



Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет
имени Н. Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н. Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

К КУРСОВОЙ РАБОТЕ

НА ТЕМУ:

«Морфинг трехмерных моделей»

Студент ИУ7-54Б
(Группа)

(Подпись, дата)

Писаренко Д. П.
(И. О. Фамилия)

Руководитель курсовой работы

(Подпись, дата)

Кузнецова О. В.
(И. О. Фамилия)

2023 г.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ	4
1 Аналитический раздел	5
1.1 Формализация объектов синтезируемой сцены	5
1.2 Способ задания моделей	6
1.3 Анализ алгоритмов морфинга	6
1.3.1 Линейный морфинг	7
1.3.2 Весовая деформация	8
1.3.3 Морфинг на основе ключевых точек	10
1.4 Анализ алгоритмов закраски	11
1.4.1 Простая закраска	11
1.4.2 Закраска методом Гуро	12
1.4.3 Закраска методом Фонга	13
1.5 Анализ алгоритмов удаления невидимых линий и поверхностей	14
1.5.1 Алгоритм Робертса	14
1.5.2 Алгоритм Z-буфера	15
1.5.3 Обратная трассировка лучей	15
2 Конструкторский раздел	17
2.1 Требования к программному обеспечению	17
2.2 Считывание моделей	17
2.3 Общий алгоритм построения изображения	18
2.4 Алгоритм морфинга	21
3 Технологический раздел	23
3.1 Средства реализации	23
3.2 Выбор структур данных	23
3.3 Структура программы	24
3.4 Реализация алгоритмов	25
4 Исследовательский раздел	26
ЗАКЛЮЧЕНИЕ	27

ВВЕДЕНИЕ

Морфинг трехмерных моделей представляет собой технику анимации в компьютерной графике, направленную на плавное переходное изменение одной трехмерной модели в другую. Этот процесс создает эффект плавного метаморфоза между двумя или более формами, что может быть использовано для создания динамичных и визуально привлекательных анимаций.

Морфинг может применяться в различных областях, таких как компьютерная анимация, визуализация данных, медицинская графика и игровая индустрия. Эта техника позволяет создавать плавные и естественные анимации, предоставляя художникам и разработчикам инструмент для трансформации объектов внутри виртуальных сцен.

Целью данной работы является разработка программного обеспечения для морфинга трехмерных моделей. Для достижения поставленной цели требуется решить следующие задачи:

- 1) Формализовать представление объектов и описать их.
- 2) Проанализировать алгоритмы морфинга трехмерных моделей и выбрать наилучшие для достижения цели.
- 3) Выбрать средства реализации алгоритмов.
- 4) Реализовать выбранные алгоритмы.
- 5) Реализовать графический интерфейс.
- 6) Исследовать временные характеристики выбранных алгоритмов на основе созданного программного обеспечения.

1 Аналитический раздел

В данном разделе производится формализация объектов сцены, анализ алгоритмов их визуализации, и выбираются наиболее подходящие для решения поставленных задач.

1.1 Формализация объектов сцены

На визуализируемой сцене могут находиться следующие объекты:

1) Точечный источник света

Данный источник света испускает его равномерно во всех направлениях из фиксированной точки в трехмерном пространстве. Источник характеризуется:

- Интенсивностью.
- Позицией.

2) Рассеянный источник света

Привносит часть освещения в каждую точку сцены, независимо от ее расположения. Источник характеризуется:

- Интенсивностью.

3) Камера

В данном случае камера будет описана:

- Матрицей проекции. Она определяет границы видимости в трехмерном пространстве.
- Позицией.

4) Трехмерные объекты

Трехмерные объекты будут описываться либо с помощью каркасной модели (во время морфинга), либо с помощью поверхностной модели (в статичном состоянии). Для этого задается:

- Набор вершин.
- Набор граней.

1.2 Способ задания моделей

Для универсальности нашей программы выберем один из самых популярных форматов - .obj файлы. Файлы формата .obj (Wavefront OBJ) представляют собой текстовый формат, используемый для хранения трехмерных графических моделей. Они могут содержать следующую информацию:

1) Вершины

Координаты точек в трехмерном пространстве, которые определяют форму объекта.

2) Нормали

Направления поверхности в каждой вершине, важные для расчета освещения.

3) Текстурные координаты

Координаты, используемые для нанесения текстуры на поверхность модели.

4) Грани

Определяют полигоны объекта, указывая индексы вершин, текстурных координат и нормалей, составляющих грани.

Поскольку мы будем использовать каркасные и поверхностные модели, из каждого файла нам нужно будет вычленять лишь вершины и грани. Таким образом, для определенного объекта будет составлен список его вершин, а также список граней, представляющих собой набор вершин.

1.3 Анализ алгоритмов морфинга

Морфинг — технология компьютерной графики, обеспечивающая плавный переход формы одного объекта в форму другого [1].

В последнее время наблюдается большой интерес к технологиям морфинга для создания плавных переходов между изображениями. Эти методы сочетают в себе 2D-интерполяцию формы и цвета для создания впечатляющих эффектов перехода. Отчасти привлекательность морфинга заключается в том, что создаваемые изображения могут выглядеть поразительно реалистичными

и визуально убедительными. Несмотря на то, что они вычисляются с помощью преобразований 2D-изображений, эффективные морфинги могут предполагать естественную трансформацию объектов в 3D-мире.

1.3.1 Линейный морфинг

Линейный морфинг (англ. Linear Morphing) является самым простым и распространенным подходом. Он основан на линейной интерполяции между начальными и конечными позициями каждой точки или вершины модели. Путем изменения координат каждой точки по ходу времени, модель постепенно переходит из одной формы в другую [2].

Сам морфинг состоит из нескольких этапов:

- 1) Загрузка начальной и конечной модели морфинга, допускается несколько промежуточных состояний для плавности анимации.
- 2) Переход между позициями вершин.
- 3) Обновление текстурных координат.
- 4) Расчет промежуточных геометрических состояний.

Переход между позициями вершин — для каждой вершины модели вычисляются ее промежуточные позиции между начальной и конечной точками. Обычно это достигается путем линейной интерполяции координат вершин по временному параметру, который определяет текущий прогресс между началом и концом морфинга.

Обновление текстурных координат — аналогично с расчетом промежуточных позиций вершин необходимо с помощью интерполяции рассчитать текстурные координаты.

Расчет промежуточных геометрических состояний — на основе рассчитанных ранее координат возможно получить промежуточные геометрические состояния модели [2].

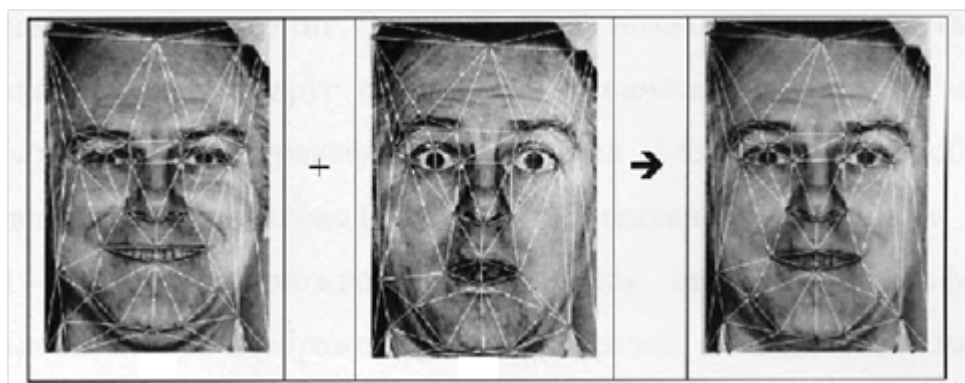


Рисунок 1.1 – Пример линейного морфинга человеческого лица

Плюсы:

- 1) Относительно небольшие вычислительные затраты [3].

Минусы:

- 1) Ограниченные возможности для создания сложных форм и детализации.
- 2) Ограниченный контроль над динамическими атрибутами, такими как цвета или нормали.
- 3) Не всегда способен обеспечить реалистичный морфинг для моделей с большим количеством вершин или сложной геометрией [3].

1.3.2 Весовая деформация

Идея данного метода аналогична методу, описанному в 1.3, однако в данном случае каждая из вершин имеет собственный вес. Изначально каждая вершина присваивается какой-либо части тела объекта (кости). Каждая вершина модели привязывается к костям с помощью весов. Вес определяет, насколько каждая кость влияет на вершину. Чаще всего веса представлены числами от 0 до 1, где 0 означает полное отсутствие влияния, а 1 — полное влияние соответствующей кости. Весовые значения для каждой вершины нормализуются, так чтобы сумма весов для каждой вершины была равна 1. Это позволяет равномерно распределить влияние костей и избежать нежелательных артефактов. В процессе анимации, для каждого кадра, путем комбинирования позиций и ориентаций костей с их соответствующими весами, определяются новые позиции вершин. Это достигается путем линейной интерполяции или других методов комбинирования весовых значений и позиций костей [4].



Рисунок 1.2 – Иллюстрация веса вершины в зависимости от ее положения

На картинке 1.2 с помощью интенсивности красного цвета показывается значение веса вершины в зависимости от ее положения.

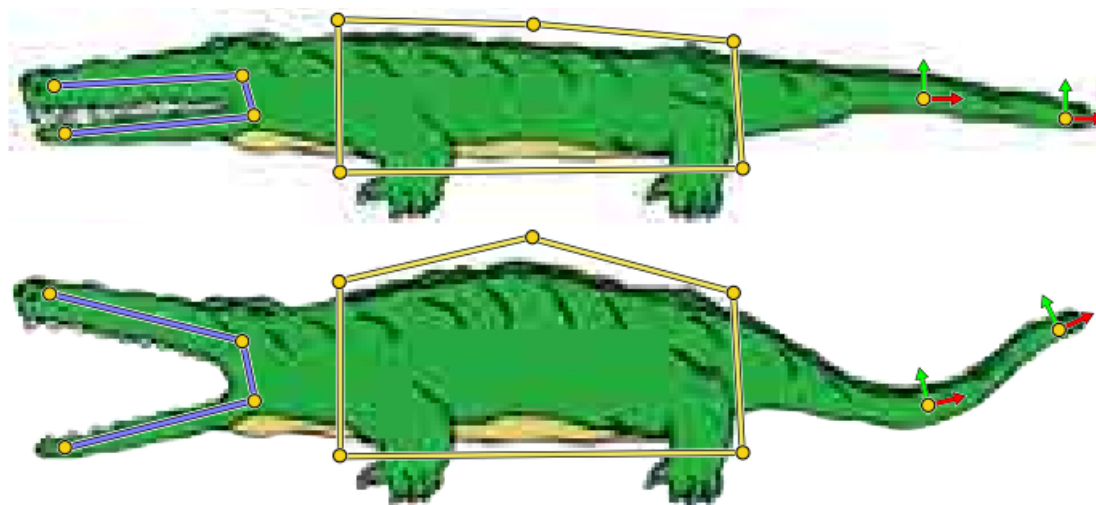


Рисунок 1.3 – Пример выбора вершин на скелете

Плюсы:

- 1) Гибкость в управлении деформацией и анимацией модели.
- 2) Возможность создания более сложных форм и эффектов деформации.
- 3) Мощный инструмент для контроля за определенными частями модели в процессе морфинга [3].

Минусы:

- 1) Дополнительная сложность в настройке и установке весовых значений для каждой вершины.
- 2) Проблемы с поддержкой ненатуральных или сложных деформаций.
- 3) Возможность создания артефактов, таких как рывки или изломы [3].

1.3.3 Морфинг на основе ключевых точек

При использовании данного метода морфинга выделяются несколько промежуточных этапов (точек) перехода одного объекта в другой. После чего, для каждого промежуточного этапа каждой вершине из предыдущего этапа ставится в соответствие вершина из последующего. Далее, для перехода из одного этапа в другой используется интерполяция.

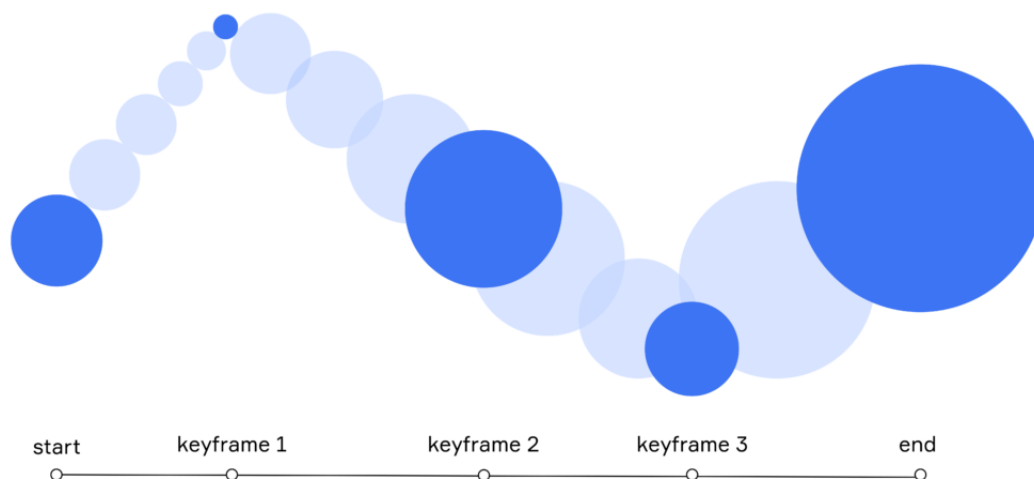


Рисунок 1.4 – Пример выделения ключевых точек

Плюсы:

- 1) Хорошая контролируемость и предсказуемость в анимации и морфинге моделей.
- 2) Возможность создания точных и выразительных анимаций с помощью набора ключевых точек.
- 3) Гибкость в управлении настройками и характеристиками каждой ключевой точки [3].

Минусы:

- 1) Необходимость тщательного планирования и размещения ключевых точек для достижения предполагаемого эффекта.
- 2) Ограничение в контроле над промежуточными состояниями между ключевыми точками.

- 3) Увеличение объема данных при использовании большого числа ключевых точек [3].

Вывод:

Проанализировав алгоритмы морфинга можно прийти к выводу, что наилучшим методом для решения поставленной задачи является линейный морфинг (англ. Linear morphing), так как он является наиболее быстродействующим из представленных алгоритмов, что играет значимую роль при большом количестве примитивов.

1.4 Анализ алгоритмов закрашки

Для сглаживания и добавления реализма изображениям можно использовать алгоритмы закрашки. В данном случае будут рассмотрены 3 основных алгоритма закрашки:

- 1) Простая закрашка.
- 2) Закрашка методом Гуро.
- 3) Закрашка методом Фонга [5].

1.4.1 Простая закрашка

В случае использования простой закрашки считается, что и источник света и наблюдатель находятся в бесконечности, так что диффузная составляющая одинакова (она зависит от угла падения), вектор наблюдения также будет для всех точек одинаковым, то есть зеркальная составляющая также не будет изменяться. Таким образом данный алгоритм является быстрым, однако в случае, если закрашиваемая грань является результатом аппроксимации тела, будет замечен резкий переход между интенсивностями [6].

Плюсы:

- 1) Простая реализация.
- 2) Быстрое выполнение программы.

Минусы:

- 1) Очень низкая реалистичность.

1.4.2 Закраска методом Гуро

В случае использования метода Гуро можно получить сглаженное изображение, для этого сначала определяется интенсивность вершин многоугольника, а затем с помощью билинейной интерполяции вычисляется интенсивность каждого пикселя на сканирующей строке.

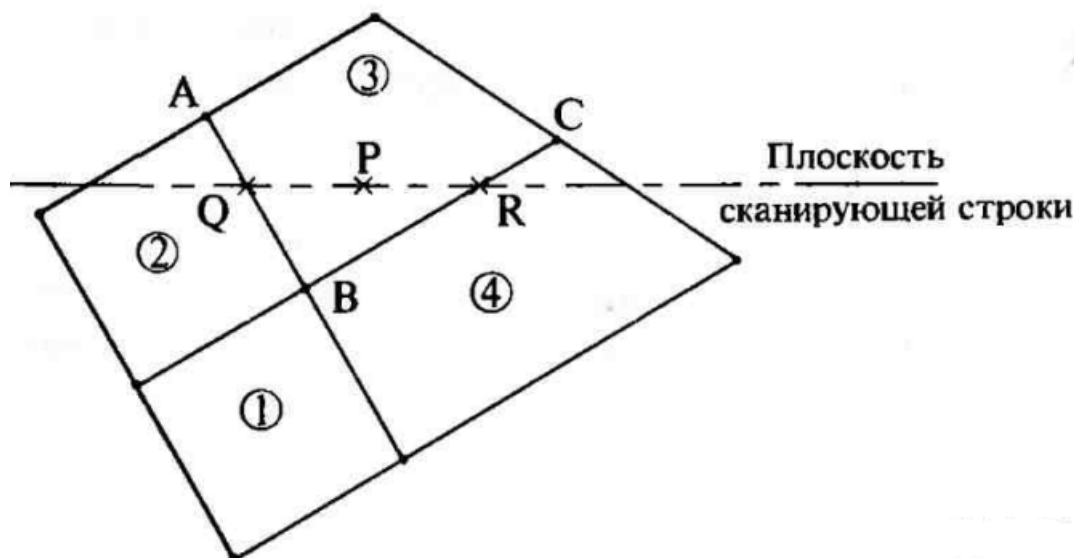


Рисунок 1.5 – Пример полигональной поверхности

Например, рассмотрим участок полигональной поверхности на рисунке 1.5. Значение интенсивности в точке Р определяется линейной интерполяцией значений интенсивностей в точках Q и R. Для получения интенсивности в точке Q можно провести линейную интерполяцию интенсивности в вершинах А и В (см. формула 1.1).

$$I_Q = uI_A + (1 - u) * I_B \quad 0 \leq u \leq 1, t = \frac{AQ}{AB}. \quad (1.1)$$

Таким же образом рассчитывается интенсивность в точке В, после чего интерполяция используется еще раз, для поиска значения интенсивности в точке Р (см. формула 1.2) [6].

$$I_P = tI_Q + (1 - t) * I_R \quad 0 \leq t \leq 1, t = \frac{QP}{QR}. \quad (1.2)$$

Плюсы:

- 1) Получение сглаженного изображения.

- 2) Небольшие трудозатраты [6].

Минусы:

- 1) Появление полос Маха - оптической иллюзии. Она преувеличивает контраст между краями слегка отличающихся оттенков, в местах, где они соприкасаются друг с другом.
- 2) Не учитывает кривизны поверхности (случай, если нормали поверхностей одинаково ориентированы) [6].

1.4.3 Закраска методом Фонга

Закраска Фонга требует больших вычислительных затрат, однако она решает множество проблем метода Гуро. В данном методе вместо интерполяции интенсивностей света, производится интерполяция вектора нормали. Таким образом учитывается кривизна поверхности. В случае картинки 1.5, нормали в соответствующих точках рассчитывалась бы формулами 1.3 [6].

$$\begin{aligned}n_Q &= un_A + (1 - u)n_B & 0 \leq u \leq 1 \\n_R &= wn_B + (1 - w)n_C & 0 \leq w \leq 1 \\n_P &= tn_Q + (1 - t)n_R & 0 \leq t \leq 1.\end{aligned}\tag{1.3}$$

где

$$\begin{aligned}u &= \frac{AQ}{AB} \\w &= \frac{BR}{BC} \\t &= \frac{QP}{QR}\end{aligned}\tag{1.4}$$

Плюсы:

- 1) Получение сглаженного изображения.
- 2) Учет кривизны поверхностей.
- 3) Улучшенная реалистичность зеркальных бликов (по сравнению с методом Гуро) [6].

Минусы:

- 1) Высокая трудозатратность алгоритма [6].

Вывод:

На рисунке 1.6 представлены различные методы закраски. Заметим, что наиболее подходящей закраской для поставленной задачи является закраска Гуро, так как мы будем использовать модели с большим количеством примитивов и нам будет важна производительность. При этом задача не настолько объемная чтобы ограничиваться однотонной закраской.

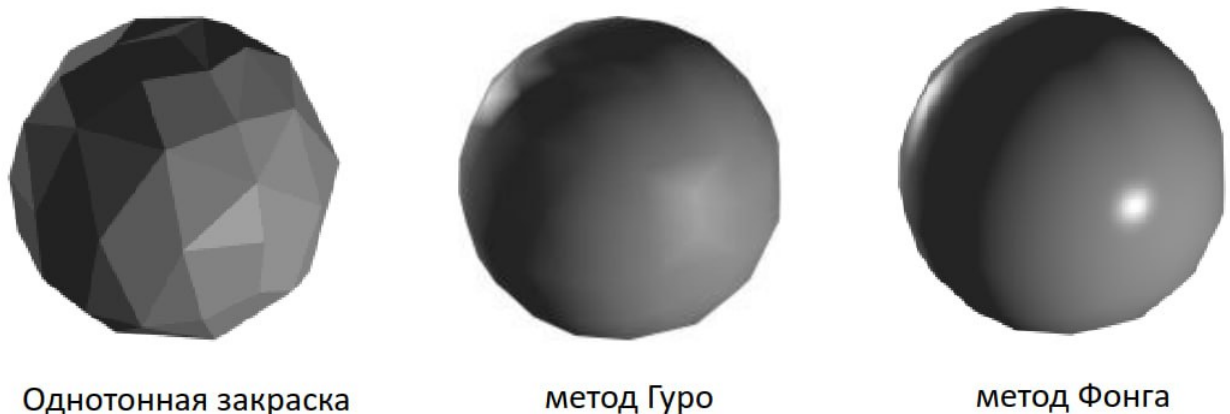


Рисунок 1.6 – Сравнение методов закраски

1.5 Анализ алгоритмов удаления невидимых линий и поверхностей

В рамках данной работы для визуализации процесса морфинга необходимо отсекал невидимые поверхности, для решения поставленной задачи существует множество алгоритмов.

1.5.1 Алгоритм Робертса

Преимуществом этого алгоритма является его эффективное использование мощных и точных математических методов. На первом этапе происходит удаление ребер или граней, которые находятся за самим объектом, а затем рассматриваются грани, закрытые другими объектами на сцене. Если объекты пересекаются друг с другом, удаляются невидимые линии пересечения. Однако недостатком алгоритма является то, что вычислительная сложность растет квадратично с увеличением числа объектов, что может привести к

замедлению работы алгоритма в случае большого числа объектов на сцене. Приведенный недостаток является критически важным для поставленной задачи, однако возможна модификация с использованием сортировки по оси Z , что улучшит производительность алгоритма [6].

1.5.2 Алгоритм Z-буфера

Данный алгоритм позволяет не только удалять невидимые линии и поверхности, но и визуализировать их пересечение. При работе данного алгоритма происходит поиск ближайшего объекта к каждому пикселю экрана. Сцена может быть любой сложности, а так как размер изображения ограничен размером экрана, сложность алгоритма зависит линейно от количества поверхностей.

У этого алгоритма есть и существенные недостатки. Он требует большой объем памяти для своей реализации, так как необходимо хранить буфер кадра и Z-буфер. Кроме того, его сложно использовать для реализации эффектов, таких как прозрачность и освещение, ввиду отсутствия наблюдения за лучами света [6].

1.5.3 Обратная трассировка лучей

В данном алгоритме предполагается, что точка зрения находится в бесконечности на положительной полуоси Z , что делает все световые лучи параллельными этой оси. Алгоритм отслеживает траекторию каждого луча и определяет, с какими объектами сцены, если они существуют, он пересекается. В случае если пересечение существует, то из данного луча образуются преломленный и отраженный.

Благодаря этому алгоритму можно достичь эффектов, таких как отражение и преломление, что делает изображение более реалистичным. Каждый пиксель изображения вычисляется независимо от других, что обеспечивает высокую параллельность расчетов.

Однако алгоритм обратной трассировки лучей имеет и недостатки. Данный алгоритм очень ресурсоемок, так как каждый луч порождает несколько отраженных и преломленных лучей, которые также необходимо хранить и обрабатывать. Также для поиска пересечения одного луча необходимо проанализировать все имеющиеся на сцене примитивы, что значительно влияет

на вычислительную сложность алгоритма [6].

Вывод

Проанализировав алгоритмы удаления невидимых линий и поверхностей, заметим что для поставленной задачи наиболее всего происходит алгоритм Z-буфера, так как он может работать со сценами любой сложности и не требует больших вычислительных мощностей для сцен с множеством объектов, что необходимо при реализации морфинга.

Выводы из аналитического раздела

В данном разделе были проанализированы алгоритмы морфинга, удаления невидимых линий, а также методы закраски. Таким образом были выбраны:

- 1) **Алгоритм морфинга** - линейный морфинг.
- 2) **Метод закраски** - закрашка Гуро.
- 3) **Алгоритм удаления невидимых линий** - алгоритм Z-буфера.

Входными данными для полученной системы будут являться:

- 1) Вершины и ребра модели.
- 2) Локальные координаты модели
- 3) Угол поворота модели по каждой оси.
- 4) Текущий масштаб модели.
- 5) Положение камеры.
- 6) Положение источника света.

2 Конструкторский раздел

В данном разделе будут более подробно рассмотрены выбранные в предыдущем разделе методы и алгоритмы, а также предоставлены требования к программному обеспечению.

2.1 Требования к программному обеспечению

Программа должна:

- 1) Загружать модели из .obj файлов.
- 2) Визуализировать трехмерные каркасные модели.
- 3) Визуализировать морфинг двух трехмерных моделей.
- 4) Преобразовывать модель: поворот, масштабирование и сдвиг.
- 5) Изменять положение источника света.
- 6) Изменять положение камеры.
- 7) Изменять тип вершин, ребер.
- 8) Изменять размер вершин и ребер.

2.2 Считывание моделей

Для задания моделей был выбран один из наиболее популярных и универсальных методов – .obj файлы. Рассмотрим этапы считывания данных:

- 1) Открытие файла .obj

Программа открывает файл, предоставляя путь к нему в качестве входного параметра.

- 2) Чтение вершин

Из файла считываются строки, содержащие координаты вершин. Каждая строка представляет собой набор координат в формате 'v x y z', где x, y и z - координаты вершины.

3) Чтение граней

Происходит чтение строк, представляющих грани. Формат строки - 'f v1 v2...vn', где v1, v2...vn - индексы вершин, образующих грань.

4) Обработка и хранение данных

Прочитанные вершины и грани сохраняются в структуры данных, предназначенные для хранения геометрических данных модели. Вершины сохраняются в массиве, а грани - в структуре, содержащей информацию о связи вершин и порядке их соединения.

2.3 Общий алгоритм построения изображения

Рассмотрим алгоритм построения кадра, он представлен на рисунке 2.1. Для визуализации объектов используется алгоритм Z-буфера, представлен на рисунке 2.2. Он был рассмотрен для полигонов поверхностной модели. Особое внимание стоит уделить расчету новых нормалей полигонов: это долгий процесс, чтобы его оптимизировать предлагается ввести список полигонов для каждой вершины, каждой вершине в данной структуре данных будет соответствовать полигоны, частями которых она является. После проведения данной подготовки для расчета нормали в соответствующей вершине достаточно будет рассчитать среднее из нормалей соответствующих ей в списке полигонов. Данный метод позволит многократно ускорить выполнение алгоритма [7].

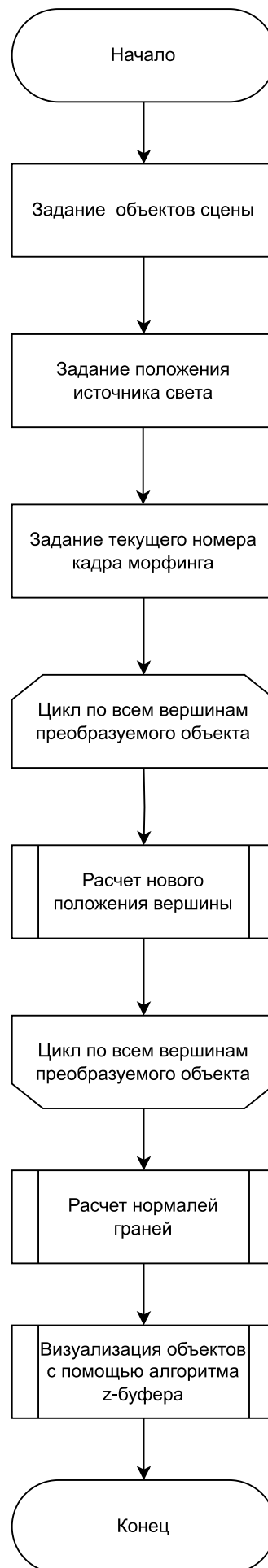


Рисунок 2.1 – Общий алгоритм построения кадра

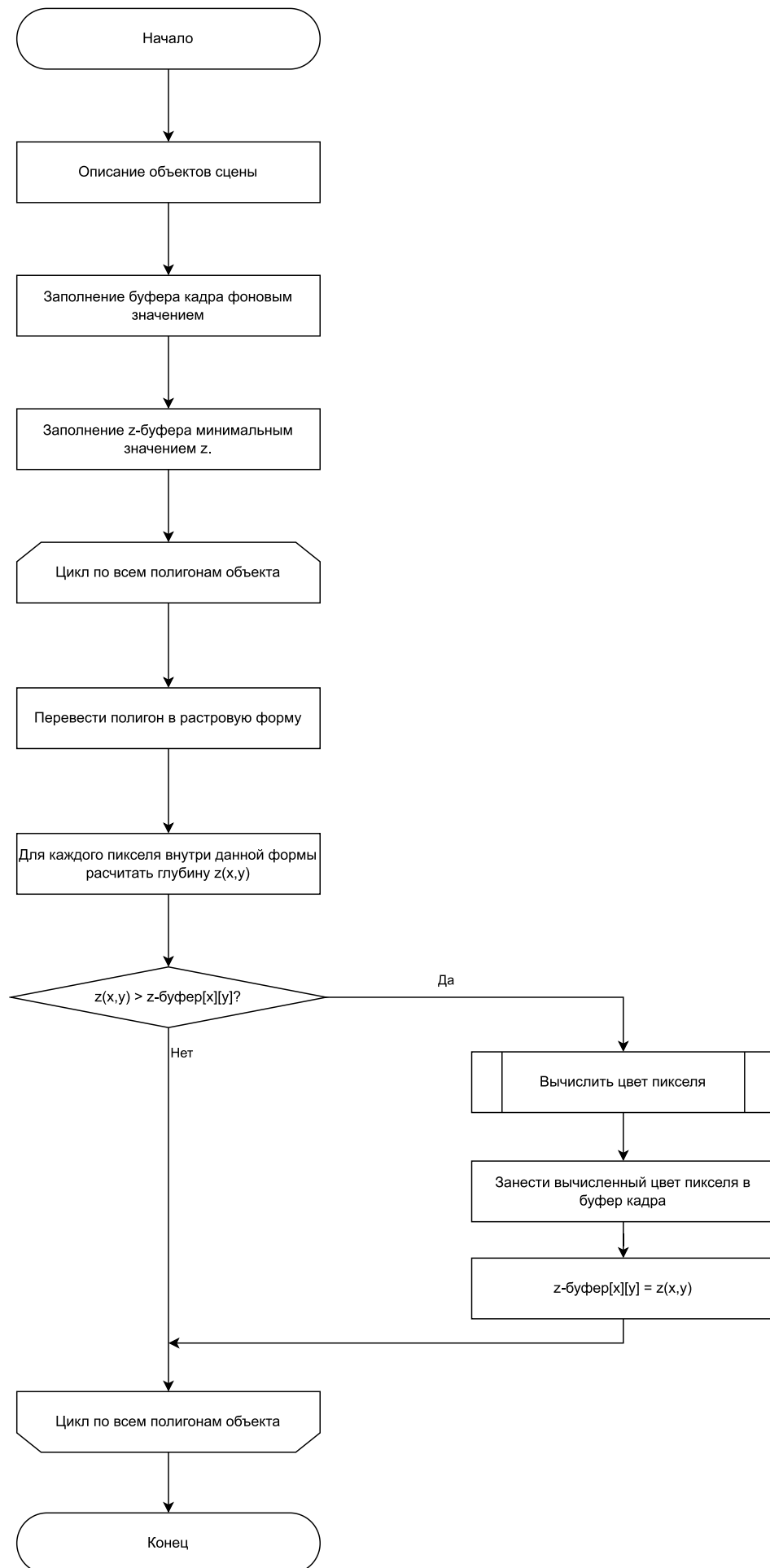


Рисунок 2.2 – Алгоритм Z-буфера

Математические соображения о алгоритме Гуро приведены в 1.4. Данный алгоритм позволяет получить довольно гладкое изображение при малых вычислительных затратах, что важно при необходимости постоянно пересчета нормалей при морфинге. Необходимо отметить что интенсивность в вершинах полигона будет определяться как скалярное произведение вектора светового луча (вектора направленного из вершины к источнику света) и нормали в вершине.

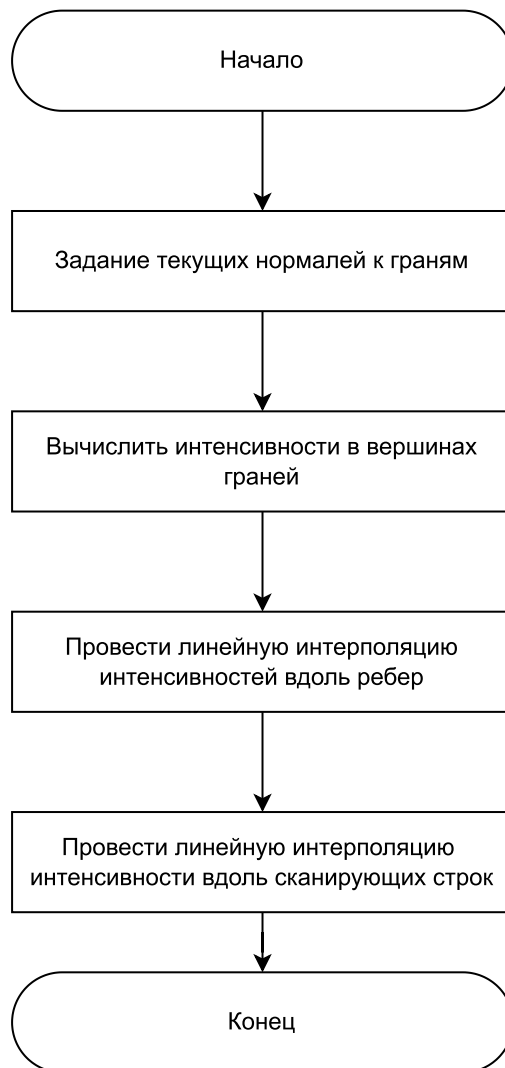


Рисунок 2.3 – Алгоритм закраски по Гуро

2.4 Алгоритм морфинга

Рассмотрим алгоритм морфинга двух объектов. Он представлен на рисунке 2.4.

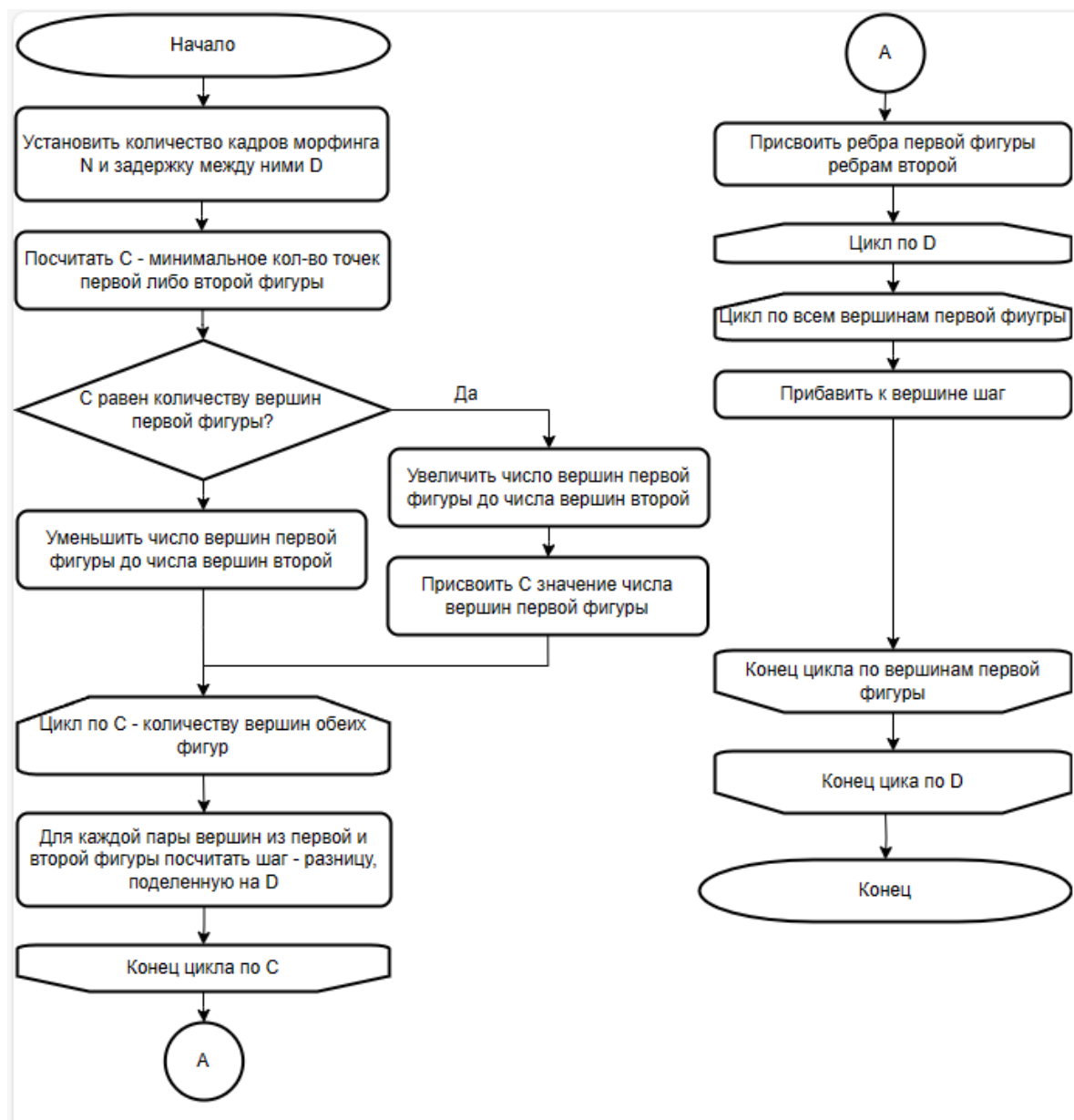


Рисунок 2.4 – Сравнение методов закрашки

Выводы из конструкторского раздела

В данном разделе была разобрана реализация выбранного алгоритма построения изображения, а также рассмотрены алгоритмы Z-буфера, закрашки по Гуро и морфинга двух фигур.

3 Технологический раздел

3.1 Средства реализации

Для реализации программного обеспечения был выбран язык C++ по следующим причинам:

- 1) Он поддерживает объектно-ориентированный стиль программирования.
- 2) В предыдущих семестрах на данном языке были успешно разработаны различные приложения, что упрощает написание новых программ связи с имеющимся опытом.
- 3) Язык обладает высокой производительностью, что важно при морфинге крупных объектов.

В качестве среды разработки был выбран Qt Creator, поскольку он достаточно удобен и обладает всем необходимым функционалом для реализации поставленной задачи, в том числе наличие библиотек Qt позволяет реализовать пользовательский интерфейс, а Qt Creator имеет средства для упрощения разработки этого интерфейса.

3.2 Выбор структур данных

В данной работе необходимо создать следующие структуры данных:

- 1) Структура, характеризующая перемещение модели.
- 2) Структура, характеризующая вращение модели.
- 3) Структура, характеризующая масштабирование модели.
- 4) Класс, описывающий точку.
- 5) Класс, описывающий точки.
- 6) Структура, описывающая ребро.
- 7) Класс, описывающая ребра.
- 8) Структура, описывающая грань.

- 9) Класс, описывающий грани.
- 10) Класс, описывающий фигуру.
- 11) Класс, описывающий отрисовываемое полотно.
- 12) Класс, необходимый для взаимодействия пользователя с программой.

3.3 Структура программы

На рисунке 3.1 представлена структура программы в виде UML-диаграммы.

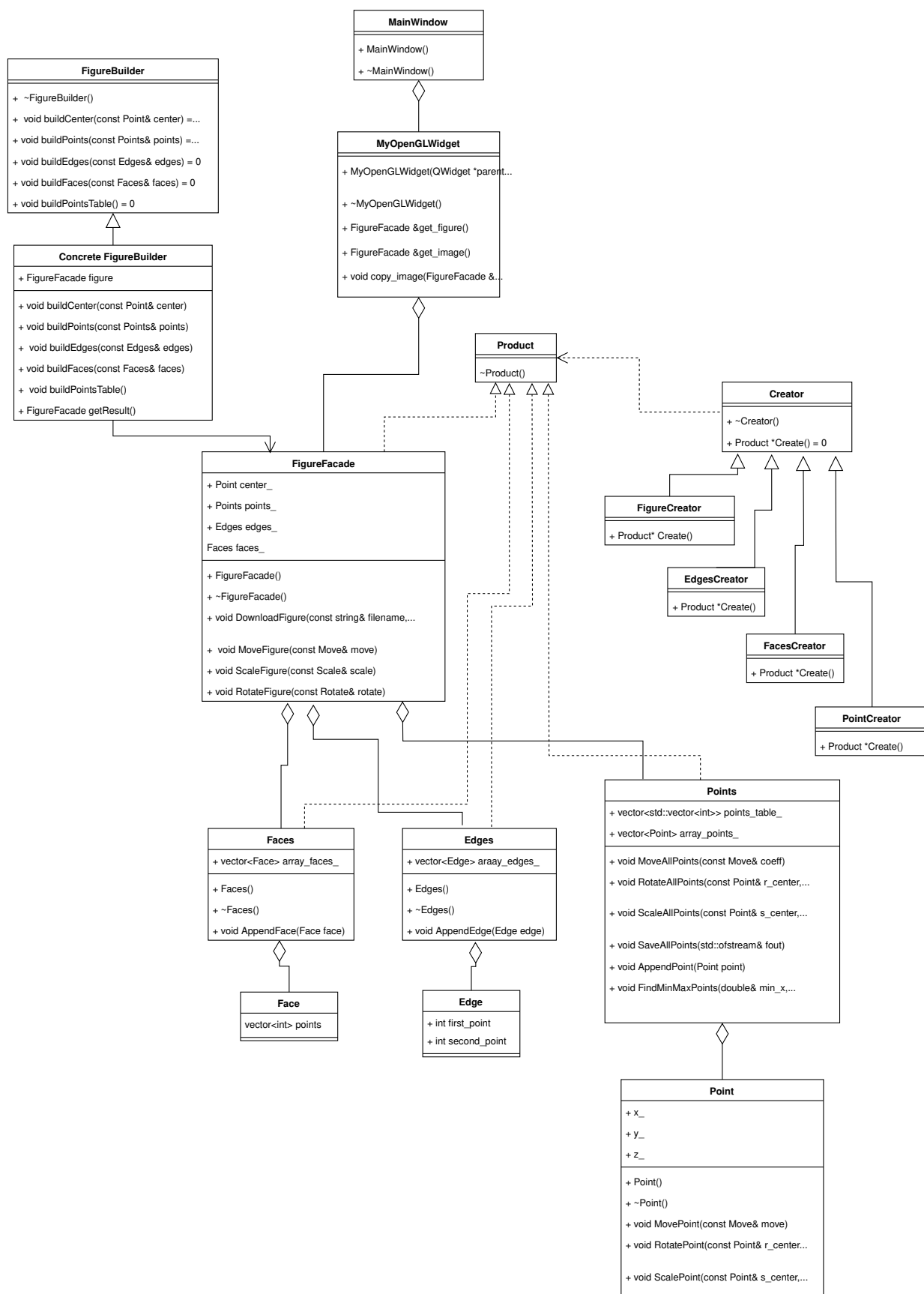


Рисунок 3.1 – Структура программы

3.4 Реализация алгоритмов

4 Исследовательский раздел

блаблабла

ЗАКЛЮЧЕНИЕ

Была достигнута цель данной работы - разработано программное обеспечение для морфинга трехмерных моделей. Для достижения поставленной цели были решены следующие задачи:

- 1) Формализовать представление объектов и описать их.
- 2) Проанализировать алгоритмы морфинга трехмерных моделей и выбрать наилучшие для достижения цели.
- 3) Выбрать средства реализации алгоритма.
- 4) Реализовать выбранный алгоритм.
- 5) Реализовать интерфейс.
- 6) Исследовать характеристики полученных алгоритмов.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Алгоритм Морфинга для создания сферических панорам [Электронный ресурс]. — Режим доступа: <https://clck.ru/35ghkJ> (дата обращения: 20.07.2023).
2. DMorph. — Режим доступа: <https://www.graphicon.ru/oldgr/courses/cg/assigns/2004/hw4/g5078.pdf> (дата обращения 25.07.23).
3. A Comparison of Mesh Morphing Methods. — Режим доступа: <https://clck.ru/35on9b> (дата обращения 25.07.23).
4. Weight morphing. — Режим доступа: <https://igl.ethz.ch/projects/bbw/bounded-biharmonic-weights-siggraph-2011-compressed-jacobson-et-al.pdf> (дата обращения 01.08.23).
5. Методы закраски. — Режим доступа: <https://clck.ru/35qmXd> (дата обращения 24.09.23).
6. *Роджерс Д.* Алгоритмические основы машинной графики //. — Москва: Мир, 1989.
7. Real-time Object Morphing. — Режим доступа: <https://www.gamedev.net/reference/articles/article670.asp> (дата обращения 20.07.23).