результат или окно не станет равным 1 пикселю. Для общего описания алгоритма необходимо классифицировать окно следующим образом:

1. Все многоугольники сцены являются внешними по отношению к окну: окно пусто, следовательно, изображается фоновым цветом и дальнейшему разбиению не подлежит.
2. Только один многоугольник сцены имеет общие точки с окном и является по отношению к нему внутренним: окно закрашивается фоновым цветом, а сам многоугольник закрашивается своим цветом.
3. Только один многоугольник сцены имеет общие точки с окном и является по отношению к нему пересекающим: окно закрашивается цветом фона, а часть многоугольника, принадлежащая окну, закрашивается цветом многоугольника.
4. Только один многоугольник охватывает окно и нет других многоугольников, имеющих общие точки с окном: окно закрашивается цветом этого многоугольника.
5. Существует хотя бы один многоугольник, охватывающий окно: среди всех таких многоугольников выбирается тот, который расположен ближе всех к точке наблюдения, и окно закрашивается цветом этого многоугольника.

Таким образом, алгоритм Варнока базируется на рекурсивном разбиении экрана. Время выполнения данного алгоритма зависит от расположения объектов и количества их пересечений.

## 1.2.3 Алгоритм Вейлера-Азертона

Данный алгоритм представляет собой оптимизацию алгоритма Варнока в отношении числа выполняемых разбиений. Был предпринят переход от прямоугольных разбиений к разбиениям вдоль границ многоугольников. Алгоритм работает в объектном пространстве, и результатом его работы являются многоугольники. В самом общем виде он состоит из четырех шагов.

1. Предварительная сортировка по глубине (нужна для формирования списка приблизительных приоритетов).
2. Отсечение по границе ближайшего к точке наблюдения многоугольника, называемое сортировкой многоугольников на плоскости.
3. Удаление многоугольников, экранируемых более близкими к точке наблюдения многоугольниками.
4. Если требуется, то рекурсивное разбиение и новая сортировка.

Эффективность алгоритма Вейлера-Азертона сводится к эффективности метода разбиения криволинейной поверхности [1] К недостатком алгоритма относится невозможность воспроизведения визуальных эффектов без дополнительных методов и оптимизаций.

## 1.2.4 Алгоритм, использующий Z буфер

Этот алгоритм работает в пространстве изображения. Идея Z-буфера является простым обобщением идеи о буфере кадра. Буфер кадра используется для запоминания атрибутов каждого пикселя в пространстве изображения, а Z-буфер предназначен для запоминания глубины (z-координаты) каждого видимого пикселя в пространстве изображения. В процессе работы значение глубины каждого нового пикселя, который нужно занести в буфер кадра, сравнивается с глубиной того пикселя, который уже занесен в Z-буфер. Если это сравнение показывает, что новый пиксель расположен впереди пикселя, находящегося в буфере кадра, то новый пиксель заносится в этот буфер и, кроме того, производится корректировка Z-буфера новым значением глубины. Если же сравнение дает противоположный результат, то никаких действий не производится.

Главное преимущество алгоритма заключается в его простоте. Сцены могут быть любой сложности. Поскольку габариты пространства изображения фиксированы, оценка вычислительной трудоемкости алгоритма не более чем линейна. Также важно отметить, что сцены или картинки можно заносить в Z-буфер в произвольном порядке, и поэтому их не нужно предварительно сортировать по приоритету глубины. Таким образом, экономится вычислительное время, затрачиваемое на сортировку по глубине.

Главные недостатки алгоритма - большой объем затрачиваемой памяти и система координат, в которой он работает, затрудняющая такие преобразования сцены, как масштабирование.

## 1.2.5 Алгоритмы трассировки лучей

Метод прямой и обратной трассировки заключается в том, что от момента испускания лучей источником света до момента попадания в камеру, траектории лучей отслеживаются, и рассчитываются взаимодействия лучей с лежащими на траекториях объектами. Луч может быть поглощен, диффузно или зеркально отражен или, в случае прозрачности некоторых объектов, преломлен.

Ray tracing – метод расчета глобального освещения, рассматривающий освещение, затенение (расчет тени), многократные отражения и преломления.

Для расчета теней применяют методы прямой и обратной трассировки лучей.

Метод прямой трассировки предполагает построение траекторий лучей от всех источников освещения ко всем точкам всех объектов сцены. Это, так называемые, первичные лучи. Точки, лежащие на противоположной от источника света стороне, исключаются из расчета. Для всех остальных точек вычисляется освещенность с помощью локальной модели освещения. Если объект не является отражающим или прозрачным, то траектория луча на этой точке обрывается. Если же поверхность объекта обладает свойством отражения или преломления, то из точки строятся новые лучи, направления которых совершенно точно определяются законами отражения и преломления. Траектории новых лучей также отслеживаются. Построение новых траекторий и расчеты ведутся до тех пор, пока все лучи либо попадут в камеру, либо выйдут за пределы видимой области. Очевидно, что при прямой трассировке лучей мы вынуждены выполнять расчеты для лучей, которые не попадут в камеру, то есть проделывать бесполезную работу. По некоторым оценочным данным, доля таких «слепых» лучей довольно велика. Эта главная, хотя и далеко не единственная причина того, что метод прямой трассировки лучей считается неэффективным и на практике не используется [3].

Алгоритм обратной трассировки лучей - основной способ расчета освещенности методом трассировки лучей. Метод трассировки лучей – первый метод расчета глобальной освещенности, учитывающий взаимное влияние объектов сцены друг на друга. Основные достоинства метода обратной трассировки лучей – расчет теней, многократных отражений и преломлений, значительно повысивших степень реалистичности компьютерных рендеров.

Основные недостатки: низкая скорость и высокая вычислительная стоимость расчетов – в классическом рейтресинге необходимо проверять на пересечение каждый луч со всеми объектами сцены, в результате от 70 до 95 процентов всего времени расчетов тратится на вычисление пересечений [3].

Обратная трассировка позволяет работать с несколькими источниками света, передавать множество разных оптических явлений [4].

## 1.3 Преобразования трехмерной сцены

Перенос:

 (1.8)

где dx, dy, dz – это смещения по оси абсцисс, ординат, аппликат, соответственно.

Масштабирование относительно начала координат:

 (1.9)

где – это коэффициенты масштабирования по оси абсцисс, ординат, аппликат, соответственно. Результат масштабирования двумерного объекта представлен на рис. 1.2.

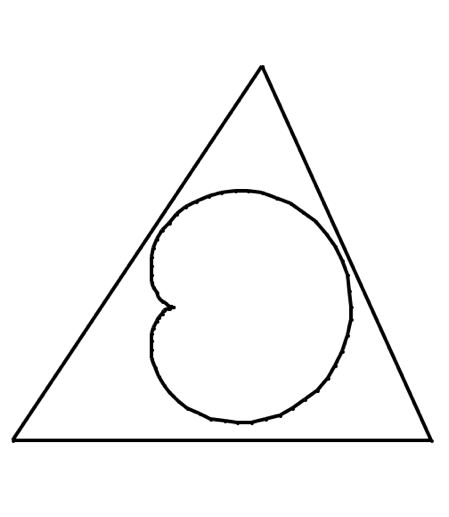
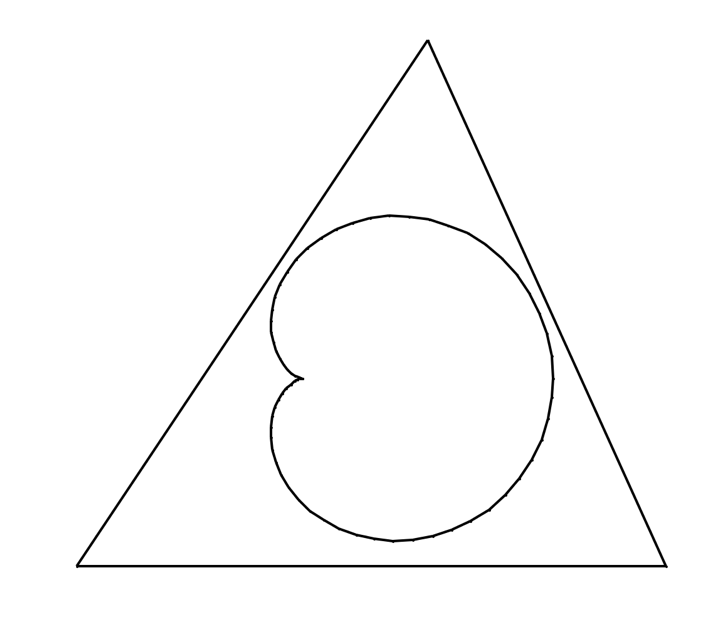
**

Рисунок 1.2 – Масштабирование двумерного объекта

Поворот относительно осей OX, OY, OZ на угол :

## 

## 1.4 Вывод

## В данном разделе были рассмотрены алгоритмы удаления невидимых линий и поверхностей, анализ которых позволяет определить, что для задачи построения статической реалистичной трехмерной модели с учетом теней и отражающих поверхностей наиболее оптимальным будет алгоритм обратной трассировки лучей, основной недостаток которого, заключающийся в низкой производительности при наличии динамических эффектов сцены, не будет негативно влиять на работу программного продукта в значительной степени, зато сумеет обеспечить наиболее высокую реалистичность полученного изображения.

**2. Конструкторский раздел**

## 2.1 Требования к программе

Программа должна предоставлять следующие возможности:

а) Визуализация сцены.

б) Масштабирование модели.

## 2.2 Алгоритм обратной трассировки лучей

Ниже представлена схема пересечения луча и объекта модели (2.1) в рамках алгоритма обратной трассировки лучей.

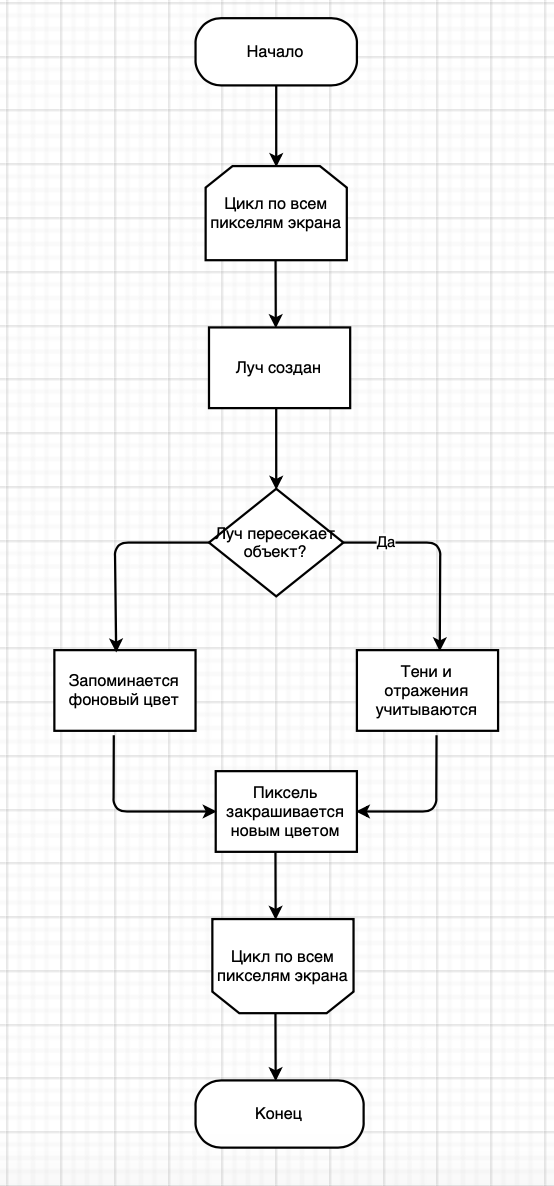


Рисунок 2.1 – Схема алгоритма обратной трассировки лучей

Рассмотрим уравнение луча:

P = O + t(V – O) (2.1)

где О - центр камеры, а V - текущий пиксель.

Обозначим направление луча:

= V – O (2.2)

Уравнение луча примет вид:

P = O + t (2.3)

Одним из объектов модели является сфера. Рассмотрим пересечение луча со сферой. Сфера представляет собой геометрическое место точек p в пространстве, равноудаленных от некоторой заданной точки C - центра сферы. Тогда имеет место быть следующее уравнение:

|p - C| = R (2.4)

где R – это длина вектора расстояния от геометрического пространства точек p до центра сферы C.

Справедливо следующее равенство:

, = (2.5)

Получим пересечение сферы и луча подстановкой (2.3) в (2.5):

, = (2.6)

Преобразовав полученное скалярное выражение (2.6), получим три исхода:

1. Одно решение – луч проходит по касательной к сфере.
2. Два решения – луч проходит через сферу.
3. Ни одного решения – луч не пересекает сферу.

## 2.3 Выбор используемых типов и структур данных

В рамках данного программного продукта нужно будет реализовать следующие типы и структуры данных.

1. Источник света – хранит направление света.
2. Сцена – хранит список объектов.
3. Объект сцены – модель здания:

3.1. Сфера – хранит радиус и центр сферы, заданный координатами точки.

3.2. Правильная призма – задается вершинами и гранями.

1. Математические абстракции:

4.1. Точка – хранит координаты x, y, z.

4.2. Вектор – хранит направление по x, y, z.

# Заключение

В ходе выполнения практики были разработаны аналитическая и конструкторская части проекта по дисциплине «Компьютерная графика» на тему «создание трехмерной модели музея «Мир Науки» города Ванкувер». Для достижения выше обозначенной цели была описана структура трехмерной модели, а именно объекты, из которых она состоит, рассмотрены существующие алгоритмы трехмерной графики для визуализации реалистичных изображений, а также был сделан и обоснован выбор наиболее подходящего для указанной выше темы алгоритма.

# Список использованных источников

1. *Роджерс, Д*. Алгоритмические основы машинной графики / Д. Роджерс. — Москва «Мир», 1989. — P. 512.
2. *Дымченко, Лев*. Пример реализации в реальном времени метода трассировки лучей: необычные возможности и принцип работы. Оптимизация под SSE / Лев Дымченко. — ixbt, 2001. — 11. https://www.ixbt.com/video/ rt-raytracing.shtml.
3. *Янова, Регина*. Метод прямой и обратной трассировки лучей / Янова Р.Ю. – 2016. https://cyberleninka.ru/article/n/metod-pryamoy-i-obratnoy-trassirovki
4. *Снижко, Елена*. Компьютерная геометрия и графика / Снижко Е.А. - Балт. гос. техн. ун-т, 2005. - 17 с. https://www.elibrary.ru/item.asp?id=19584948