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 ИУ «Информатика и системы управления»
КАФЕДРА ИУ-7 «Программное обеспечение ЭВМ и информационные технологии»

РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

К КУРСОВОЙ РАБОТЕ

НА ТЕМУ:

«Конструирование и визуализация загородного посёлка»

Студент ИУ7-53Б (группа) _____ (подпись, дата) Звягин Д.О. (И.О. Фамилия)

Руководитель курсовой работы _____ (подпись, дата) Вишневская Т.И. (И.О. Фамилия)

2024 г.

РЕФЕРАТ

СОДЕРЖАНИЕ

ОПРЕДЕЛЕНИЯ

ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ

В этой расчётно-пояснительной записке применяются следующие сокращения и обозначения.

ПО – Программное обеспечение.

Алгоритм QWFC (WFC) – Алгоритм квантового коллапса волновой функции.

ВВЕДЕНИЕ

Компьютерная графика – это область информатики, занимающаяся созданием, обработкой и отображением изображений с использованием вычислительных технологий. Она включает в себя как двумерную, так и трёхмерную графику, а также анимацию и визуализацию данных. Актуальность компьютерной графики возрастает с развитием технологий, таких как виртуальная и дополненная реальность, которые требуют высококачественной визуализации для создания реалистичных и интерактивных пользовательских опытов. В современных приложениях, от видеоигр до медицинской визуализации, компьютерная графика играет ключевую роль в представлении информации и взаимодействии с пользователями, что делает её важной областью для исследования и развития.

Целью данного курсового проекта является разработка ПО с пользовательским интерфейсом для генерации и визуализации загородного посёлка. Сцена содержит модели домов, источник света и камеру. Интерфейс пользователя должен позволять задавать параметры для генерации посёлка: размер домов, количество домов, ширина дорог, шаблоны расположения (кварталами или случайно).

Интерфейс приложения также должен предоставлять возможность для передвижения камеры и источника света.

Для достижения поставленной цели нужно решить следующие задачи:

— сравнение существующих алгоритмов процедурной генерации сцены;

— сравнение существующих алгоритмов компьютерной графики, использующихся для визуализации трёхмерной модели (сцены);

— выбор подходящих алгоритмов для решения поставленных задач;

— проектирование архитектуры и графического интерфейса ПО;

— выбор средств реализации ПО;

— разработка ПО;

— замер временных характеристик разработанного ПО.

1 Аналитическая часть

1.1 Формализация синтезируемой сцены

Сцена состоит из следующего набора объектов:

1) камера;
2) источник света;
3) модель загородного посёлка, состоящая из:
    - домов;
    - дорог;
    - деревьев.

Камера

Камера не является видимым объектом сцены. Она характеризуется только своим положением в пространстве и направлением просмотра. Изображение с камеры отображается в пользовательском интерфейсе ПО;

Источник света

Источник света отображает солнце, поэтому является всенаправленным и всегда находится на сравнительно большом расстоянии от других объектов сцены.

В пользовательском интерфейсе должна быть возможность задать положение источника света путём задания двух углов, которые задают направление, в котором будет находиться источник света относительно центра сцены.

Также в пользовательском интерфейсе должна быть возможность изменить расстояние, на котором находится источник света относительно центра сцены.

Модель загородного посёлка

Модель загородного посёлка является основной частью сцены. Сама модель должна генерироваться в соответствии с параметрами, заданными пользователем в пользовательском интерфейсе.

Далее формализуются объекты, входящие в сцену:

Ландшафт

В сцене не подразумевается использование сложного ландшафта, поэтому в качестве ландшафта используется плоское поле зелёного цвета (поле)

Дома

Дома, входящие в сцену состоят из геометрических примитивов, таких как:

— кубы;

— параллелепипеды;

— призмы.

Дома могут иметь как простой (прямоугольный), так и сложный (набор прямоугольников) фундамент. В качестве крыши всегда используются призмы

Дороги

Дороги должны проходить между домами и соединять их в улицы.

С точки зрения модели, дорогами являются прямоугольники или параллелепипеды, лежащие на плоскости ландшафта

Деревья

Деревья также состоят из геометрических примитивов:

— конусы;

— сферы;

— многоугольные пирамиды.

Все деревья будут представляться елями, имеющими цилиндр коричневого цвета в качестве ствола и тёмно-зелёные многоугольные пирамиды в качестве хвои.

Расположение объектов на модели

Основная плоскость является двумерной сеткой.

Пусть ячейка – это такой блок двумерной сетки который может:

— пустовать — в пределах ячейки нет объектов модели помимо основной плоскости;

— быть занят — в пределах ячейки содержится один или несколько объектов сцены.

В одной ячейке не может быть одновременно больше одного типа объектов (дом, дорога, дерево), но может быть несколько объектов одного типа.

Объекты не обязательно должны занимать всю площадь ячейки.

Все дома должны быть соединены дорогами, то есть от любого дома можно добраться по дорогам до любого другого дома.

1.2 Способы процедурной генерации сцены

Алгоритм квантового коллапса волновой функции

Алгоритм квантового коллапса волновой функции (далее - QWFC или WFC) — это метод генерации контента, который используется для создания двумерных и трёхмерных структур, таких как уровни в видеоиграх, текстуры и другие элементы. Он был разработан Максимом Гуминым и основан на концепциях из квантовой механики, хотя и не имеет прямого отношения к физике. [2]

Алгоритм работает путём заполнения сетки, каждая ячейка которой может принять одно из определённых состояний. Для заполнения сетки требуется предопределённый набор правил и приоритетов, который описывает то, какие ячейки могут быть "соседями"других ячеек. Изначально все ячейки находятся в состоянии суперпозиции – каждая может принять любое состояние (отсюда происходит слово "квантовый"в названии алгоритма). Затем, у случайной ячейки фиксируется одно из состояний (которое также выбирается случайно, но есть возможность задать приоритеты для тех или иных состояний). На основе этой зафиксированной ячейки, обновляется состояние остальной сетки, чтобы учесть появившиеся ограничения. Цикл повторяется, пока сетка не будет заполнена.

Этот алгоритм подходит для выполнения поставленной задачи (генерации загородного посёлка), так как он позволяет нам заполнить заданную плоскость в соответствии с ограничениями, а также предоставляет возможность влиять на результат посредством коэффициентов, которые можно задавать в пользовательском интерфейсе.

1.3 Способы описания трёхмерных геометрических моделей на сцене

Существует три основных типа трёхмерных моделей [1]:

1) *каркасная модель* – описывает вершины и рёбра между ними;
2) *поверхностная модель* – описывает вершины, рёбра и поверхности;
3) *твердотельная модель* – описывает ещё и с какой стороны расположен материал;

Для решения поставленной задачи лучше всего подходит поверхностная модель. Каркасная модель не позволяет задать свойства плоскостей (такие, как цвет, параметры отражения и блеска), требуемые для правильного восприятия сцены. Твердотельная модель содержит данные, нужды в которых для достижения цели — нет.

1.4 Сравнение алгоритмов удаления невидимых поверхностей

Задача удаления невидимых линий и поверхностей является одной из наиболее сложных в машинной графике. Эти алгоритмы служат для определения линий, рёбер и поверхностей, которые видимы или невидимы для наблюдателя, находящегося в данной точке пространства. [3]

В этом разделе рассмотрим несколько алгоритмов удаления невидимых поверхностей.

Алгоритм Варнока

Алгоритм Варнока (или алгоритм художника) основывается на разбиении пространства на области, которые могут быть классифицированы, как сложные и простые.

Под сложной областью подразумевается такая область (часть пространства на экране), которая превышает по размеру один пиксель и в которой возникает любой из следующих случаев:

— в область попали несколько объектов;

— в область попал один объект, который занимает не всю область.

Простой областью, в свою очередь, называются такие области, которые либо не превышают по размеру один пиксель, либо не являются сложными.

Алгоритм рекурсивно делит экранное пространство на области, пока не достигает простых областей, которые могут быть закрашены тем или иным цветом.

Трассировка лучей

Трассировка лучей — это более сложный и точный метод, который использует физический принцип распространения света.

В этом методе лучи "выстреливаются"из камеры в сцену, и для каждого луча вычисляется, какие объекты он пересекает. При пересечении луча с объектом, луч может не только остановиться, но и отразиться или преломиться.

Трассировка лучей позволяет добиться высокой степени реализма, однако она требует значительных вычислительных ресурсов.

Алгоритм с Z-буфером

Алгоритм с Z-буфером (или глубинным буфером) — это один из простейших [3] методов удаления невидимых поверхностей.

Он использует дополнительный буфер для хранения информации о глубине каждого пикселя на экране. При отрисовке каждого объекта в сцене сравнивается его глубина с уже сохраненной в z-буфере. Если новый объект ближе к камере, его цвет записывается в цветовой буфер, а его глубина — в z-буфер.

Этот метод позволяет эффективно обрабатывать сложные сцены и обеспечивает высокую производительность, что позволяет использовать его и при отрисовке сцен в реальном времени.

Выбор подходящего алгоритма

Было принято решение использовать алгоритм с Z-буфером.

Алгоритм Варнока хоть и предлагает относительную простоту, он будет не совсем оптимален для отрисовки большого количества примитивов, из которых будет состоять модель загородного посёлка.

Алгоритм трассировки лучей, в свою очередь, позволяет достичь относительного реализма по сравнению с другими алгоритмами, но он делает это затрачивая большое количество вычислительных ресурсов. Поскольку целью работы не является реалистичная визуализация модели, затраты на алгоритм трассировки лучей не будут оправданными.

Алгоритм с Z-буфером, в свою очередь, предлагает и относительную скорость работы и достаточное для цели работы качество изображения.

1.5 Сравнение алгоритмов построения теней

В данной работе будет использоваться метод теневых карт для построения теней на изображении.

Метод теневых карт

Метод теневых карт основывается на построении карты теней методом заполнения Z-буфера с точки зрения источника света и сравнения этого буфера с точки зрения камеры для правильного затенения пикселей [4].

Процесс применения этого метода можно разбить на следующие этапы:

1) создание теневых карт;
2) сравнение глубин пикселей;
3) применение освещения.

Создание теневых карт

На этом этапе информация о сцене заносится в Z-буфер с точки зрения источника света. Вместо цвета и яркости пикселей в Z-буфер будет сохраняться значение глубины каждого фрагмента изображения. В результате такого заполнения, будет получена текстура, называемая теневой картой. Эта текстура будет использоваться для определения, находится ли фрагмент изображения в тени.

Сравнение глубин пикселей

На втором этапе информация о сцене заносится в Z-буфер с точки зрения камеры. Для каждого фрагмента изображения координаты преобразуются в координаты теневой карты и значение глубины фрагмента сравниваются с соответствующим значением в теневой карте. Если значение глубины фрагмента больше, чем значение в теневой карте, фрагмент находится в тени.

Применение освещения

На последнем этапе полученная информация о тенях применяется к изображению: те пиксели (фрагменты) которые не находятся в тени, становятся светлее, а остальные - затеняются, в зависимости от интенсивности света.

Преимущества и недостатки

Метод теневых карт в сочетании с алгоритмом Z-буфера имеет свои преимущества и недостатки. К преимуществам можно отнести:

— высокую производительность для динамических сцен;

— возможность создания реалистичных теней для сложных объектов.

Однако существуют и недостатки:

— при низком разрешении теневых карт, могут наблюдаться проблемы с качеством изображения;

— ограниченная точность теней для объектов: чем дальше расположен объект от источника света, тем менее точной будет его тень.

Вывод

Метод теневых карт достаточен для достижения целей работы. Поскольку сцена имеет всего один источник света, не придётся производить большого количества вычислений при визуализации сцены, а качество теней должно быть достаточным, так как источник света расположен на условно бесконечном расстоянии от сцены, а следовательно равноудалён от всех объектов.

2 Конструкторская часть

2.1 Требования к программному обеспечению

Программа должна предоставлять графический интерфейс со следующим функционалом:

— задавать параметры генерации модели загородного посёлка и генерировать его;

— изменять положение источника света;

— изменять положение и направление камеры.

Программа должна соответствовать следующим требованиям:

— генерация (или конструирование) посёлка происходит при нажатии на соответствующую кнопку в интерфейсе;

— генерация осуществляется в соответствии с параметрами, заданными пользователем;

— программа должна корректно обрабатывать ввод некорректных данных.

2.2 Описание структур данных

Структуры данных, описывающие содержимое сцены

1) сцена представляет собой список, содержащий указатели на объекты сцены (дома, дороги и т.п.);

2) объекты сцены включают в себя следующие данные:

   — массив поверхностей;

   — матрица преобразований.

3) поверхность включает в себя следующие данные:

   — массив вершин;

   — вектор нормали;

   — спектральные характеристики.

4) вершины включают в себя следующие данные:

   — координаты вершины.

5) источник света включает в себя следующие данные:

   — координаты источника.

6) камера включает в себя следующие данные:

   — матрица преобразований.

Следует отметить, что матрица преобразований камеры описывает как её координаты, так и её направление.

Камера с единичной матрицей трансформации находится в точке с координатами $(0, 0, 0)$ и направлена вдоль оси X.

Структуры данных, необходимые для генерации сцены

В процессе генерации, содержимое сцены описывается матрицей (двумерным массивом), каждая ячейка которой означает нахождение определённого объекта на координатах, соответствующих данной ячейке матрицы.

Объекты матрицы содержат следующие данные:

— идентификатор типа объекта;

— таблица возможных идентификаторов типов соседних клеток (по одной записи на каждую соседнюю ячейку).

2.3 Алгоритм построения изображения

К алгоритму построения алгоритма приведены следующие требования:

— на вход подаётся список объектов сцены, описание источника света и камеры;

— на выход подаётся построенное изображение.

Алгоритм построения изображения представлен на рисунке 2.1.

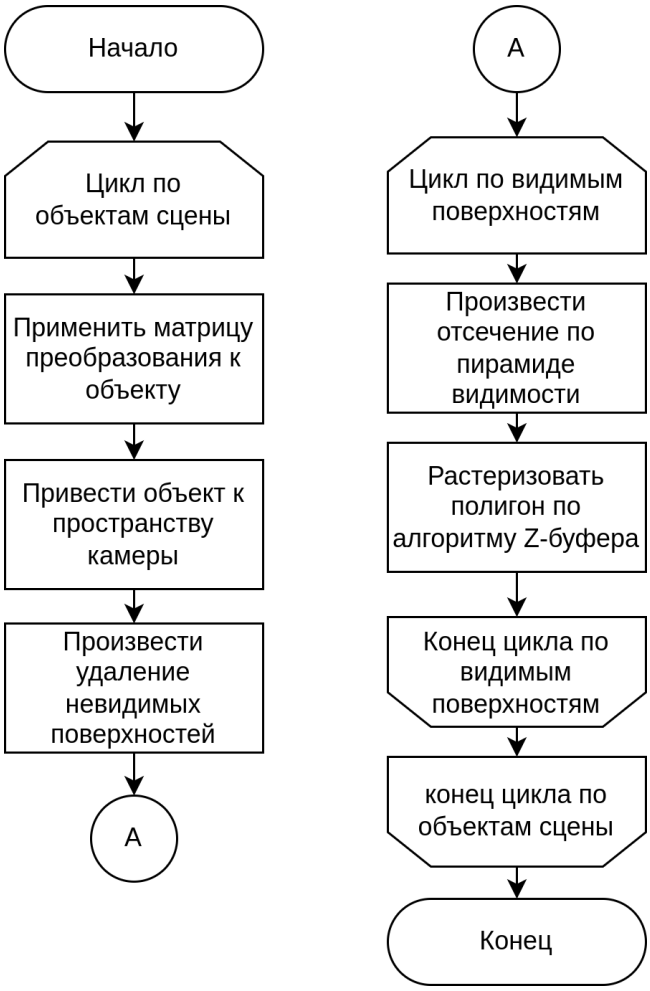

Рисунок 2.1 — Алгоритм построения изображения

В следующих пунктах будут подробнее описаны этапы данного алгоритма.

2.3.1 Матрицы преобразования

В данном алгоритме (и структурах данных) под словосочетанием “матрица преобразования” подразумеваются такая матрица, при умножении всех точек объекта на которую, будет получен преобразованный объект [1].

Данная матрица, как правило, получается как результат произведения нескольких матриц аффинных преобразований.

В данной работе используются следующие аффинные преобразования:

— перенос;

— масштабирование;

— поворот.

Матрица переноса представляется следующим образом:

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ dx & dy & dz & 1 \end{pmatrix}$$

здесь, dx, dy, dz — это смещения по осям Ox, Oy и Oz соответственно

Матрица масштабирования представляется следующим образом:

$$\begin{pmatrix} kx & 0 & 0 & 0 \\ 0 & ky & 0 & 0 \\ 0 & 0 & kz & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

здесь, kx, ky, kz — это коэффициенты масштабирования по осям Ox, Oy и Oz соответственно

Матрицы поворота представляются по-разному для каждой оси.

Матрица поворота вокруг оси Ox представляется следующим образом:

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & cos\alpha & sin\alpha & 0 \\ 0 & -sin\alpha & cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Матрица поворота вокруг оси Oy представляется следующим образом:

$$\begin{pmatrix} cos\alpha & 0 & -sin\alpha & 0 \\ 0 & 1 & 0 & 0 \\ sin\alpha & 0 & cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Матрица поворота вокруг оси Oz представляется следующим образом:

$$\begin{pmatrix} cos\alpha & sin\alpha & 0 & 0 \\ -sin\alpha & cos\alpha & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Во всех матрицах вращения, α — угол поворота вокруг соответствующей оси в радианах.

2.3.2 Приведение к пространству камеры

Поскольку положение и направление камеры описывается матрицей преобразований, приведение точки к пространству координат камеры равносильно применению к данной точке обратных преобразований, применённых к камере.

Например, при смещении камеры на 10 точек по оси Ox, относительно камеры объекты сцены сместятся на -10 точек по той же оси. При повороте камеры на 10 градусов вокруг оси Ox, объекты сцены повернутся на -10 градусов вокруг той же оси относительно координат камеры.

Пользуясь этим свойством, для упрощения перевода координат точек в пространство камеры, можно воспользоваться следующим приёмом: при преобразовании камеры (координат и направления) можно записывать в её матрицу преобразования обратные значения.

Таким образом, для приведения точки к пространству камеры, достаточно применить к этой точке матрицу преобразования камеры.

2.3.3 Удаление невидимых поверхностей

Удаление невидимых поверхностей для этой работы будет проводиться по двум критериям:

— скалярное произведение нормали поверхности и вектора направления камеры;

— попадание поверхности в пирамиду видимости камеры.

Первый критерий позволяет удалить нелицевые (с точки зрения камеры) поверхности. Это позволяет существенно уменьшить количество поверхностей, которые придётся обрабатывать при построении буфера кадра.

Для определения видимости поверхности, достаточно вычислить значения скалярного произведения $(\vec{N}, \vec{V})$, где $\vec{N}$ — вектор нормали поверхности, а $\vec{V}$ — вектор направления камеры. Тогда, при значении $(\vec{N}, \vec{V})$ меньшем нулю, поверхность будет лицевой, а во всех остальных случаях - нелицевой.

Следует отметить, что для правильного определения нелицевых граней, их необходимо также переводить в пространство перспективной проекции, если таковое используется при построении сцены.

Второй критерий позволяет удалить те поверхности которые не попадают в поле зрения камеры. Часто для упрощения вычислений этого критерия, используется не сама поверхность

(которая может задаваться сложной фигурой), а параллелепипед, вписывающий данную поверхность в себя.

Удобно использовать такой параллелепипед, стороны которого параллельны осям системы координат камеры. Тогда, для вычисления его параметров будет достаточно вычислить максимальные и минимальные значения каждой из координат.

После этого, выполняется определение видимость этого параллелепипеда путём пересечения его с пирамидой видимости камеры [3].

Определение пересечения бесконечной пирамиды и прямоугольника тривиально, и сводится к определению определению пересечения прямых.

2.3.4 Алгоритм Z-буфера

Алгоритм Z-буфера используется для растеризации поверхностей объектов. Поскольку под поверхностью в рамках этой работы подразумевается часть плоскости, ограниченная некоторым многоугольником, задача растеризации многоугольника и внесения его в буфер сводится к определению принадлежности точки этому многоугольнику и поиску значения координаты z этой точки.

Поскольку плоскость можно задать тремя точками, достаточно взять любые три точки из многоугольника, которым задаётся поверхность объекта. Сделать это можно следующим образом:

Пусть заданы точки A, B, C, задающие плоскость в пространстве камеры.

$$\vec{N} = \vec{AB} \times \vec{AC}$$

Теперь, можно записать уравнение плоскости в виде

$$N_x(x - x_a) + N_y(y - y_a) + N_z(z - z_a) = 0$$

Раскрыв скобки, можно перейти к следующему виду:

$$N_x x + N_y y + N_z z - (N_x x_a + N_y y_a + N_z z_a) = 0 \tag{2.1}$$

Поскольку занесение координат в z-буфер можно производить построчно (последовательно для каждой строки буфера), вычисление координаты z можно производить пошаговым способом (в пределах одной строки) [3].

Поскольку в пределах одной строки, $y = const$, глубина пикселя данной строки с координатой $x_1 = x + \Delta x$ можно вычислить по формуле 2.2.

$$z_1 = z - \frac{a}{c} \tag{2.2}$$

Где a и c — коэффициенты уравнения плоскости при x и z из уравнения 2.1

Для первой точки на каждой из строк, значение координаты z придётся вычислять из уравнения плоскости.

Для определения точек, принадлежащих многоугольникам, можно использовать алгоритмы растровой развёртки многоугольников.

На рисунке 2.2 изображён алгоритм Z-буфера.

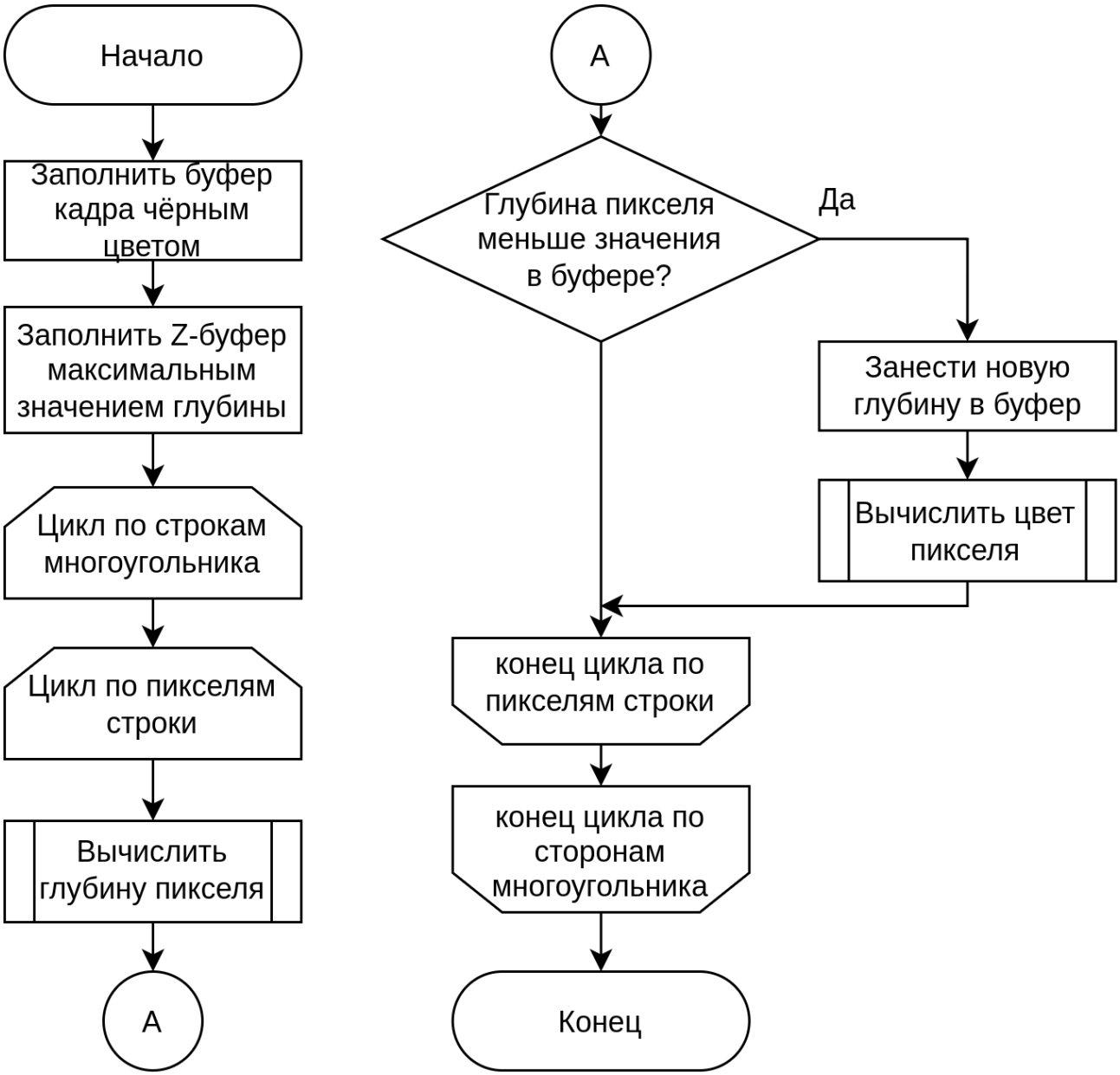

Рисунок 2.2 — Алгоритм z-буфера для многоугольника

2.4 Алгоритм генерации сцены

Для генерации сцены будет использован алгоритм квантового коллапса волновой функции.

Для корректной работы алгоритма крайне важно правильное задание начальных ограничений. Под ограничением подразумевается возможность или невозможность появления опре-

делённой ячейки вплотную к другой ячейке с определённой стороны [2].

Например, если в одной ячейке содержится прямой участок дороги, то к тем сторонам ячейки, которые пересекает дорога, могут быть "присоединены"только такие ячейки, в которых к этим же сторонам будут присоединены другие участки дороги. Это ограничение изображено на рисунке 2.3.

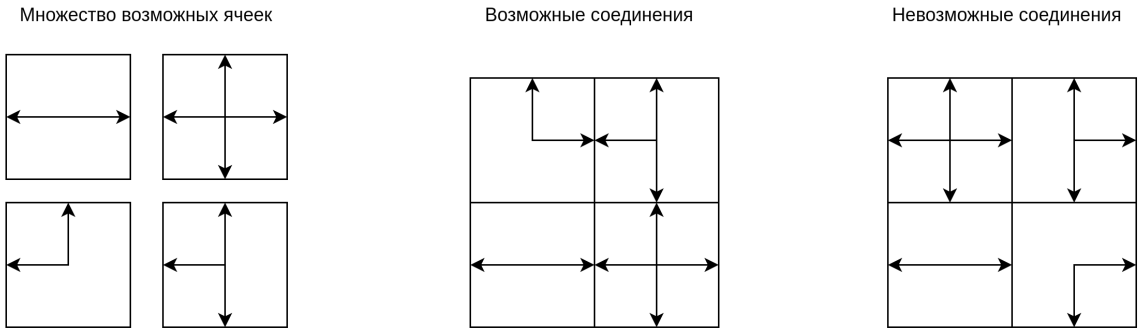

Рисунок 2.3 — Визуализация ограничений генерации

Описание ограничений выполняется вручную для каждой возможной ячейки и её поворота, однако этот процесс можно значительно упростить, если учитывать симметрию и возможные повороты ячеек на уровне программного кода.

Пусть необходимо заполнить матрицу генерации M размеров 50 на 50 ячеек. Изначально все ячейки находятся в состоянии суперпозиции, то есть каждая ячейка может принять любое возможное значение. Возможные значения, в свою очередь, определяются пересечением ограничений соседних ячеек. Если какой-либо из соседей не находится в строго определённом состоянии, его ограничениями будет являться объединение ограничений каждого из возможных состояний.

Пока все ячейки матрицы не примут строго определённое состояние, выполняется цикл, который выбирает ячейку с наименьшим количеством возможных состояний и фиксирует её значение, случайным образом выбирая одно из возможных значений.

Предусмотрена возможность повлиять на вероятность выбора того или иного состояния ячейки путём задания ему некоторого веса. Например, для случая с дорогами, можно повысить вероятность появления поворота и уменьшить вероятность появления перекрёстка.

После обновления ячейки в матрице, ограничения её соседей обновляются.

Алгоритм коллапса волновой функции представлен на рисунке 2.4

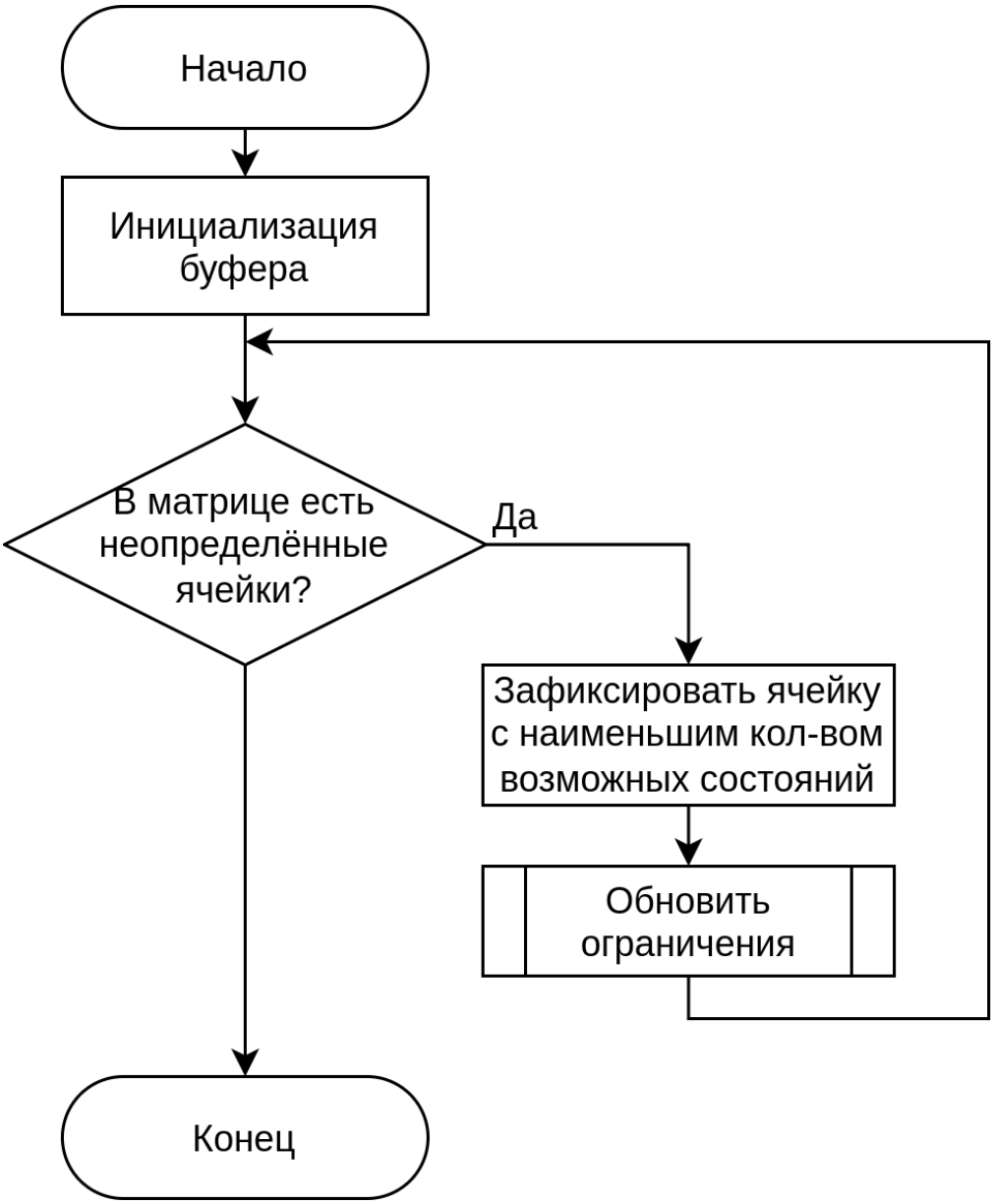

Рисунок 2.4 — Алгоритм квантового коллапса волновой функции

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. John F. Hugens. Computer Graphics Principles and Practice // 3-е издание // Бостон: Addison Wesley Professional // 1263 страницы

2. Heese, Raoul. Quantum Wave Function Collapse for Procedural Content Generation // статья // Institute of Electrical and Electronics Engineers (IEEE) // 13 страниц

3. Д. Роджерс Алгоритмические основы машинной графики // Перевод С.А. Вичес // Москва: Мир 1989г. // 512 страниц

4. А.В. Мальцев Моделирование теней в 3D сценах с помощью каскадных теневых карт в режиме реального времени // статья // Журнал ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ И ВЫЧИСЛИТЕЛЬНЫЕ СИСТЕМЫ // Москва: Российская Академия Наук 2014г. // 52 страницы

5. Michael Abrash. Quake's lighting model: Surface caching. // Онлайн-ресурс: https://www.bluesnews.com/abrash/chap68.shtml // дата обращения: 03.11.2004