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)

УТВЕРЖДАЮ
Заведующий кафедрой ИУ7
(Индекс)
_____ И. В. Рудаков
(И.О.Фамилия)
« _____ » _____ 2023 г.

З А Д А Н И Е
на выполнение курсовой работы

по дисциплине Компьютерная графика

Студент группы ИУ7-56Б

Жаворонкова Алина Андреевна
(Фамилия, имя, отчество)

Тема курсовой работы Визуализация модели цветка герберы

Направленность КР (учебная, исследовательская, практическая, производственная, др.)
учебная
Источник тематики (кафедра, предприятие, НИР) кафедра

График выполнения работы: 25% к _ нед., 50% к _ нед., 75% к _ нед., 100% к _ нед.

Задание Разработать программное обеспечение с пользовательским интерфейсом для визуализации модели цветка герберы. Программа должна обеспечивать возможность смены времени суток для визуализации движения цветка от влияния освещенности. Модель цветка должна состоять из следующих объектов: стебель, цветоложе, лепестки, лист. Программное обеспечение должно предоставлять возможность задания положения камеры и источника освещения. Программа должна предусматривать построение теней цветка. Сцена должна состоять из следующих объектов: камера, источник освещения, ограничивающая поверхность, модель цветка герберы.

Оформление курсовой работы:

Расчетно-пояснительная записка на 25-30 листах формата А4.
Расчетно-пояснительная записка должна содержать постановку введение, аналитическую часть, конструкторскую часть, технологическую часть, экспериментально-исследовательский раздел, заключение, список литературы, приложения.
Перечень графического материала (плакаты, схемы, чертежи и т.п.). На защиту проекта должна быть представлена презентация, состоящая из 15-20 слайдов. На слайдах должны быть отражены: постановка задачи, использованные методы и алгоритмы, расчетные соотношения, структура комплекса программ, диаграмма классов, интерфейс, характеристики разработанного ПО, результаты проведенных исследований.

Дата выдачи задания « ___ » _____ 2023 г.

Руководитель курсовой работы _____ А. В. Куров
(Подпись, дата) (И.О.Фамилия)

Студент _____ А. А. Жаворонкова
(Подпись, дата) (И.О.Фамилия)

Оглавление

Введение

В настоящее время существует множество алгоритмов для построения реалистичных изображений. Их разнообразие объясняется тем, что не существует универсального способа построения изображения, который одновременно быстродействующий и создающий высоко реалистичное изображение. Таким образом, различные алгоритмы предоставляют возможность сделать акцент на наиболее важной характеристике в конкретной задаче.

Целью данной работы является разработка программного обеспечения для создания реалистичного изображения цветка. В программе должна быть возможность изменения положения камеры и источника освещения.

Для достижения поставленной цели требуется решить следующие задачи:

- выделить объекты сцены и выбрать модель их представления;
- проанализировать алгоритмы визуализации трехмерной сцены, при необходимости рассмотреть модификации, обосновать выбор конкретного алгоритма;
- спроектировать архитектуру и графический интерфейс программы;
- реализовать выбранные ранее алгоритмы;
- исследовать производительность программы.

1 Аналитический раздел

1.1 Формализация объектов синтезируемой сцены

Сцена состоит из следующих объектов:

1. Ограничивающая плоскость – расположена параллельно плоскости OXZ;
2. Цветок – расположен на ограничивающей плоскости. В нем можно выделить следующие составляющие:

   2.1. Стеблевая часть:

   2.1.1. Цветоножка – длинный изогнутый цилиндр, описывающийся следующими уравнениями:

   $$\begin{cases} x = 0{,}6\cos t + \frac{1}{5}\sin z \\ y = 0{,}6\sin t \\ z = z \end{cases}, \text{ где } t \in [0; 2\pi), z \in [0; 20]$$

   2.1.2. Цветоложе – часть эллипсоида, задаваемая уравнением: $z = \frac{(x-\frac{1}{4})^2}{2} + \frac{y^2}{2}$ при $x \in [-2{,}25; 2{,}75], y \in [-2{,}5; 2{,}5]$

   2.1.3. Лист – пара кривых, образующих поверхность, описываются следующими уравнениями:

   $$\begin{cases} x = t \\ y = \frac{1}{3}t^2 \\ z = \frac{2}{t+1} + 3t \end{cases}, \quad \text{где} \quad t \in [0; 5{,}2]$$

   Вторая кривая будет являться поворотом первой кривой на угол $\varphi = \frac{-\pi}{3}$. Полученные уравнения, задающие вторую кривую:

   $$\begin{cases} x = t\cos\frac{-\pi}{3} - \frac{1}{3}t^2\sin\frac{-\pi}{3} \\ y = t\sin\frac{-\pi}{3} + \frac{1}{3}t^2\cos\frac{-\pi}{3}, \\ z = \frac{-2}{t-1} - 3t \end{cases} \quad \text{где} \quad t \in [-5{,}2; 0)$$

   Описанные выше кривые пересекаются в точках:

   $$A(0; 0; 2), B(3\sqrt{3}; 9; \frac{2}{3\sqrt{3}+1} + 9\sqrt{3})$$

2.2. Листовая часть – совокупность поверхностей, образующих непосредственно лепестки цветка. Один лепесток описывается уравнением:

$$z = \frac{1}{2}x^2 + y \quad \text{при} \quad x^2 + \frac{1}{5}y^2 \leq 3$$

Остальные лепестки будут получены путем поворота или отражения одного лепестка относительно центра листовой части.

3. Источник освещения – расположен в бесконечности. Начальное положение источника указывается по умолчанию, но пользователь может его изменить;
4. Камера – начальное положение источника указывается по умолчанию, но пользователь может его изменить.

1.2 Выбор способа определения моделей

Модели могут задаваться в следующих формах:

- **Каркасная модель.** В данной модели задается информация о вершинах и ребрах объекта. Это одна из простейших форм задания модели. Основная проблема отображения объектов с помощью каркасной модели заключается в том, что модель не всегда однозначно передает представление о форме объекта.
- **Поверхностная модель.** Данный тип модели часто используется в компьютерной графике. Поверхность может описываться аналитически, либо задаваться другим способом. Недостатком поверхностной модели является отсутствие информации о том, с какой стороны поверхности находится материал.
- **Твердотельная модель.** Данная форма задания модели отличается от поверхностной формы тем, что в объемных моделях к информации о поверхностях добавляется информация о том, с какой стороны расположен материал. Это можно сделать путем указания направления внутренней нормали.

Для решения поставленной задачи не подойдет каркасная форма модели, потому что из-за большого количества составных частей изображение будет трудно воспринимать как единое целое. Твердотельная модель является избыточной формой представления, поскольку пользователю не важно, из какого материала выполнен объект. Таким образом, достаточной является поверхностная форма модели.

1.3 Выбор способа задания поверхностных моделей

Способы задания поверхностной модели:

1. **Аналитический способ.** Этот способ задания модели характеризуется описанием модели объекта, которое доступно в неявной форме, то есть для получения визуальных характеристик необходимо дополнительно вычислять некоторую функцию, которая зависит от параметра.
2. **Полигональная сетка** – совокупность пересекающихся плоскостей. Данный способ характеризуется совокупностью вершин, ребер и граней, определяющих форму объекта в трехмерном пространстве.

Для решения поставленной задачи аналитический способ будет сложен в реализации, потому что большинство поверхностей, которые должны быть построены для создания реалистичного изображения, имеют произвольную форму, то есть к ним не применимы математические закономерности. Таким образом, в качестве способа задания поверхностных моделей был выбран способ полигональной сетки.

1.4 Выбор алгоритма удаления невидимых ребер и поверхностей

Задача удаления невидимых линий и поверхностей является одной из наиболее сложных в машинной графике. Алгоритмы удаления невидимых линий и поверхностей служат для определения линий ребер, поверхностей или объемов, которые видимы или невидимы для наблюдателя, находящегося в

заданной точке пространства. Решать поставленную задачу удаления можно как в объектном пространстве (в мировой системе координат), так и в пространстве изображения (в экранных координатах).

Алгоритм может работать в любом пространстве и должен быть достаточно быстрым при работе с множеством объектов сцены, чтобы пользователь не ожидал долгой загрузки изображения.

Рассмотрим алгоритмы для удаления невидимых ребер и поверхностей.

1.4.1 Алгоритм Робертса

Алгоритм работает в объектном пространстве. Требуется, чтобы все изображаемые тела были выпуклыми.

Алгоритм работает в 3 этапа: подготовка исходных данных, удаление ребер, экранируемых самим телом, удаление ребер, экранируемых другими телами.

Этап 1. Подготовка исходных данных.

Необходимо сформировать матрицы тел, которые будут представлять выпуклые твердые тела. Полученные матрицы будут иметь вид:

$$\mathrm{A} = \begin{bmatrix} a_1 & a_2 & \dots & a_n \\ b_1 & b_2 & \dots & b_n \\ c_1 & c_2 & \dots & c_n \\ d_1 & d_2 & \dots & d_n \end{bmatrix}$$

В такой матрице каждый столбец содержит коэффициенты одной плоскости. При этом точки, лежащие внутри тела, дают положительное скалярное произведение с каждым столбцом матрицы.

Этап 2. Удаление ребер, экранируемых самим телом.

На данном этапе используется вектор направления взгляда:

$$\mathrm{E} = [0 \quad 0 \quad -1 \quad 0]$$

При умножении вектора E на матрицу тела отрицательные компоненты полученного вектора будут соответствовать задним граням.

Если объект на сцене один, то работа алгоритма завершается на данном этапе.

Этап 3. Удаление ребер, экранируемых другими телами.

Необходимо провести луч из произвольной точки анализируемого отрезка в точку наблюдения. Если луч проходит через тело, то точка невидима, а луч расположен с положительной стороны от каждой грани тела.

Преимущества:

- Точность вычислений благодаря тому, что алгоритм работает в объектном пространстве;
- Использование математически простых и точных методов.

Недостатки:

- Возможность работать только с выпуклыми объектами;
- Сложность алгоритма $O(n^2)$, где n – количество объектов сцены.

1.4.2 Алгоритм, использующий z-буфер

Алгоритм работает в пространстве изображений. Основная идея: поиск по x и y наибольшего значения функции $z(x, y)$.

Используются два буфера:

- Буфер кадра, используемый для запоминания интенсивности каждого пикселя;
- Z-буфер – буфер глубины, используемый для запоминания координаты z (глубины каждого видимого пикселя).

В начале работы алгоритма буфер кадра заполнен фоновым значением интенсивности или цвета, а z-буфер – минимальным значением координаты z. Также удаляются нелицевые грани, если это целесообразно.

Затем каждый многоугольник преобразовывается в растровую форму в произвольном порядке. Для каждого пикселя в многоугольнике вычисляется его глубина и записывается в z-буфер, если она больше хранящегося значения.

Преимущества:

- Алгоритм делает тривиальной визуализацию пересечений сложных поверхностей;
- Сцены могут быть любой сложности;
- Сложность алгоритма $O(n)$, где n – количество объектов сцены;
- Экономия вычислительного времени, так как элементы сцены не сортируются.

Недостатки:

- Большой объем требуемой памяти;
- Трудоемкость устранения лестничного эффекта.

1.4.3 Алгоритм обратной трассировки лучей

Алгоритм работает в пространстве изображений.

Наблюдатель видит объект благодаря испускаемому неким источником свету, который падает на этот объект и каким-либо образом доходит до наблюдателя: отразившись от поверхности, преломившись или пройдя через нее. Так как немногие из лучей, выпущенных источником, доходят до наблюдателя, то целесообразно трассировать (отслеживать) лучи в обратном направлении – от наблюдателя к объекту.

Предполагается, что сцена уже преобразована в пространство изображения. Каждый луч, исходящий от наблюдателя, проходит через центр пикселя на растре до сцены. Траектория каждого луча отслеживается, чтобы определить, какие именно объекты сцены, пересекаются с данным лучом. Необходимо проверить пересечение каждого объекта сцены с каждым лучом. Если луч пересекает объект, то определяются все возможные точки пересечения луча и объекта. Можно получить большое количество пересечений, если рассматривать много объектов. Эти пересечения упорядочиваются по глубине. Пересечение с максимальным значением координаты z представляет видимую поверхность для данного пикселя. Атрибуты этого объекта используются для определения характеристик пикселя.

Если точка зрения находится не в бесконечности, предполагается, что наблюдатель по-прежнему находится на положительной полуоси z. Картинная плоскость, перпендикулярна оси z. Задача состоит в построении одноточечной центральной проекции на картинную плоскость.

Преимущества:

- Высокая реалистичность получаемого изображения;
- Простота модификации при работе с несколькими источниками освещения, реализации различных оптических явлений;
- Алгоритм не требует дополнительных вычислений для нахождения теней.

Недостатки:

- Большая трудоемкость вычислений, производительность.

1.4.4 Алгоритм Варнока

Алгоритм работает в пространстве изображений.

В пространстве изображения рассматривается окно и решается вопрос о том, пусто ли оно, или его содержимое достаточно просто для визуализации. Если это не так, то окно разбивается на фрагменты до тех пор, пока содержимое подокна не станет достаточно простым для визуализации или его размер не достигнет требуемого предела разрешения. В последнем случае информация, содержащаяся в окне, усредняется, и результат изображается с одинаковой интенсивностью или цветом.

Устранение лестничного эффекта можно реализовать, доведя процесс разбиения до размеров, меньших, чем разрешение экрана на один пиксель, и усредняя атрибуты подпикселей, чтобы определить атрибуты самих пикселей.

Преимущества:

- Эффективность для простых сцен;
- Простота устранения лестничного эффекта.

Недостатки:

- Неэффективность при большом количестве объектов.

1.4.5 Вывод

Сравнение описанных выше алгоритмов представлено таблицей 1.

Таблица 1. Сравнение алгоритмов

	Алгоритм Робертса	Алгоритм, использующий z-буфер	Алгоритм обратной трассировки лучей	Алгоритм Варнока
Сложность алгоритма (N – количество граней, C – количество пикселей)	$O(N^2)$	$O(CN)$	$O(CN)$	$O(CN)$
Эффективность для сцен с большим количеством объектов	Низкая	Высокая	Низкая	Средняя
Пространство работы алгоритма	Объектное пространство	Пространство изображений	Пространство изображений	Пространство изображений
Сложность реализации	Высокая	Низкая	Средняя	Средняя

Таким образом, оптимальным для решения данной задачи является алгоритм, использующий z-буфер, так как он обеспечивает достаточную реалистичность изображения и скорость работы.

1.5 Выбор алгоритма построения теней

Поскольку в качестве алгоритма удаления невидимых линий и поверхностей был выбран алгоритм, использующий z-буфер, для построения теней будет использована его модификация.

Строится сцена из точки наблюдения, совпадающей с источником. Значения z для этого вида хранятся в отдельном теневом z-буфере. Значения интенсивности не рассматриваются.

Затем сцена строится из точки, в которой находится наблюдатель. При обработке каждой поверхности или многоугольника его глубина в каждом пикселе сравнивается с глубиной в z-буфере наблюдателя. Если поверхность видима, то значения (x, y, z) из вида наблюдателя линейно преобразуются в значения (x', y', z') на виде из источника. Для того чтобы проверить, видимо ли значение z' из положения источника, оно сравнивается со значением теневого z-буфера при x', y'. Если оно видимо, то оно отображается в буфер кадра в точке x, y без изменений. Если нет, то точка находится в тени и изображается согласно соответствующему правилу расчета интенсивности с учетом затенения, а значение в z-буфере наблюдателя заменяется на z'.

1.6 Анализ методов закрашивания

1.6.1 Простая закраска

Вся грань закрашивается одним уровнем интенсивности. Используется минимальное количество вычислений, но снижается качество получаемого изображения.

Используется при выполнении трех условий:

1. Предполагается, что источник находится в бесконечности;
2. Предполагается, что наблюдатель находится в бесконечности;
3. Закрашиваемая грань является реально существующей, а не полученной в результате аппроксимации поверхности.

Недостатком является возникновение ребер. При закраске каждой грани со своей интенсивностью граница между ними становится видна, и возникают ребра.

1.6.2 Закраска по Гуро

Закраска по Гуро выполняет сглаживание на основе биполярной интерполяции интенсивности.

Вводится понятие нормали к вершине, на основе которой вычисляется интенсивность каждой вершины и выполняется первая интерполяция вдоль ребер. Вторая интерполяция выполняется при вычислении интенсивности пикселей, расположенных на сканирующей строке. Качество изображения улучшится. Граница между двумя гранями визуально сгладится.

Закраска по Гуро не предусматривает учет кривизны поверхности. При применении закраски по Гуро возможно получение плоского изображения, когда углы, образованные гранями, одинаковые.

Закраска по Гуро хорошо сочетается с диффузной составляющей поверхности (матовой).

1.6.3 Закраска по Фонгу

Основная идея закраски по Фонгу: интерполировать нормали, а не интенсивности, как в закраске по Гуро.

От точки к точке в пределах грани нормали изменяются, учитывается криволинейный характер поверхности. Изображение получается более качественное, но трудоёмкость закраски по Фонгу будет выше.

Закраска по Фонгу хорошо сочетается с зеркальной составляющей: моделирует блики, возникающие при зеркальном отражении.

1.6.4 Вывод

Таблица 2. Сравнение алгоритмов закраски

	Простая закраска	Закраска по Гуро	Закраска по Фонгу

Реалистичность получаемого изображения	Низкая	Средняя	Высокая
Эффективность для сцен с большим количеством объектов	Высокая	Средняя	Низкая
Сочетаемость с диффузной составляющей поверхности	Нет	Да	Нет

Для данной задачи оптимальным алгоритмом закраски является закраска по Гуро, так как она обеспечивает достаточную реалистичность и скорость работы при построении криволинейных поверхностей.

1.7 Вывод

В данном разделе были формализованы объекты синтезируемой сцены, проведен обзор предметной области: рассмотрены существующие методы удаления невидимых линий и поверхностей, методы закрашивания и методы удаления теней. Из рассмотренных методов были выбраны оптимальные для поставленной задачи.

В качестве алгоритма удаления невидимых линий и поверхностей был выбран алгоритм, использующий Z-буфер. В качестве алгоритма закрашивания была выбрана закраска по Гуро. В качестве алгоритма построения теней была выбрана модификация алгоритма, использующего Z-буфер.

2 Конструкторская часть

2.1 Общий алгоритм решения поставленной задачи

Структура программы представлена на рисунке 1 в виде IDF0-диаграммы.

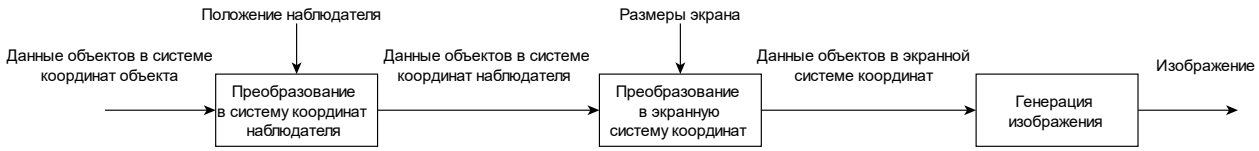

Рисунок 1. Структура программы

2.2 Алгоритм, использующий z-буфер

1. Всем элементам буфера кадра присвоить фоновое значение;
2. Инициализировать z-буфер минимальным значением глубины;
3. Для каждого многоугольника сцены в произвольном порядке:
   3.1. Для каждого пикселя, который принадлежит многоугольнику вычислить его глубину $z(x, y)$;
   3.2. Сравнить глубину $z(x, y)$ со значением $Z_{\text{буфер}}(x, y)$, хранящимся в z-буфере в этой же позиции. Если $z(x, y) > Z_{\text{буфер}}(x, y)$, то:
      3.2.1. Записать атрибут этого многоугольника в буфер кадра;
      3.2.2. Заменить $Z_{\text{буфер}}(x, y)$ на $z(x, y)$
4. Вывести итоговое изображение

На рисунке 2 изображена схема алгоритма z-буфера.

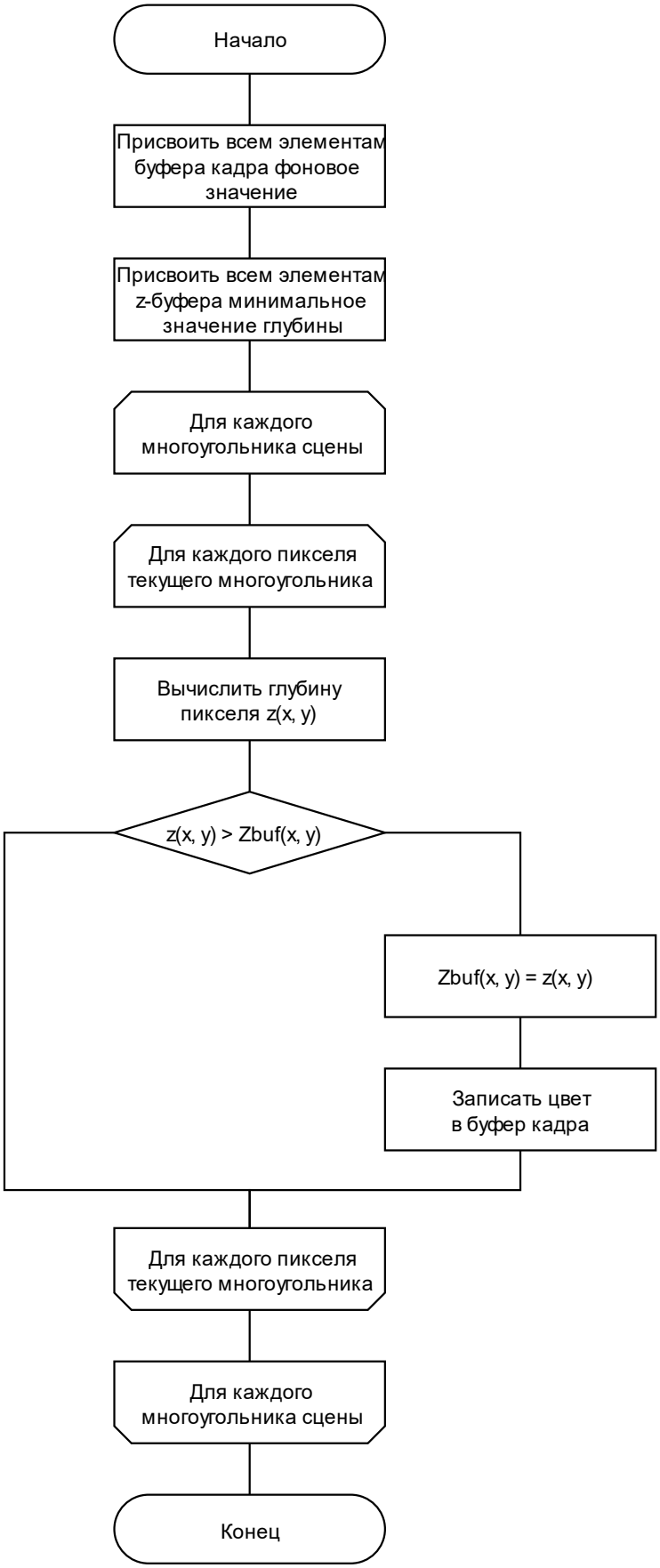

Рисунок 2. Схема алгоритма z-буфера

2.3 Модифицированный алгоритм, использующий z-буфер

1. Для источника света:
   1.1. Инициализировать теневой z-буфер минимальным значением глубины;
   1.2. Определить теневой z-буфер для источника.
2. Выполнить алгоритм z-буфера для точки наблюдения. При этом, если некоторая поверхность оказалась видимой относительно текущей точки наблюдения, то проверить, видима ли данная точка со стороны источников света.

   Для источника света:
   2.1. Координаты рассматриваемой точки (x, y, z) линейно преобразовать из вида наблюдателя в координаты (x', y', z') на виде из рассматриваемого источника света;
   2.2. Сравнить значение $Z_{shadowBuf}(x', y')$ со значением $z'(x', y')$. Если $z'(x', y') < Z_{shadowBuf}(x', y')$, то пиксель высвечивается с учетом его затемнения, иначе высвечивается без затемнения.

На рисунках 3-4 представлен модифицированный алгоритм z-буфера для построения теней.

Рисунок 3. Схема модифицированного алгоритма z-буфера для отображения теней (часть 1)

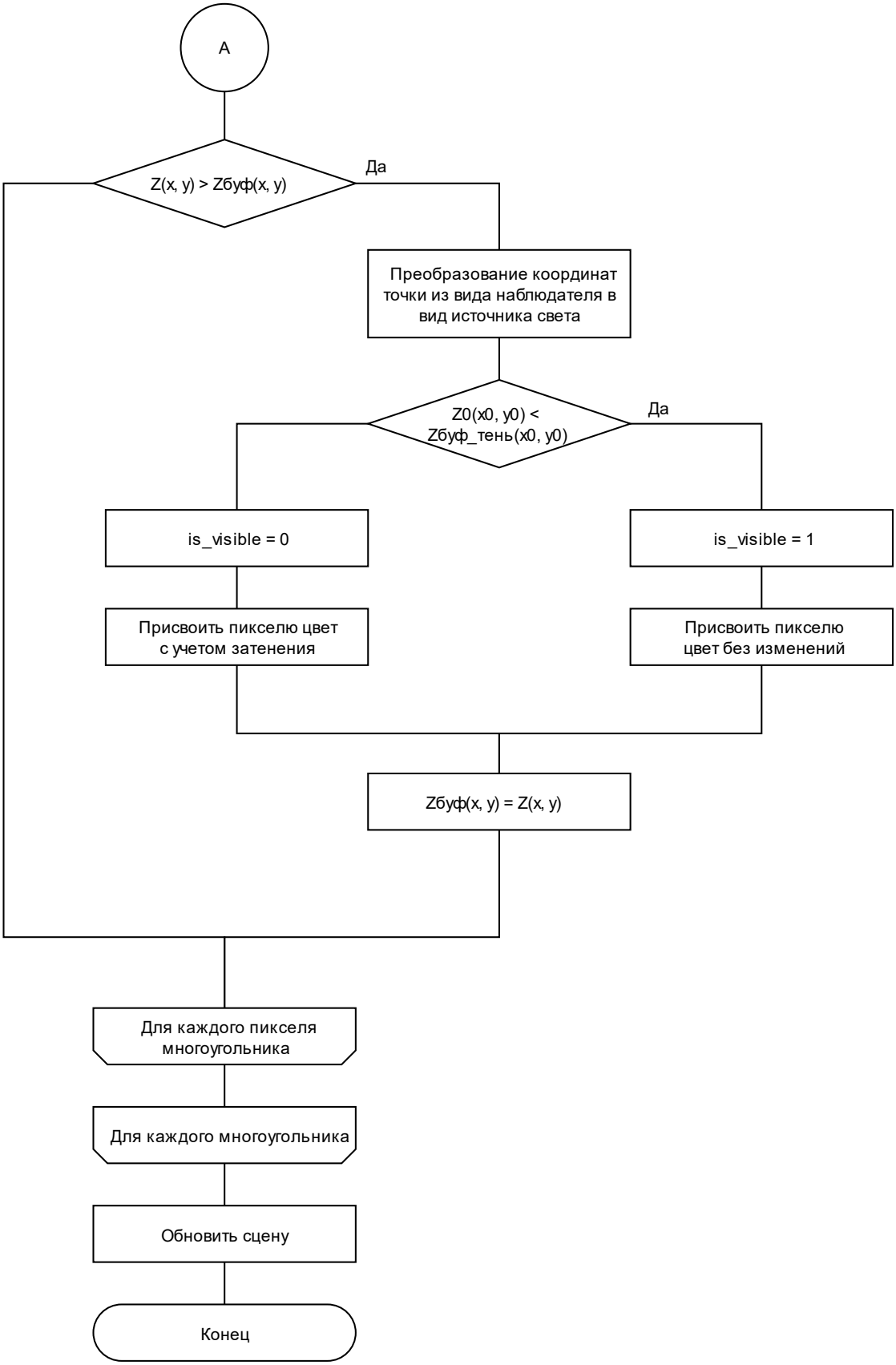

Рисунок 4. Схема модифицированного алгоритма z-буфера для отображения теней (часть 2)

2.4 Алгоритм закраски Гуро

В алгоритме закраски Гуро сначала определяется интенсивность вершин, а затем вычисляется интенсивность соответствующего пикселя. Схема алгоритма представлена на рисунке 5.

Этот метод хорошо сочетается с построчным сканированием. Для каждой сканирующей строки определяются её точки пересечения с рёбрами. В этих точках интенсивность вычисляется с помощью линейной интерполяции интенсивностей в вершинах рёбра. Затем для всех пикселей, находящихся внутри многоугольника и лежащих на сканирующей строке, аналогично вычисляется интенсивность.

Рисунок 5. Алгоритм закраски по Гуро

2.5 Схема алгоритма генерации одного кадра изображения

На рисунках 6-7 представлена схема алгоритма составления одного кадра изображения, объединяющего в себе модифицированный алгоритм Z-буфера и алгоритм закраски по Гуро.

Рисунок 6. Схема алгоритма генерации одного кадра изображения (часть 1)

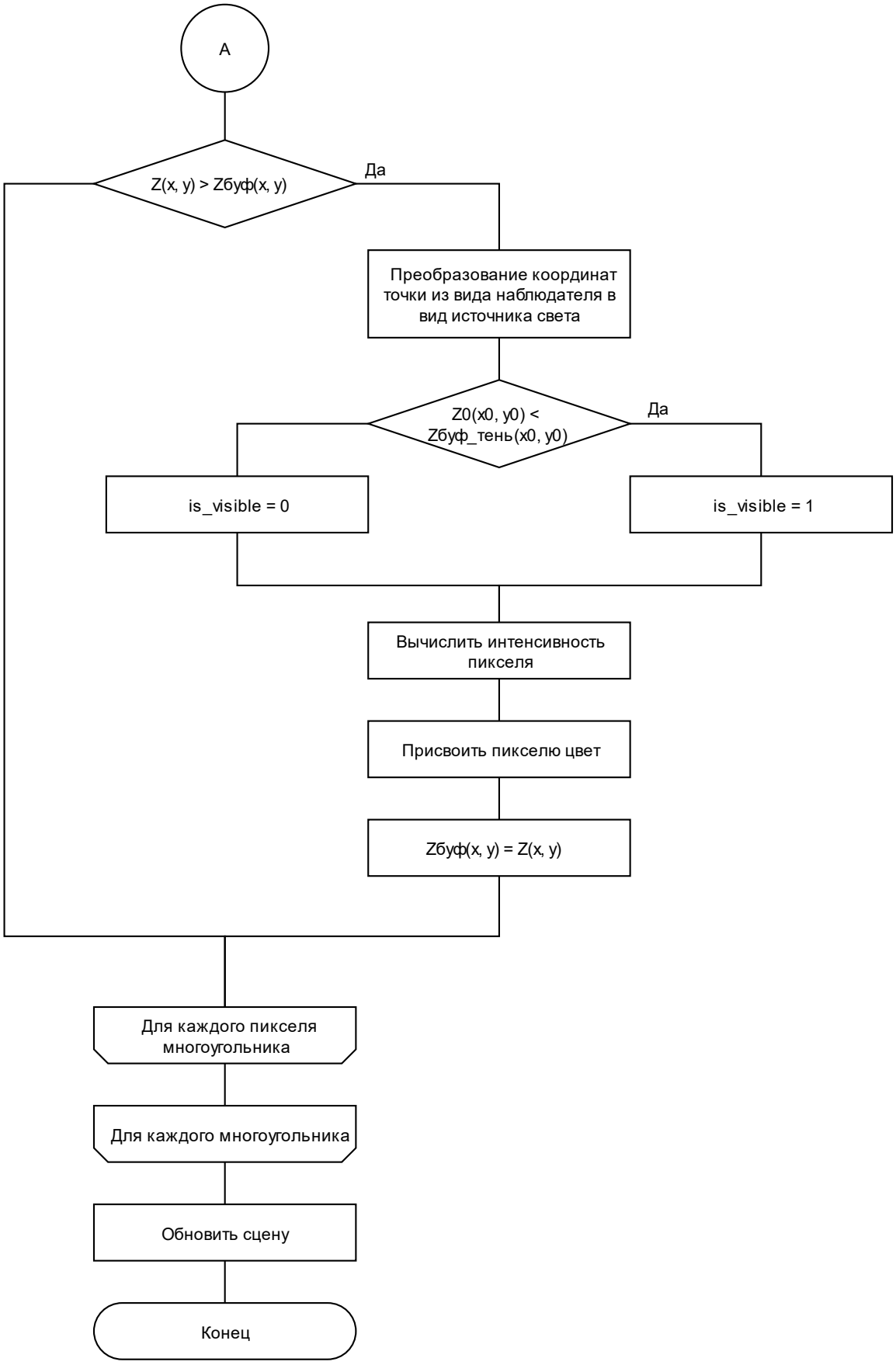

Рисунок 7. Схема алгоритма генерации одного кадра изображения (часть 2)

2.6 Используемые типы и структуры данных для представления объектов

В таблице 2 представлены объекты, способ их представления в программе и типы и структуры данных для их описания.

Таблица 3. Используемые типы и структуры данных

Данные	Представление	Тип
Точка в пространстве	Координаты точки по трем осям	Список типа *double*
Грань	Номера задействованных вершин	Список типа *int*
Полигональная модель	Набор задействованных вершин и граней	Списки типа *Vertex* (вершина) и *Facet* (грань)
Источник света	Углы по двум осям	Тип *int*

2.7 Вывод

В данном разделе был представлен общий алгоритм решения поставленной задачи в виде диаграммы IDF0 0 уровня, схемы алгоритмов использующего z-буфер, его модификации для построения теней и закраски Гуро. Также были описаны используемые типы и структуры данных для представления объектов.

3 Технологическая часть

3.1 Выбор языка программирования и среды разработки

В качестве языка для разработки программы был выбран язык программирования C++. Данный выбор основан на следующих аспектах:

- C++ – объектно-ориентированный язык, а именно такая методология программирования была выбрана для разработки программы;
- C++ обладает высокой вычислительной производительностью, что очень важно для выполнения поставленной задачи;
- Статическая типизация позволит устранять ошибки на стадии компиляции;
- Доступность учебной литературы.

В качестве среды разработки был выбран QtCreator. Данный выбор обусловлен следующими факторами:

- Данная среда разработки предоставляет удобную графическую библиотеку;
- Позволяет работать с графическим интерфейсом;
- Является бесплатной.

3.2 Описание используемых классов

На рисунке 8 представлена схема взаимодействия объектов сцены и показаны их составляющие.

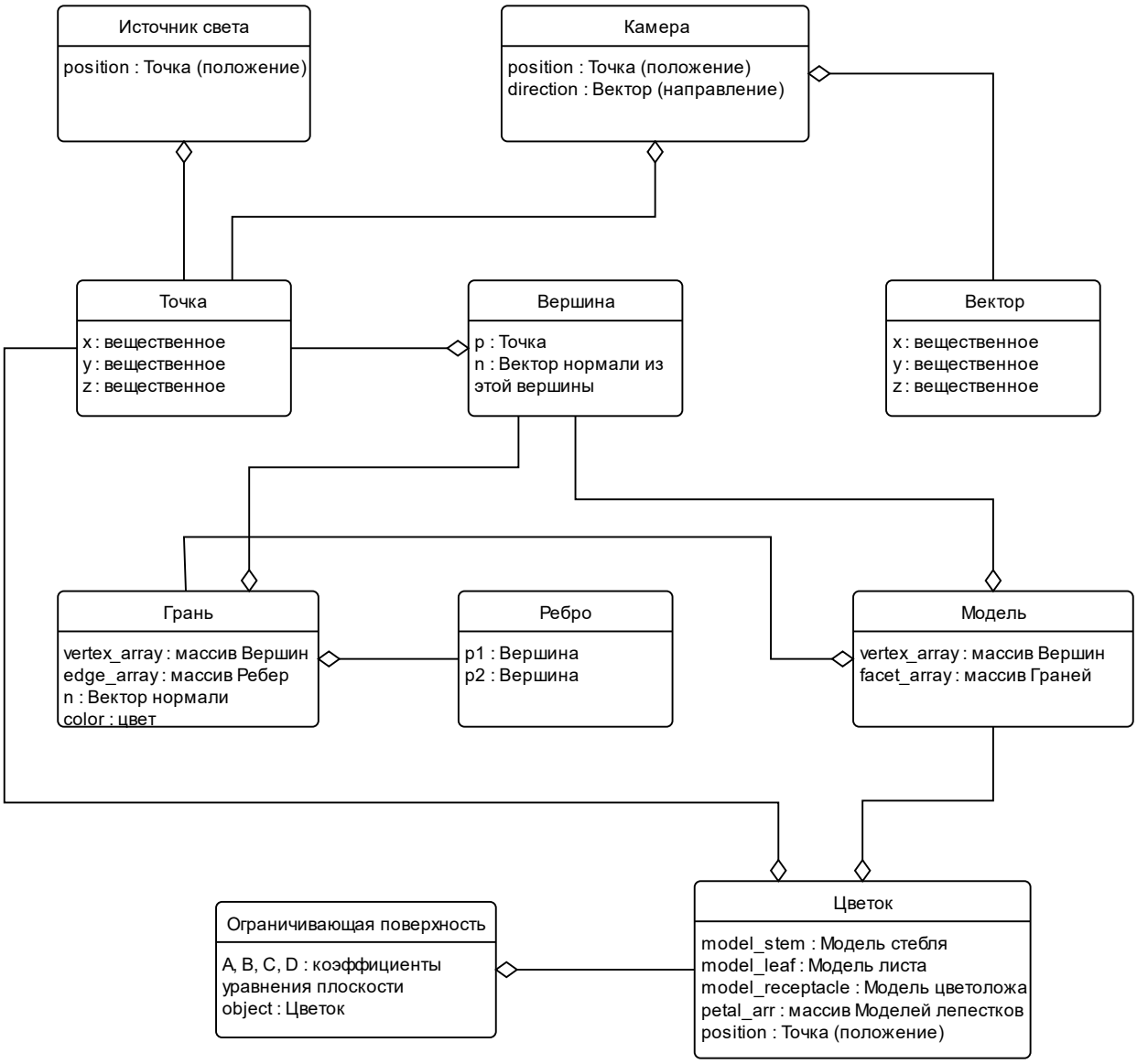

Рисунок 8. Объекты сцены

Список использованных источников

1. Роджерс Д. Алгоритмические основы машинной графики: Пер. с агл. – М.: Мир, 1989. – 512 с.
2. Методы представления дискретных трехмерных данных [Электронный ресурс]. – Режим доступа: https://www.graphicon.ru/oldgr/ru/library/multires_rep/index.html
3. Алгоритмы закраски [Электронный ресурс]. – Режим доступа: https://studbooks.net/2248060/informatika/odnotonnaya_zakraska_metod_graneniya
4. Обзор алгоритмов построения теней в реальном времени [Электронный ресурс]. – Режим доступа: https://www.ixbt.com/video/realtimeshadows.shtml
5. Ошаровская Е.В., Солодка В.И. Синтез трехмерных объектов с помощью полигональных сеток, Цифровые технологии, 2012, № 12, 2-9
6. Простые модели освещения [Электронный ресурс]. – Режим доступа: http://grafika.me/node/344
7. Куров А.В., Курс лекций по дисциплине «Компьютерная графика» [Текст]
8. Польский С.В., Компьютерная графика: учебн.-методич. Пособие. – М. : ГОУ ВПО МГУЛ, 2008. – 38 с.