

# СОДЕРЖАНИЕ

<b>ВВЕДЕНИЕ</b>	<b>5</b>
<b>1 Аналитическая часть</b>	<b>6</b>
1.1 Явление морфинга	6
1.2 Формализация объектов сцен	6
1.3 Выбор модели описания объекта	7
1.3.1 Аналитическая модель	7
1.3.2 Полигональная модель	7
1.3.3 Воксельная модель	7
1.3.4 Сравнение моделей описания объектов	8
1.4 Основные этапы морфинга	8
1.4.1 Установление соответствия между объектами	8
1.4.2 Параметризация	8
1.4.3 Создание общего представления	10
1.4.4 Интерполяция геометрии	10
1.5 Анализ алгоритмов удаления невидимых линий и поверхностей	11
1.5.1 Алгоритм Робертса	11
1.5.2 Алгоритм, использующий $z$ -буфер	12
1.5.3 Алгоритм художника	12
1.5.4 Сравнение алгоритмов	13
1.6 Модель освещения	14
1.7 Анализ алгоритмов закрашки	15
1.7.1 Однотонная закрашка	15
1.7.2 Закрашка методом Гуро	15
1.7.3 Закрашка методом Фонга	16
1.7.4 Сравнение алгоритмов закрашки	16

<b>2</b>	<b>Конструкторская часть . . . . .</b>	<b>18</b>
2.1	Требования к программному обеспечению . . . . .	18
2.2	Используемые типы и структуры данных . . . . .	18
2.3	Математические основы алгоритмов . . . . .	20
2.3.1	Барицентрические координаты . . . . .	20
2.3.2	Поиск пересечения дуг на единичной сфере . . . . .	21
2.3.3	Поиск точки внутри объекта . . . . .	22
2.4	Описание алгоритмов . . . . .	23
<b>3</b>	<b>Технологическая часть . . . . .</b>	<b>28</b>
<b>4</b>	<b>Исследовательская часть . . . . .</b>	<b>29</b>
4.1	Вывод . . . . .	29
	<b>ЗАКЛЮЧЕНИЕ . . . . .</b>	<b>30</b>
	<b>СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ . . . . .</b>	<b>31</b>

# ВВЕДЕНИЕ

TODO: intro.

# 1 Аналитическая часть

## 1.1 Явление морфинга

Слово «морфинг» происходит от слова «метаморфоза», которое, согласно Оксфордскому словарю [?], имеет следующее значение: "Процесс, в ходе которого кто-то/что-то полностью превращается во что-то другое".

Таким образом, в случае трехмерных объектов термин «морфинг» можно интерпретировать как построение последовательности кадров, соответствующей постепенному переходу между двумя различными объектами, так называемыми исходными (начальными) и целевыми (конечными) моделями. На рисунке 1.1 представлен пример морфинга трехмерных объектов.

Цель морфинга заключается в вычислении преобразования, обеспечивающего визуально приятный переход между исходной и целевой формами [1]. TODO: возможно здесь добавить про монотонность, сохранение точек интереса и тд (в зависимости от того, как будет работать feature preservation)

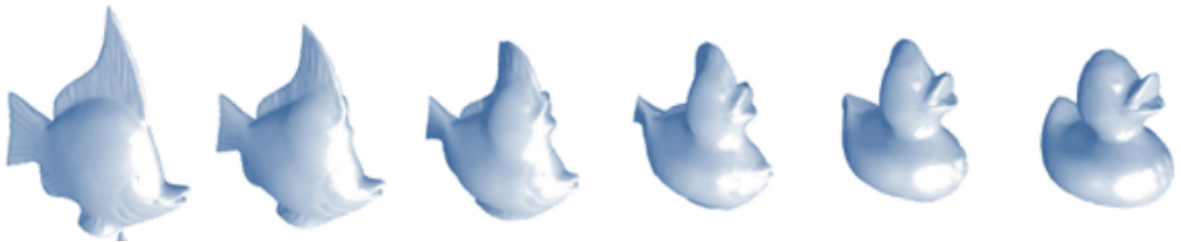


Рисунок 1.1 — Пример последовательности кадров морфинга

## 1.2 Формализация объектов сцен

На сцене выбора начальных и конечных объектов могут присутствовать 2 фрукта: начальный и конечный. Фрукт (объект) задается моделью описания объектов, данными для этой модели, оптическими характеристиками поверхности.

На сцене просмотра морфинга находится результат морфинга (задается исходным и целевым объектами, стадией морфинга).

На обеих сценах могут присутствовать следующие типы объектов:

- источник света (задается положением в пространстве, цветом и интенсивностью света);
- наблюдатель (задается положением в пространстве, точкой в пространстве, на которую направлен взгляд, направлением верха обзора TODO);

### 1.3 Выбор модели описания объекта

Наиболее распространенные модели описания трехмерных объектов в компьютерной графике: *аналитическая, полигональная модель, воксельная модель* [2].

#### 1.3.1 Аналитическая модель

Аналитическая модель представляет собой описание поверхности математическими формулами [2]. Обычно поверхность задается уравнением вида  $z = f(x, y)$  или  $F(x, y, z) = 0$ .

Отличительные черты [2]:

- легкая процедура расчета координат точек, нормалей;
- небольшой объем информации для описания форм;
- сложные формулы, которые медленно вычисляются компьютером;
- задание объекта набором поверхностей, если невозможно описать его аналитически;
- отсутствие погрешности при задании сферического объекта;

#### 1.3.2 Полигональная модель

В полигональной модели информация об объекте состоит из следующих компонентов [2]:

- вершина — точка  $((x, y, z))$  в декартовой системе координат;
- отрезок прямой — задается двумя вершинами;
- полилиния — задается несколькими отрезками прямой;
- полигон — описывает плоскую грань объемного объекта в виде замкнутой линии;

Несколько граней (полигонов) составляют объемный объект в виде полигональной поверхности, также называемой «полигональной сеткой».

#### 1.3.3 Воксельная модель

TODO

### 1.3.4 Сравнение моделей описания объектов

TODO: а надо ли мне вообще выбирать, если у меня в ТЗ написано НИЗКОПОЛИГОНАЛЬНЫЕ ОБЪЕКТЫ???? TODO: можно сравнить по критериям: сложность вычисления нормали, изображение любых объектов, сохранение качества при увеличении

Среди всех моделей наиболее подходящей является полигональная модель, т. к. с помощью нее можно описать объекты любой сложности, и сохраняет качество при увеличении, что необходимо в задаче интерактивной визуализации морфинга, поэтому использованная будет именно она.

## 1.4 Основные этапы морфинга

### 1.4.1 Установление соответствия между объектами

3D-сетки часто различаются по топологии (число вершин/граней и связность), поэтому прямое сопоставление поверхностей затруднено. Соответствие получают косвенно через *параметризацию* — биективное отображение поверхности сетки в общую параметрическую область  $D$ , обычно единичный диск для незамкнутых сеток (*плоская параметризация*) или единичная сфера для замкнутых (*сферическая параметризация*) [1].

Сферическая параметризация применяется к замкнутым поверхностям нулевого рода [1], или не имеющим отверстий, коими являются фрукты в рамках данной работы.

### 1.4.2 Параметризация

Выделяют 3 основных подхода к параметризации сеток: *плоская, сферическая, разбиение на участки с плоской параметризацией* [1, 3].

Плоская параметризация применима для незамкнутых сеток [1, 3], поэтому она непригодна для морфинга замкнутых объектов, таких, как фрукты.

Если исходный и целевой объекты имеют существенно различающиеся формы или представляют собой поверхности разного рода, т. е. имеют разное число отверстий, применяют предварительное разбиение на плоские участки [1]. Пользователь вручную разбивает исходную и целевую сетки на участки и задаёт их соответствие; затем для каждого участка выполняется плоская

параметризация. Данный метод требует значительного участия пользователя, от которого зависит качество результата, поэтому в настоящей работе он не рассматривается.

Для сферической параметризации произвольных поверхностей нулевого рода, применяется метод релаксации, основанный на итеративном уточнении положений вершин [3]. Процесс релаксации сетки представлен на рисунке 1.2.

В качестве начального состояния строят грубую проекцию сетки на единичную сферу: выбирают любую внутреннюю точку модели в качестве центра сферы и проецируют все вершины на её поверхность (нормализуют радиус-векторы). Полученная начальная конфигурация обычно содержит значительные искажения и грани с неправильной ориентацией. Процесс релаксации продолжают до тех пор, пока все грани не приобретут корректную ориентацию — внешняя сторона каждой грани должна быть обращённой наружу сферы [3].

На каждом раунде релаксации вершины сдвигаются к центру масс своих соседей:

$$v_i^{k+1} = \frac{\sum_{j \in N(i)} v_j^k}{\left\| \sum_{j \in N(i)} v_j^k \right\|}, \quad (1.1)$$

где

- $v_i^{k+1}$  — положение  $i$ -ой вершины после  $k$ -го раунда релаксации;
- $v_i^k$  — положение  $i$ -ой вершины на момент  $k$ -го раунда релаксации;
- $N(i)$  — множество вершин, смежных с  $i$ -ой.

Для предотвращения коллапса всех вершин в одну точку после каждой итерации выполняется ре-центрирование всей сетки относительно начала координат [3]:

$$v'_i = v_i - \frac{\sum_{j=0}^n v_j}{n}, \quad (1.2)$$

где

- $v'_i$  — новое положение  $i$ -ой вершины;
- $n$  — количество вершин.

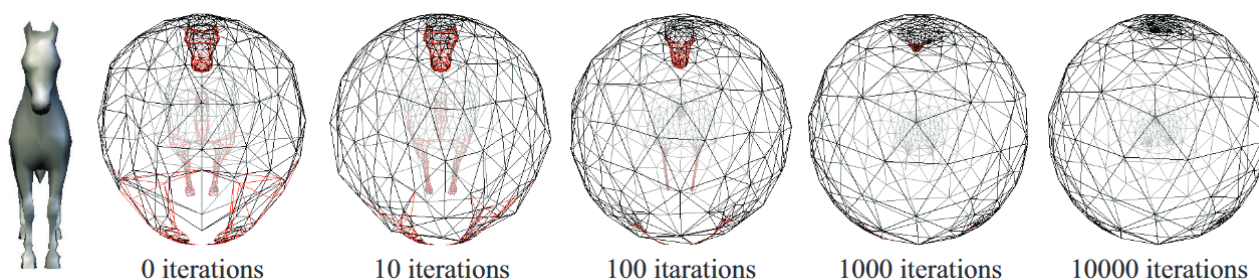


Рисунок 1.2 — Процесс релаксации сетки, красными отмечены грани с неправильной ориентацией

### 1.4.3 Создание общего представления

После того как установлено соответствие, необходимо создать единую структуру, которая будет использоваться для всех промежуточных форм. Обычно для этого сроят *суперсетку* — сетку, содержащую вершины исходной и целевой сетки, а также их точки пересечения [1, 3] (пример на рисунке 1.3).

На параметрической сфере сетки накладываются друг на друга; в местах пересечения ребер создаются новые вершины, что формирует объединенную сетку, которая может принимать форму, как исходной так и целевой модели; поскольку полученная сетка обычно не является треугольной, выполняют её триангуляцию.

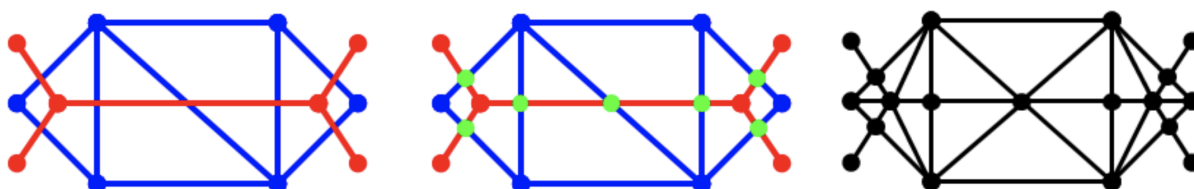


Рисунок 1.3 — Создание суперсетки

### 1.4.4 Интерполяция геометрии

Заключительный этап — вычисление траекторий движения вершин общего представления из их начальных положений до конечных. Для каждой вершины суперсетки необходимо определить ее координаты на исходной и целевой моделях. Процесс генерации промежуточных кадров сводится к интерполяции этих координат.

Положение вершины на каждой из моделей определяется с помощью барицентрических координат в параметрическом пространстве [1, 3]. Алгоритм



сводится к следующим шагам:

Процесс вычисления состоит из следующих шагов:

- 1) Поиск содержащего треугольника: Для каждой определяется треугольник исходной (или целевой) сетки, в который она попадает в общем параметрическом пространстве (на сфере).
- 2) Вычисление барицентрических координат данной вершины относительно вершин найденного треугольника.
- 3) Барицентрические координаты применяются к вершинам соответствующего треугольника в мировых координатах.

После получения начальных и конечных положений  $P_0$  и  $P_1$  для каждой вершины траектории генерируют посредством линейной интерполяции [1, 3]. Положение вершины в момент времени  $t \in [0, 1]$  вычисляется по формуле (??).

$$P(t) = (1 - t) \cdot P_0 + t \cdot P_1 \quad (1.3)$$

TODO: добавить про интерполяцию материала и нормалей

## **1.5 Анализ алгоритмов удаления невидимых линий и поверхностей**

Алгоритмы удаления невидимых линий и поверхностей служат для удаления ребер, поверхностей или объемов, которые видимы или невидимы для наблюдателя, находящегося в заданной точке пространства [4].

Рассмотрим следующие алгоритмы: *алгоритм Робертса, алгоритм, использующий z-буфер, алгоритм трассировки лучей*.

### **1.5.1 Алгоритм Робертса**

Данный алгоритм применим только к выпуклым телам. Если обрабатываемое тело невыпуклое — его необходимо предварительно разбить на выпуклые [4].

Алгоритм состоит из следующих этапов [4]:

- 1) Удаление граней, экранируемых самим телом.
- 2) Удаление граней, экранируемых другими телами.
- 3) Удаление линий пересечения тел, экранируемых самими телами.

Ассимптотическая оценка трудоемкости:  $O(N^2)$ , где  $N$  — количество граней. TODO: проверить

### 1.5.2 Алгоритм, использующий $z$ -буфер

Идея  $z$ -буфера является обобщением идеи буфера кадра. Буфер кадра используется для запоминания атрибутов (интенсивности) каждого пикселя в пространстве изображения.  $z$ -буфер — это отдельный буфер глубины, используемый для запоминания координаты  $z$  каждого видимого пикселя в пространстве изображения [4].

Этапы работы алгоритма [4]:

- 1) Заполнить буфер кадра фоновым значением.
- 2) Заполнить  $z$ -буфер минимальным значением глубины.
- 3) Выполнить переход в пространство изображения.
- 4) Для каждого пикселя  $(x, y)$ , принадлежащего телу вычислить его глубину  $z(x, y)$ .
- 5) Если глубина  $z(x, y) > z\text{-буфер}(x, y)$ , то записать атрибут текущего тела в  $\text{буфер-кадра}(x, y)$ , записать глубину  $z(x, y)$  в  $z\text{-буфер}(x, y)$ .

Ассимптотическая оценка трудоемкости:  $O(N)$ , где  $N$  — количество граней. TODO: проверить

### 1.5.3 Алгоритм художника

Идея алгоритма состоит в том, чтобы подобно художнику отрисовывать объекты по мере их приближения к наблюдателю.

Основные этапы алгоритма [4]:

- 1) Отсортировать грани по минимальному или максимальному значению глубины.
- 2) Отрисовать грани в отсортированном порядке.

Простая сортировка не всегда дает корректный список приоритетов, тогда приходится использовать дополнительные методы разрешения конфликтов [4].

Алгоритм не справляется со случаями циклического перекрытия и пересечения многоугольников.

Ассимптотическая оценка трудоемкости:  $O(N)$ , где  $N$  — количество граней, однако стоит дополнительно учитывать трудоемкость предварительной сор-

тировки.

## **Алгоритм трассировки лучей**

Идея метода заключается в том, что наблюдатель видит любой объект посредством испускаемого неким источником света, который падает на этот объект и некоторым путем доходит до наблюдателя. Однако обычно используют алгоритм обратной трассировки лучей, в котором лучи пускаются от наблюдателя [4].

Полагая, что объекты сцены уже находятся в пространстве изображения, выполняются следующие этапы для каждого пикселя [4]:

- 1) Составить уравнение отслеживаемого луча.
- 2) Найти пересечение луча со всеми гранями.
- 3) Среди всех пересечений найти ближайшее к наблюдателю и использовать атрибуты объекта, которому соответствует пересечение для определения цвета пикселя. Если пересечений нет — закрасить пиксель цветом фона.

Асимптотическая оценка трудоемкости:  $O(WHN)$ , где  $W$  — ширина экрана в пикселях,  $H$  — высота экрана в пикселях,  $N$  — количество граней.

### **1.5.4 Сравнение алгоритмов**

В таблице 1.1 представленные результаты сравнения алгоритмов и использованы следующие обозначения:

- Р — алгоритм Робертса;
- ЗБ — алгоритм, использующий  $z$ -буфер;
- Х — алгоритм Художника;
- ТЛ — алгоритм трассировки лучей.

Таблица 1.1 — Сравнение алгоритмов удаления невидимых линий и поверхностей

	<b>Р</b>	<b>ЗБ</b>	<b>Х</b>	<b>ТЛ</b>
Совместимость с любыми телами	-	+	+	+
Возможность использования без сортировки	+	+	-	+
Возможность учета прозрачности	-	+	+	+
Асимптотическая оценка трудоемкости	$O(N^2)$	$O(N)$	$O(N)$	$O(WHN)$

Среди всех рассмотренных алгоритмов, алгоритм, использующий  $z$ -буфер подходит больше остальных, т. к. он имеет лучшую асимптотическую оценку трудоемкости и применим к любым телам, что необходимо при решении задачи визуализации морфинга.

## 1.6 Модель освещения

В компьютерной графике наиболее распространенными являются две модели освещения: *локальная* и *глобальная* [4].

Локальная модель учитывает только свет, падающий от источника (источников), и ориентацию поверхности [4].

Глобальная модель освещения учитывает также свет, отраженный от других объектов сцены или пропущенный через них [4].

Поскольку на сцене будет находиться только один объект, то будет использована локальная модель освещения.

Интенсивность  $I$  в точке  $P$  в локальной модели вычисляется по формуле (??) [4].

$$I = k_a I_a + \frac{I_l}{d + K} [k_d(\hat{\mathbf{n}} \cdot \hat{\mathbf{L}}) + k_s(\hat{\mathbf{R}} \cdot \hat{\mathbf{S}})^\alpha], \quad (1.4)$$

где

$k_a$  — коэффициент TODO;

$I_a$  — интенсивность фонового освещения;

$I_l$  — интенсивность источника света;

$d$  — расстояние от источника света до точки  $P$ ;

$K$  — добавка уменьшения интенсивности света с расстоянием (выбирается из эстетических предпочтений);

$k_d$  — коэффициент диффузного отражения поверхности;

$\mathbf{n}$  — вектор нормали к поверхности в точке  $P$ ;

$\mathbf{L}$  — вектор, обратный вектору падения луча;

$k_s$  — коэффициент зеркального отражения поверхности;

$\mathbf{R}$  — вектор, отраженного луча;

$\mathbf{S}$  — вектор, направленный на наблюдателя из точки  $P$ ;

$\alpha$  — степень, аппроксимирующая пространственное распределение зеркально отраженного света.

## 1.7 Анализ алгоритмов закрашки

Основными алгоритмами закрашки в компьютерной графике являются: *однотонная закрашка*, *закрашка методом Гуро*, *закрашка методом Фонга* [4].

### 1.7.1 Однотонная закрашка

При однотонной закрашке для каждой грани (многоугольника) полигональной поверхности вычисляется один уровень интенсивности, с которым закрашивается вся грань. В результате такой закрашки изображенный состоит из отдельных многоугольников и объект выглядит, как многогранник [4].

### 1.7.2 Закрашка методом Гуро

Этот метод предназначен для создания иллюзии гладкой криволинейной поверхности, описанной в виде многогранников или полигональной сетки с плоскими гранями [2, 4, 5]

Метод Гуро основан на интерполяции интенсивности каждого пикселя при закрашке. Закрашивание граней по методу Гуро осуществляется в четыре этапа [2]:

- 1) Определение нормали к каждой грани.
- 2) Определение нормалей в вершинах путем усреднения нормалей прилежащих граней.
- 3) На основе нормалей в вершинах вычисляются значения интенсивностей в вершинах согласно выбранной модели освещения.

- 4) Закрашиваются полигоны граней цветом, соответствующим интерполяции значений интенсивности в вершинах.

### 1.7.3 Закраска методом Фонга

Аналогичен методу Гуро, но при использовании метода Фонга для определения цвета в каждой точке интерполируются не интенсивности отраженного света, а векторы нормалей. При этом значение интенсивность вычисляется в каждом внутреннем пикселе грани [2, 4].

### 1.7.4 Сравнение алгоритмов закрашки

Визуальное сравнение алгоритмов представлено на рисунке 1.4.

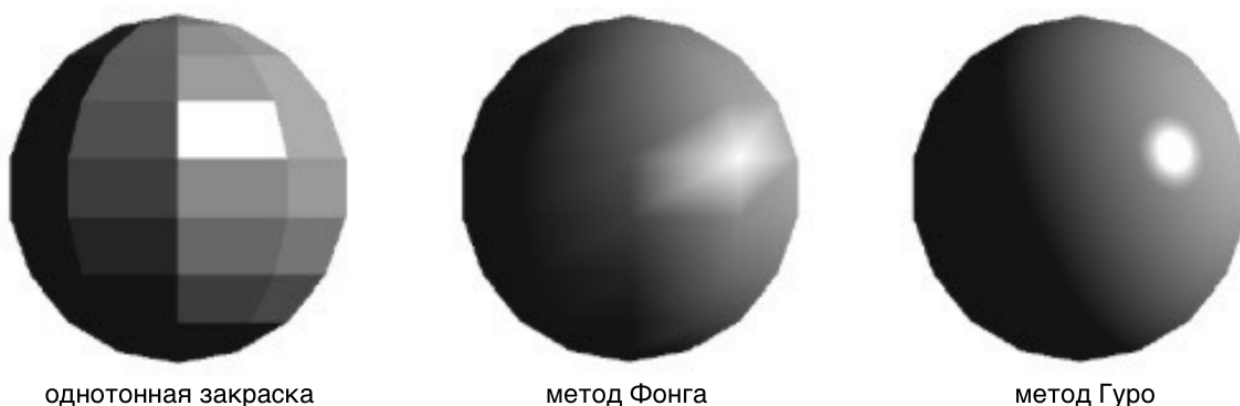


Рисунок 1.4 — Визуальное сравнение методов закрашки

Поскольку объекты сцены (фрукты) представлены низкополигональными объектами, предпочтительной является однотонная закрашки, обеспечивающая дискретность освещения между смежными гранями. В результате моффринга часто возникает множество полигонов, составляющих одну грань, из-за чего при однотонной закрашке возникают световые артефакты, как на рисунке 1.5, а.

Для устранения этой проблемы будет использована модифицированная версия закрашки Гуро: значение интенсивности вершин в пределах одной грани вычисляется с помощью нормали к этой грани, а не нормалей в вершинах. Это обеспечивает равномерное распределение интенсивности по грани и предотвращает сглаживание рёбер, что продемонстрировано на рисунке 1.5, б.

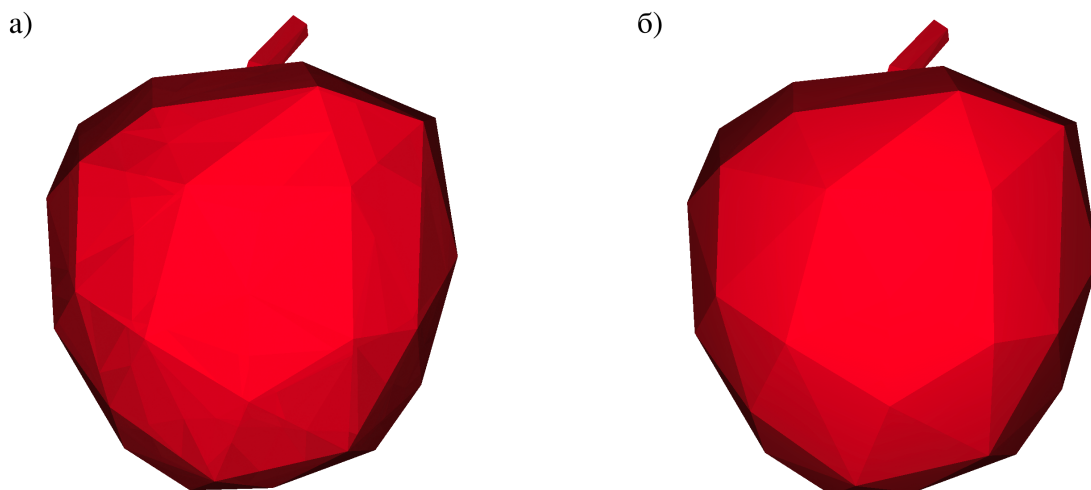


Рисунок 1.5 — Результат морфинга с использованием различных алгоритмов закраски: а) однотонная закраска; б) закраска модифицированным методом Гуро

## Вывод

В аналитической части были рассмотрены основные этапы морфинга, проанализированны модели представления объектов и алгоритмы решения основных задач компьютерной графики. По результатам проведенного анализа были выбраны: полигональное представление модели, алгоритм, использующий  $z$ -буфер для удаления невидимых линий и поверхностей, простая модель освещения, модифицированная закраска Гуро.

## 2 Конструкторская часть

### 2.1 Требования к программному обеспечению

Программа должна предоставлять пользователю графический интерфейс, позволяющий:

- загрузить исходный и целевой объект из *.obj* файла;
- выбрать цвет и оптические свойства поверхности загруженных объектов;
- просматривать каждый объект и результат морфинга посредством поворота и масштабирования;
- изменять масштабирование и поворот исходного и целевого объектов морфинга;
- выбирать стадию морфинга объектов.

Программа должна корректно реагировать на любые действия пользователя.

### 2.2 Используемые типы и структуры данных

1) Сцена состоит из:

- объекта;
- камеры;
- источника света.

2) Объект состоит из:

- массива точек;
- массива треугольников (индексы вершин);
- массива внешних нормалей к треугольникам;
- материала;
- матрицы преобразований.

3) Результат морфинга, состоит из:

- массива точек;
- массива функций, интерполирующих вершины в момент  $t$ ;
- массива треугольников (индексы вершин);
- массива внешних нормалей к треугольникам;
- массива функций, интерполирующих внешние нормали в момент



$t$ ;

- материала;
- функции интерполирующей материал в момент  $t$ ;
- матрицы преобразований.

4) камера состоит из:

- положения;
- точки, на которую направлен взгляд;
- угол обзора в радианах;
- отношение сторон;
- удаление ближней плоскости пирамиды видимости;
- удаление дальней плоскости пирамиды видимости.

5) источник света состоит из:

- положения;
- интенсивности;
- цвета; TODO: возможно убрать

6) материал состоит из:

- цвета в формате *RGB*;
- коэффициента диффузного отражения;
- коэффициента зеркального отражения;
- степени, аппроксимирующей пространственное распределение зеркально отраженного света.

Для построения суперсетки используется структура данных DCEL (doubly connected edge list) [3]. Визуальное представление структуры данных приведено на рисунке ??.

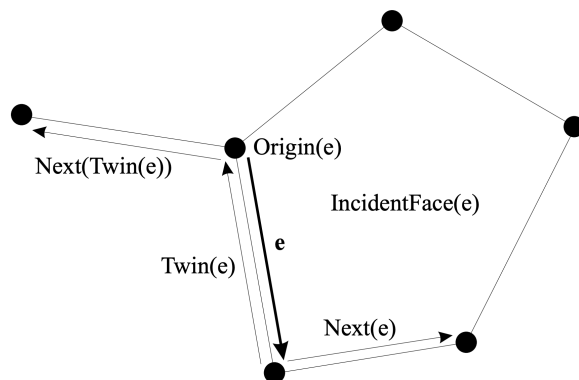


Рисунок 2.1 — DCEL (doubly connected edge list) [3]

DCEL состоит из:

- массив вершин;
- массив полуребер;
- массив граней;

Полуребро состоит из:

- индекса вершины-начала;
- индекса полуребра-двойника;
- индекса смежной грани;
- индекса следующего полуребра;

Грань состоит из индекса полуребра, связанного с ней.

## 2.3 Математические основы алгоритмов

### 2.3.1 Барицентрические координаты

Барицентрические координаты  $\alpha$ ,  $\beta$  и  $\gamma$  определяют положение точки  $P$  относительно вершин треугольника  $A$ ,  $B$  и  $C$  [5].

Точка  $P$  находится внутри или на границах треугольника  $ABC$ , если она может быть представлена в виде аффинной комбинации вершин [5]:

$$P = \alpha A + \beta B + \gamma C, \text{ где } \alpha + \beta + \gamma = 1, \text{ и } \alpha, \beta, \gamma \geq 0 \quad [5] \quad (2.1)$$

Если хотя бы одна из координат  $\alpha, \beta, \gamma$  отрицательна, то точка  $P$  лежит вне треугольника [5].

Барицентрическая координата  $\alpha$  точки  $P$  пропорциональна площади треугольника  $PBC$  (см. рисунок ??) (треугольника, противолежащего вершине  $A$ ) [5].

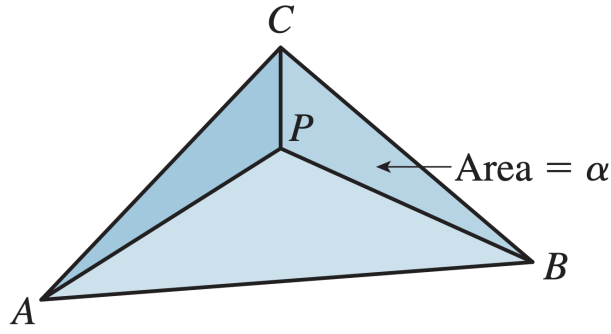


Рисунок 2.2 — Точка  $P$  разделяет треугольник  $ABC$  на три меньших треугольника, площади которых соотносятся как  $\alpha$ ,  $\beta$  и  $\gamma$ ; Барицентрические координаты точки  $P$  равны  $(\alpha, \beta, \gamma)$  [5]

Координата  $\alpha$  точки  $P$  определяется как отношение площади треугольника  $PBC$  к площади всего треугольника  $ABC$  [5]:

$$\alpha = \frac{S_{\triangle PBC}}{S_{\triangle ABC}} \quad (2.2)$$

Аналогичные соотношения используются для  $\beta$  (пропорционально  $S_{\triangle PAC}$ ) и  $\gamma$  (пропорционально  $S_{\triangle PAB}$ ).

На плоскости барицентрические координаты точки  $P$  в треугольнике  $ABC$  могут быть вычислены при помощи крестового произведения:

$$\begin{cases} \alpha = \frac{|BC \times BP|}{|AB \times AC|} = \frac{(C-B)_x \cdot (P-B)_y - (C-B)_y \cdot (P-B)_x}{(B-A)_x \cdot (C-A)_y - (B-A)_y \cdot (C-A)_x} \\ \beta = \frac{|CA \times CP|}{|AB \times AC|} = \frac{(A-C)_x \cdot (P-C)_y - (A-C)_y \cdot (P-C)_x}{(B-A)_x \cdot (C-A)_y - (B-A)_y \cdot (C-A)_x} \\ \gamma = 1 - \alpha - \beta \end{cases} \quad (2.3)$$

### 2.3.2 Поиск пересечения дуг на единичной сфере

Для нахождения пересечения дуг необходимо:

- 1) найти прямую по которой пересекаются плоскости, содержащие дуги;
- 2) найти точки пересечения прямой с единичной сферой;
- 3) проверить принадлежность точек обеим дугам.

Пусть дуга  $A$  ограничена точками  $A_0, A_1$ , а дуга  $B$  —  $B_0, B_1$ .

Направляющий вектор прямой можно вычислить как векторное произведение нормалей к плоскостям, содержащим дуги:

$$\mathbf{d} = \mathbf{n}_A \times \mathbf{n}_B = (\mathbf{A}_0 \times \mathbf{A}_1) \times (\mathbf{B}_0 \times \mathbf{B}_1) \quad (2.4)$$

Точки пересечения с единичной сферой равны:

$$P_{1,2} = \pm \frac{\mathbf{d}}{\|\mathbf{d}\|} \quad (2.5)$$

Т. к. длина радиус-векторов всех точек на единичной сфере равна 1, скалярное произведение равно косинусу угла между векторами. В таком случае точка  $P_{1,2}$  принадлежит обеим дугам  $A$  и  $B$ , а следовательно является их пересечением если:

$$\begin{cases} A_0 \cdot A_1 \leq A_0 \cdot P_{1,2} \\ A_0 \cdot A_1 \leq A_1 \cdot P_{1,2} \\ B_0 \cdot B_1 \leq B_0 \cdot P_{1,2} \\ B_0 \cdot B_1 \leq B_1 \cdot P_{1,2} \end{cases} \quad (2.6)$$

### 2.3.3 Поиск точки внутри объекта

Основные шаги:

- 1) Испустить луч из центра масс  $O$  любой грани в направлении внутренней нормали.
- 2) Найти все точки пересечения с другими гранями.
- 3) Определить ближайшую к началу луча точку пересечения  $I$ .
- 4) Вернуть точку по середине отрезка  $OI$ .

### Пересечение луча с полигоном

В параметрическом представлении луча задается уравнением (??).

$$P(t) = P_0 + \mathbf{d} \cdot t, \quad (2.7)$$

где  $P_0$  — начальная точка луча;  $\mathbf{d}$  — направляющий вектор луча.

Произвольная точка  $M$  лежит на плоскости  $A$ , если для нее выполняется равенство (??).

$$(M - M_A) \cdot \mathbf{n}_A = 0, \quad (2.8)$$

где  $M$  — произвольная точка

$M_A$  — известная точка на плоскости  $A$

$\mathbf{n}_A$  — вектор нормали к  $A$ .

Подставив (??) в (??), получим уравнение (??)

$$t = \frac{(M_A - P_0) \cdot \mathbf{n}_A}{\mathbf{d} \cdot \mathbf{n}_A} \quad (2.9)$$

Если знаменатель  $\mathbf{d} \cdot \mathbf{n}_A$  равен нулю, то луч параллелен плоскости полигона. Если  $t < 0$ , то пересечение находится позади начала луча, и такая точка отбрасывается, так как поиск ведется внутри объекта.

Найдя точку пересечения луча с плоскостью, необходимо определить, лежит ли она внутри полигона. Для этого можно использовать барицентрические координаты.

## 2.4 Описание алгоритмов

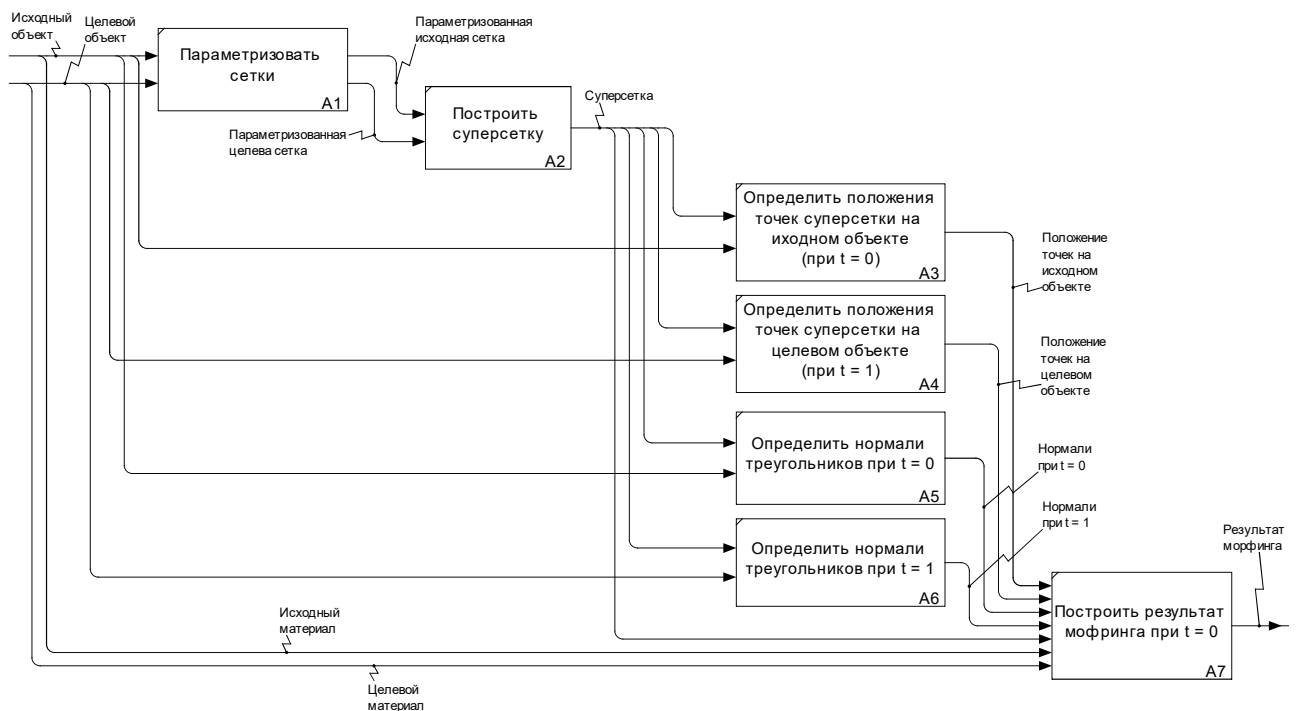


Рисунок 2.3 — Функциональная модель построения морфинга

Схемы основных алгоритмов, используемых в программном обеспечении представлены на рисунках ??–??.

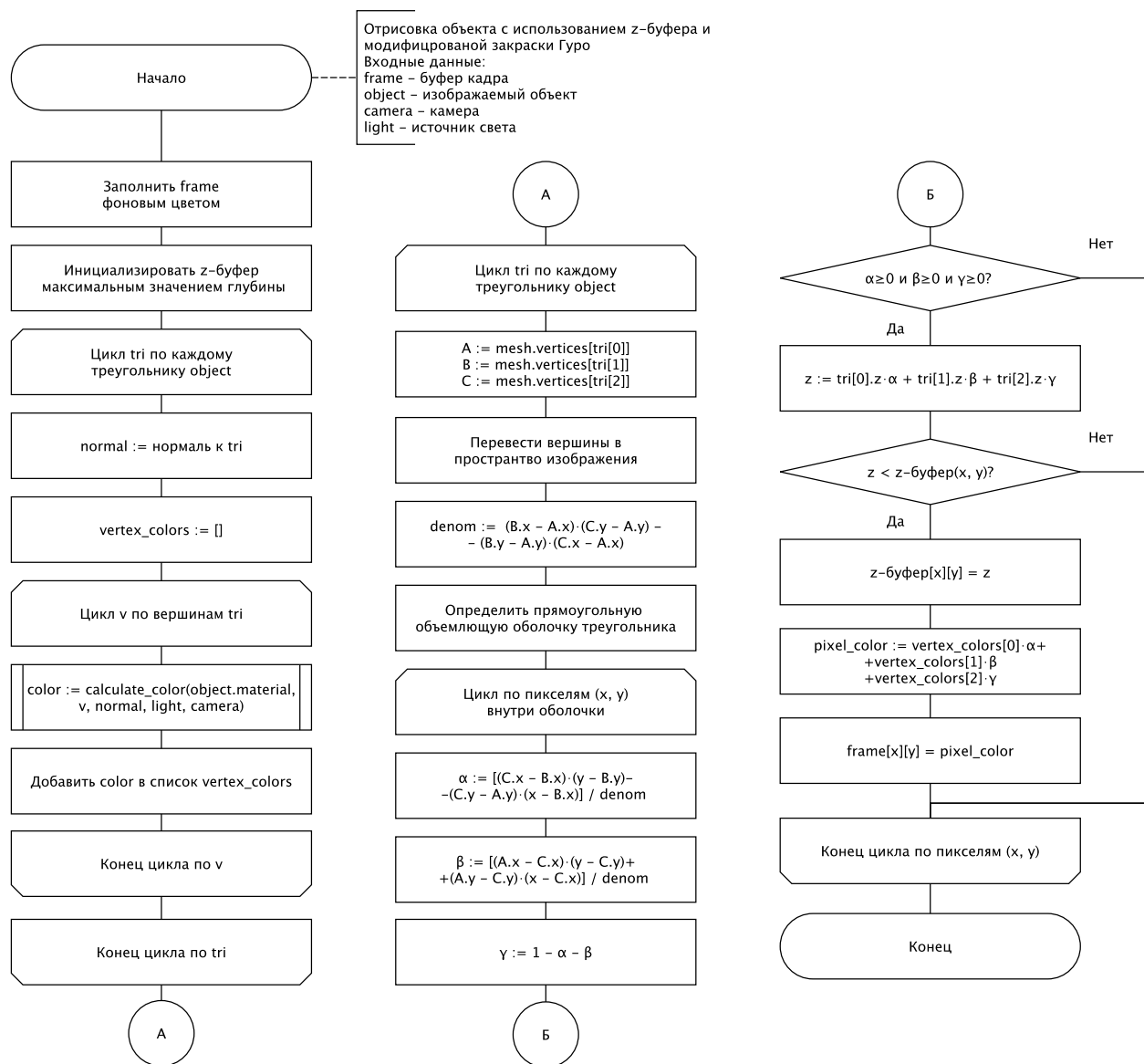


Рисунок 2.4 — Схема алгоритма отрисовки объекта с использованием z-буфера и модифицированной закрашки Гуро

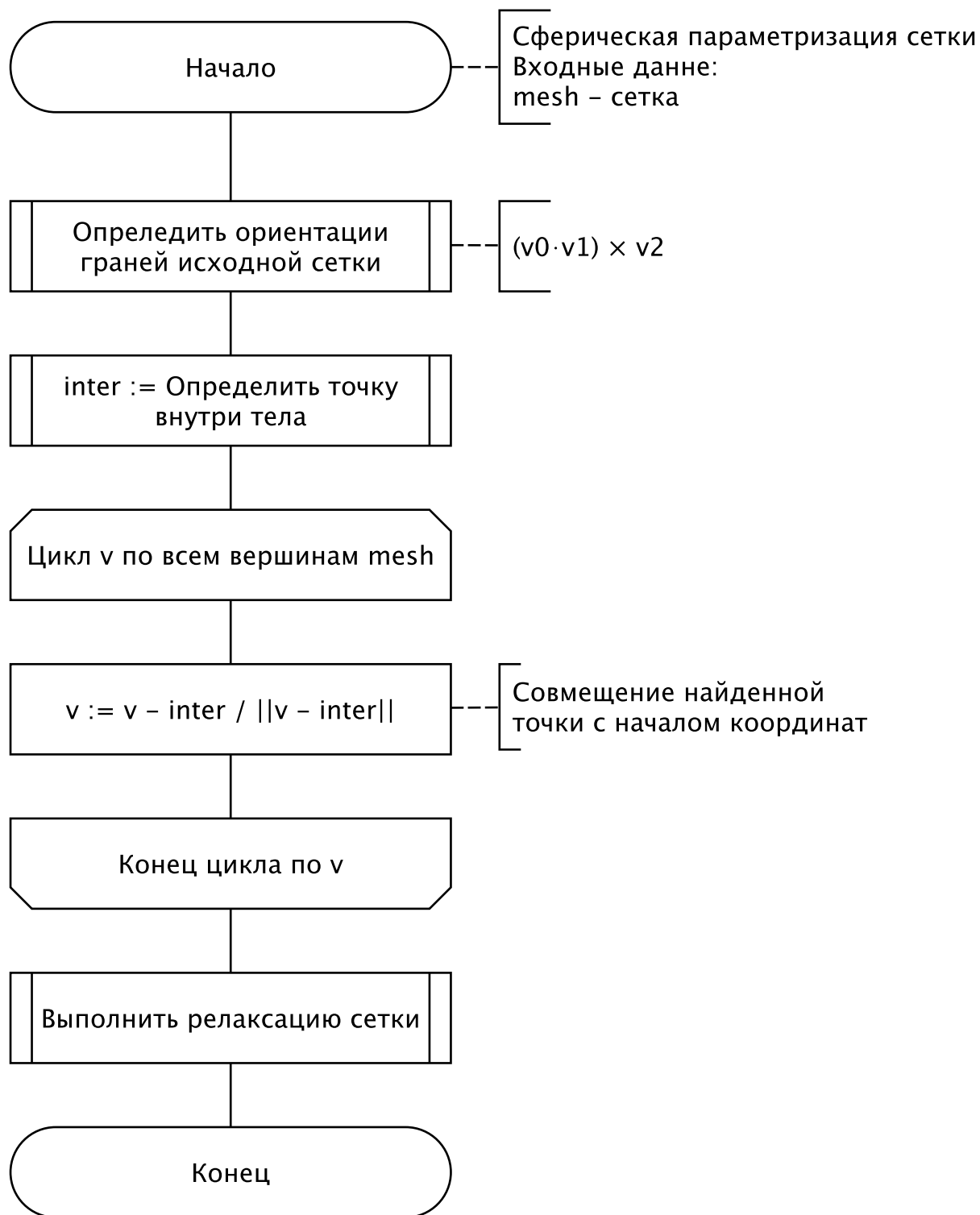


Рисунок 2.5 — Схема алгоритма сферической параметризации сетки

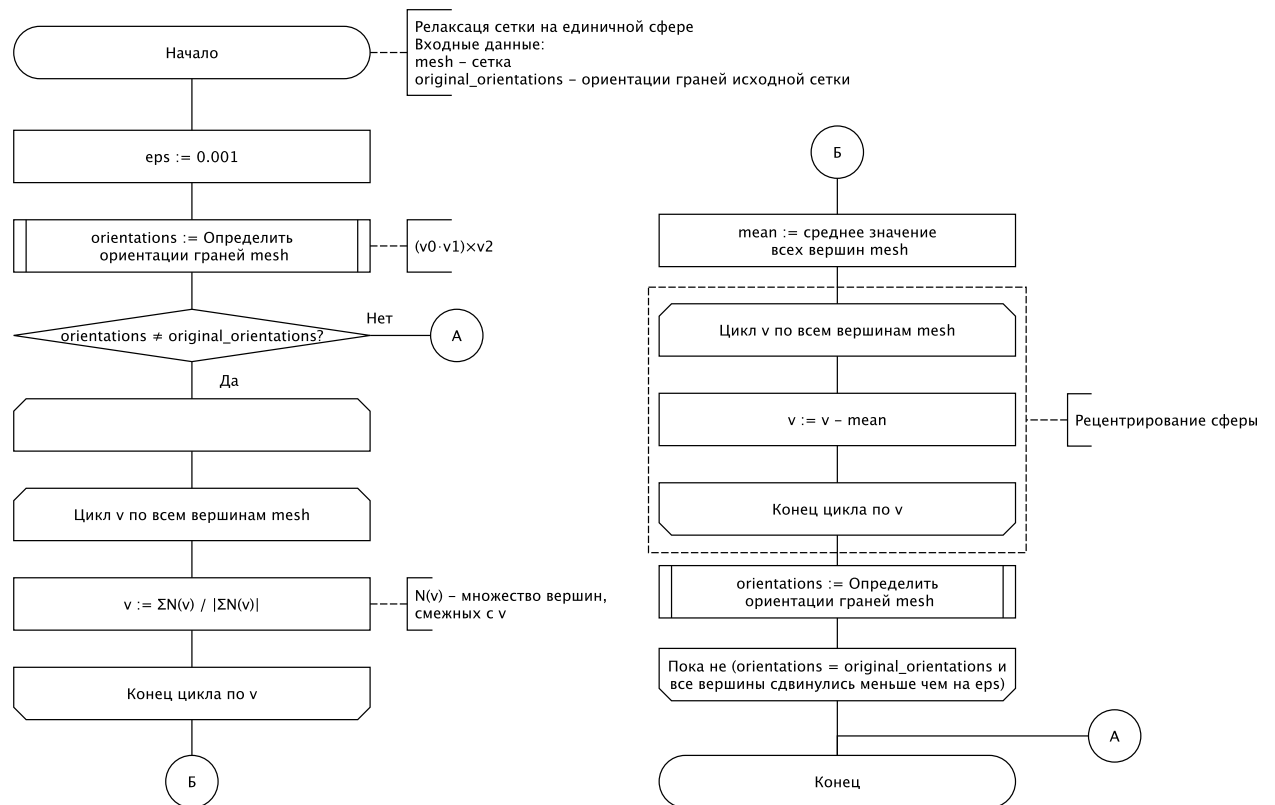


Рисунок 2.6 — Схема алгоритма релаксации сетки

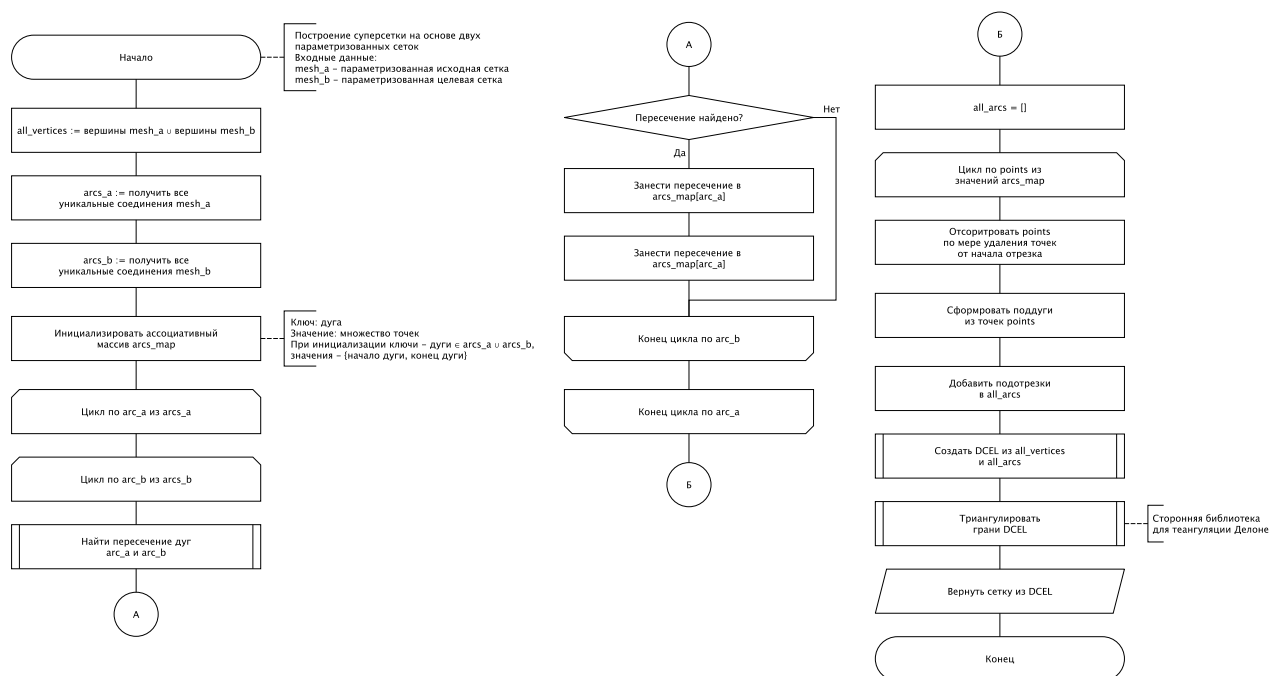


Рисунок 2.7 — Схема алгоритма построения суперсетки на единичной сфере



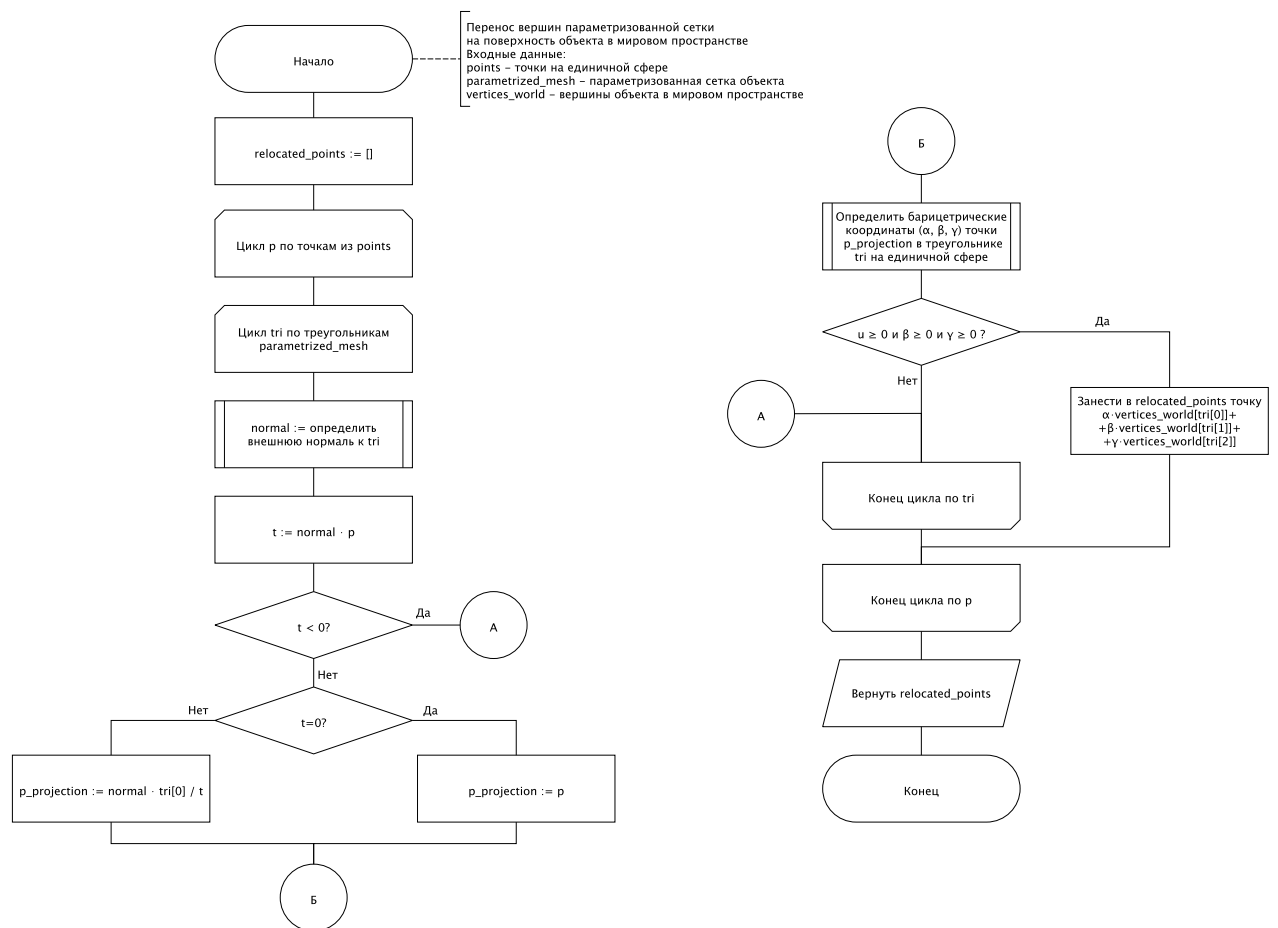


Рисунок 2.8 — Схема алгоритма переноса вершин суперсетки с единичной сферы в мировое пространство

## Вывод

### **3 Технологическая часть**

#### **Вывод**

## **4 Исследовательская часть**

### **4.1 Вывод**

# **ЗАКЛЮЧЕНИЕ**

## **СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. Mocanu Bogdan-Costel. 3D Mesh Morphing. Thèse de doctorat / Doctoral thesis: TELECOM SudParis; Universitatea Politehnica Bucuresti. France / Roumanie, 2013. Numéro national de thèse : 2013TELE0010; HAL Id: tel-00836048.
2. Порев В. Компьютерная графика. СПб.: БХВ-Петербург., 2002.
3. Alexa M. Mesh Morphing. 2001.
4. Роджерс Д. Алгоритмические основы машинной графики: Пер. с англ. — СПб: БХВ-Петербург, 1989. — С. 512.
5. Foley J. Computer Graphics: Principles and Practice. 3 edition. Addison Wesley, 2013.