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

КУРСОВАЯ РАБОТА

НА ТЕМУ:

Визуализация процесса извержения вулкана

Студент ИУ7-64Б (группа) _____ (подпись, дата) И. С. Бугаков (И.О. Фамилия)

Руководитель курсового проекта _____ (подпись, дата) Т. Н. Романова (И.О. Фамилия)

2025 г.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Вулканическая деятельность оказывает существенное влияние на окружающую среду, экосистемы и ландшафты Земли. Компьютерное модели вулканов активно применяется в научных исследованиях, геологии, мониторинге катастроф и образовательных программах. Современные технологии позволяют не только визуализировать вулканическую активность, но и анализировать её влияние на окружающую среду, что открывает новые возможности для исследований и прогнозирования. [1]

Цель курсовой работы – разработка программного обеспечения для визуализации процесса извержения вулкана. В рамках реализации курсовой работы должны быть решены следующие задачи:

— провести анализ алгоритмов удаления невидимых линий и поверхностей, построения теней и освещения и выбрать наиболее подходящие для решения задачи;

— спроектировать программное обеспечение;

— выбрать средства реализации программного обеспечения и разработать его;

— провести исследование характеристик разработанного программного обеспечения.

1 Аналитический раздел

1.1 Объекты сцены

Сцена состоит из следующих объектов: Гора Дым

1.2 Способы задания модели трехмерных объектов

1.2.1 Каркасная

Модель состоит из вершин и ребер, без данных о внутренних точках поверхностей, ограниченных этими ребрами. [2].

1.2.2 Поверхностная

Поверхность объекта может быть представлена либо аналитическими выражениями, либо полигональной сеткой. Данная модель описывает исключительно внешние геометрические характеристики, такие как форма и границы, но не включает сведения о внутреннем строении и не определяет, где относительно поверхности расположен материал. [2].

1.2.3 Объемная

Поверхность каждого объекта представлена как непрерывная и математически описываемая. Такая модель полностью характеризует трёхмерную форму и включает данные о материале, из которого создан объект. Также содержится информация о направлении внутренней нормали, что позволяет определить расположение материала относительно поверхности. [2].

Для визуализации вулкана была выбрана поверхностная модель.

1.3 Алгоритмы закраски

1.3.1 Закраска Гуро

Метод Гуро основывается на интерполяции интенсивности света, при которой разные точки на грани получают различные значения освещенности. Для этого вычисляются нормали в каждой вершине грани, а затем значения интенсивности интерполируются между точками смежных граней. Нормали

для граней определяются в первую очередь, затем нормали в вершинах рассчитываются как среднее значение нормалей прилегающих граней. На основе этих нормалей вычисляются значения интенсивности с учетом выбранной модели отражения света, после чего полигоны граней закрашиваются с учетом линейной интерполяции интенсивности в вершинах.

Этот метод эффективно работает для небольших граней, находящихся далеко от источника света. Однако, когда грань становится достаточно большой, расстояние от источника света до её центра будет меньше, чем до вершин, что должно привести к более яркому освещению центра. Из-за линейной интерполяции, применяемой в методе, этого эффекта не удается достичь, что приводит к неестественному распределению освещенности на некоторых участках грани. [2].

1.3.2 Закраска Фонга

Метод закраски по Фонгу использует интерполяцию векторов нормалей вместо простой интерполяции интенсивности вдоль сканирующей строки. Это позволяет точно моделировать кривизну поверхности и создавать изображения с высокой реалистичностью. Такой подход особенно эффективен для симуляции зеркальных отражений, поскольку он обеспечивает плавные переходы освещенности и высокую детализацию. В результате метод Фонга минимизирует эффект полос на поверхностях и создает более гладкие и детализированные переходы света, что способствует получению естественного и детализированного изображения.

Для визуализации вулкана выбран метод закраски Гуро, т.к он обеспечивает сглаженные переходы освещения между полигонами, позволяет корректно передать форму и основные градиенты рельефа.

1.4 Алгоритмы удаления невидимых линий и поверхностей

1.4.1 Алгоритм Робертса

Алгоритм Робертса является первым известным методом удаления невидимых линий и работает в объектном пространстве. Он предназначен

для обработки выпуклых объектов и выполняет удаление рёбер и граней, экранируемых самим телом. Далее оставшиеся видимые рёбра каждого объекта сравниваются со всеми другими объектами сцены, чтобы определить, какие их части оказываются перекрытыми. Метод обладает вычислительной сложностью, возрастающей пропорционально квадрату числа объектов. [2].

1.4.2 Алгоритм Варнока

Алгоритм работает в контексте изображения, выполняя анализ содержимого окна. Его задача — определить, является ли окно пустым или слишком сложным для визуализации. В случае, если содержимое окна слишком детализировано, алгоритм рекурсивно разделяет окно на более мелкие сегменты, пока каждый из них не станет достаточно простым для отображения или не достигнет минимального разрешения. Основным недостатком такого метода является увеличение вычислительных затрат и сложности из-за использования рекурсии, что может привести к замедлению обработки изображений с высоким уровнем детализации. [2].

1.4.3 Алгоритм z – буфера

Этот алгоритм для удаления невидимых поверхностей является одним из базовых. Он функционирует в пространстве изображения, расширяя концепцию буфера кадра. Буфер кадра используется для хранения атрибутов (например, интенсивности) каждого пикселя на изображении. Вместе с буфером кадра используется Z-буфер, представляющий собой специализированную память для хранения координат Z (глубины) каждого видимого пикселя. В процессе работы глубина нового пикселя, который требуется добавить в буфер кадра, сравнивается с глубиной пикселя, уже записанного в Z-буфер. Если новый пиксель находится ближе к наблюдателю, чем тот, который уже находится в буфере, его значения добавляются в буфер кадра, и Z-буфер обновляется соответствующей глубиной. Если же новый пиксель дальше, то никаких изменений не происходит. Таким образом, алгоритм для каждой точки (x, y) вычисляет максимальное значение функции Z(x, y). [2].

1.4.4 Алгоритм трассировки лучей

В этом методе для каждого пикселя на изображении определяется ближайшая грань, к которой этот пиксель относится. Для этого через пиксель направляется луч, который пересекает все грани, и из этих пересечений выбирается ближайшее. Алгоритм трассировки лучей включает несколько шагов. Сначала необходимо найти уравнение луча, который проходит через рассматриваемый пиксель и точку взгляда наблюдателя. Вычисляется точка пересечения этого луча с ближайшим объектом сцены. После этого пиксель закрашивается цветом объекта, с которым луч пересекся, и процесс продолжается для следующего пикселя. В случае, если пересечений с объектами нет, пиксель закрашивается цветом фона, и алгоритм переходит к следующему пикселю. Трассировка лучей учитывает процессы отражения, преломления и прохождения через поверхности объектов, заканчиваясь, когда луч встречает непрозрачный объект, который виден наблюдателю. [2].

Для моделирования вулкана был выбран Z-буфер, т. к. метод позволяет эффективно работать с объектами, представленными большим числом полигонов, например, реальными горными ландшафтами.

1.5 Анализ моделей освещения

1.5.1 Модель Ламберта

Модель Ламберта является одной из простейших моделей освещения, в которой учитывается только идеальное диффузное отражение света от поверхности. В этой модели предполагается, что свет, падающий на точку поверхности, равномерно рассеивается во всех направлениях полупространства, и освещенность в точке зависит лишь от плотности света в этой точке. Освещенность линейно зависит от косинуса угла падения света, при этом положение наблюдателя не имеет значения, так как диффузно отраженный свет равномерно рассеивается во всех направлениях.

Основной закон, описывающий диффузное отражение, — это закон косинусов Ламберта, который утверждает, что интенсивность отраженного света пропорциональна косинусу угла падения света на поверхность. Этот закон

выражается следующим уравнением:

$$I = I_l k_d \cos(\theta), \tag{1.1}$$

где:

— I — интенсивность отраженного света,

— I_l — интенсивность падающего света от точечного источника,

— k_d — коэффициент диффузного отражения,

— θ — угол между направлением падающего света L и нормалью к поверхности N.

Закон может быть также представлен в виде, использующем нормированный вектор нормали к поверхности и нормированный вектор направления падающего луча света:

1.5.2 Модель Фонга

Модель освещения Фонга сочетает три компонента: фоновое (амбентное) освещение, диффузное рассеяние и зеркальное (спекулярное) отражение. Модель Фонга представляет собой комбинированную модель освещения, которая учитывает как диффузное, так и зеркальное отражение света, а также фоновое освещение. Она особенно полезна для моделирования ярких блестящих объектов, так как зеркальное отражение вызывает появление световых бликов на поверхностях с высокой степенью отражения, таких как металлы или влага.

Коэффициент зеркального отражения в модели Фонга зависит от угла падения света. Даже при перпендикулярном падении света только часть энергии отражается зеркально, а остальная часть либо поглощается, либо рассеивается диффузно. Эти соотношения зависят от материала поверхности и длины волны света. Например, для различных материалов (металлы, стекло, матовые поверхности) коэффициенты отражения будут различаться, что влияет на видимость бликов и отражений.

Формула модели освещения Фонга:

$$I = I_a \cdot K_a + I_d \cdot K_d \cdot (L \cdot N) + I_s \cdot K_s \cdot (R \cdot V)^n.$$

Модель Ламберта учитывает только диффузное отражение света, что подходит для визуализации вулкана, где поверхность имеет матовую структуру и рассеивает свет в разных направлениях.

1.6 Алгоритмы построения теней

Алгоритмы затенения для точечных источников света аналогичны алгоритмам удаления невидимых линий и поверхностей, но с тем отличием, что наблюдатель в данном случае перемещается в точку источника света. Алгоритм z-буфера основываются на вычислении видимости объектов через их глубину. В этом методе точка считается затенённой, если на пути от неё к источнику света находится объект, блокирующий прямую видимость. Каждый пиксель анализируется с учётом его глубины, и если глубина объекта, находящегося перед ним, меньше, то пиксель считается затенённым

1.7 Алгоритм для визуализации поведение газов

Уравнения Навье-Стокса описывают движение вязкой жидкости и газа, и могут быть использованы для моделирования различных физических процессов, включая генерацию дыма. Для симуляции дыма в рамках уравнений Навье-Стокса необходимо учитывать динамику потока воздуха и взаимодействие частиц дыма с окружающей средой. Математически состояние жидкости в данный момент времени моделируется как векторное поле скорости, которое назначает вектор скорости каждой точке в пространстве. Например, скорость воздуха в помещении изменяется из-за присутствия источников тепла, сквозняков и других факторов. Скорость воздуха рядом с радиатором будет преимущественно направлена вверх из-за подъема тепла. Распределение скоростей в помещении также представляет собой сложную структуру, что можно наблюдать при восходящем дыме или движении частиц пыли в воздухе. Для визуализации дыма, в том числе в контексте извержения вулкана, можно применить математическую модель, основанную на уравнениях Навье-Стокса, которая описывает эволюцию векторного поля скорости во времени. Уравнения этих моделей дают точное описание того, как скорость жидкости или газа будет изменяться в зависимости от действующих сил, что важно для создания визуальных эффектов дыма, особенно в таких динамич-

ных процессах, как извержения вулканов.. [3]

$$\frac{\partial u}{\partial t} = -(u \cdot \nabla)u + \nu\nabla^2 u + f$$

где:

— u — вектор скорости жидкости или газа,

— $\frac{\partial u}{\partial t}$ — изменение скорости по времени,

— $(u \cdot \nabla)u$ — нелинейный член, описывающий изменение скорости из-за движения жидкости,

— ∇p — градиент давления,

— $\nu\nabla^2 u$ — вязкость (диффузия),

— f — внешние силы (например, гравитация, теплоотдача),

— ρ — плотность жидкости или газа.

$$\frac{\partial \rho}{\partial t} = -(u \cdot \nabla)\rho + \kappa\nabla^2 \rho + S$$

где:

— $\rho_d(x, t)$ — плотность дыма в точке x в момент времени t,

— $(u \cdot \nabla)\rho_d$ — изменение плотности дыма из-за потока жидкости или газа,

— $\kappa\nabla^2\rho_d$ — диффузия дыма (влияние турбулентности или молекулярной диффузии),

— S — источник дыма (например, извержение вулкана, где горячие газы и пепел выбрасываются в атмосферу).

В контексте визуализации дыма, особенно при моделировании извержений вулканов, важно моделировать движение частиц, связанных с этим потоком горячих газов и пепла. Поскольку моделирование каждой частицы дыма требует значительных вычислительных ресурсов, эффективным решением является использование плотности дыма — непрерывной функции, которая определяет количество частиц в каждой точке пространства. Эта плотность обычно принимает значения от нуля (отсутствие дыма) до единицы (максимальное количество частиц в точке). Эволюция плотности через поле скорости жидкости или газа может быть описана с использованием уравнений, аналогичных уравнениям для скорости, но с упрощением, поскольку модель плотности является линейной, в отличие от нелинейности модели для скоро-

сти. Для визуализации дымовых эффектов, в том числе при моделировании вулканических извержений, модель плотности дыма используется для расчета того, как частицы будут распространяться через пространство, учитывая изменения в потоке горячих газов и пепла. Визуализация достигается путем отображения плотности дыма в различных точках, что позволяет создать реалистичные эффекты подъема и распространения дыма, характерные для извержений вулканов.

Вывод

Была выбрана поверхностная модель для визуализации вулкана. Для удаления невидимых линий и поверхностей выбран алгоритм использующий z-буфер. Был рассмотрен алгоритм для визуализации поведение газов.

2 Конструкторский раздел

В конструкторской части разработано программное обеспечение, а также формально описаны применяемые алгоритмы.

2.1 Описание структур данных

!!!!

2.2 Общий алгоритм

2.3 Схемы алгоритм визуализации трехмерной схемы

2.3.1 Алгоритм z-буффер

На рисунке 2.1 представлена схема алгоритма z-буфера.

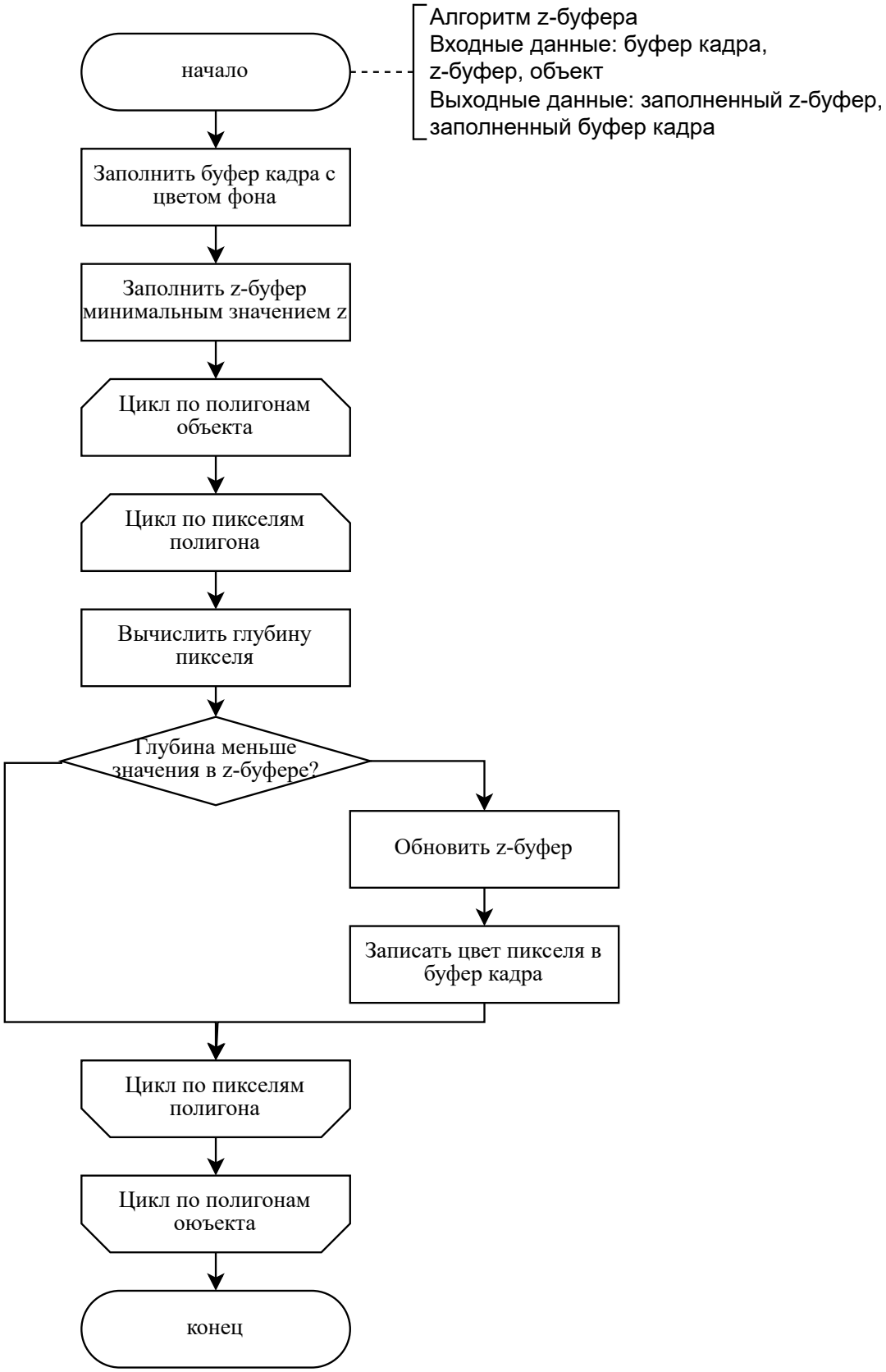

Рисунок 2.1 — Схема алгоритма z-буфера

На рисунке 2.2 представлена схема алгоритма z-буфера.

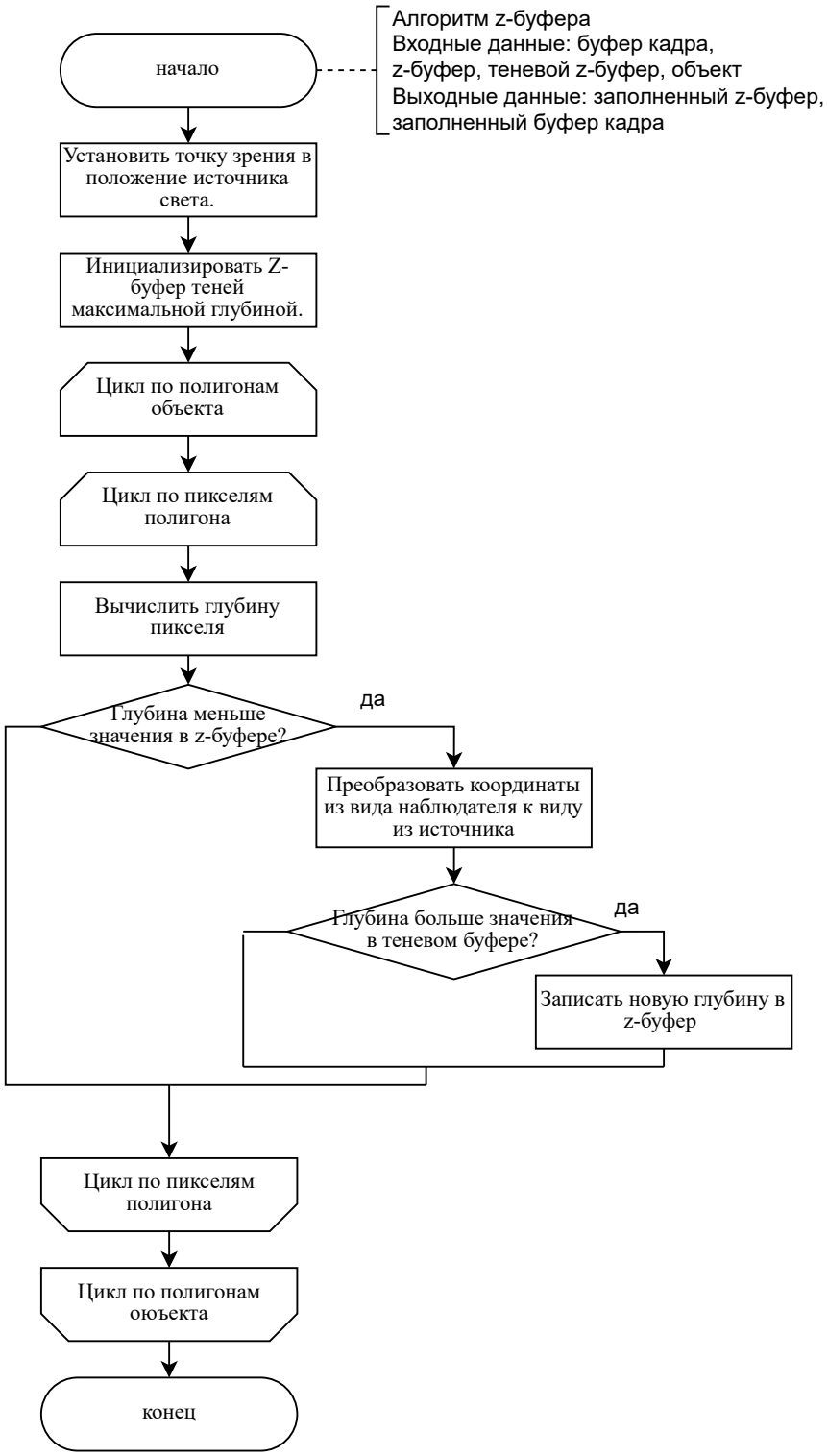

Рисунок 2.2 — Схема алгоритма алгоритма теневого z-буфера

На рисунке 2.3 представлена схема алгоритма имитации ветра.

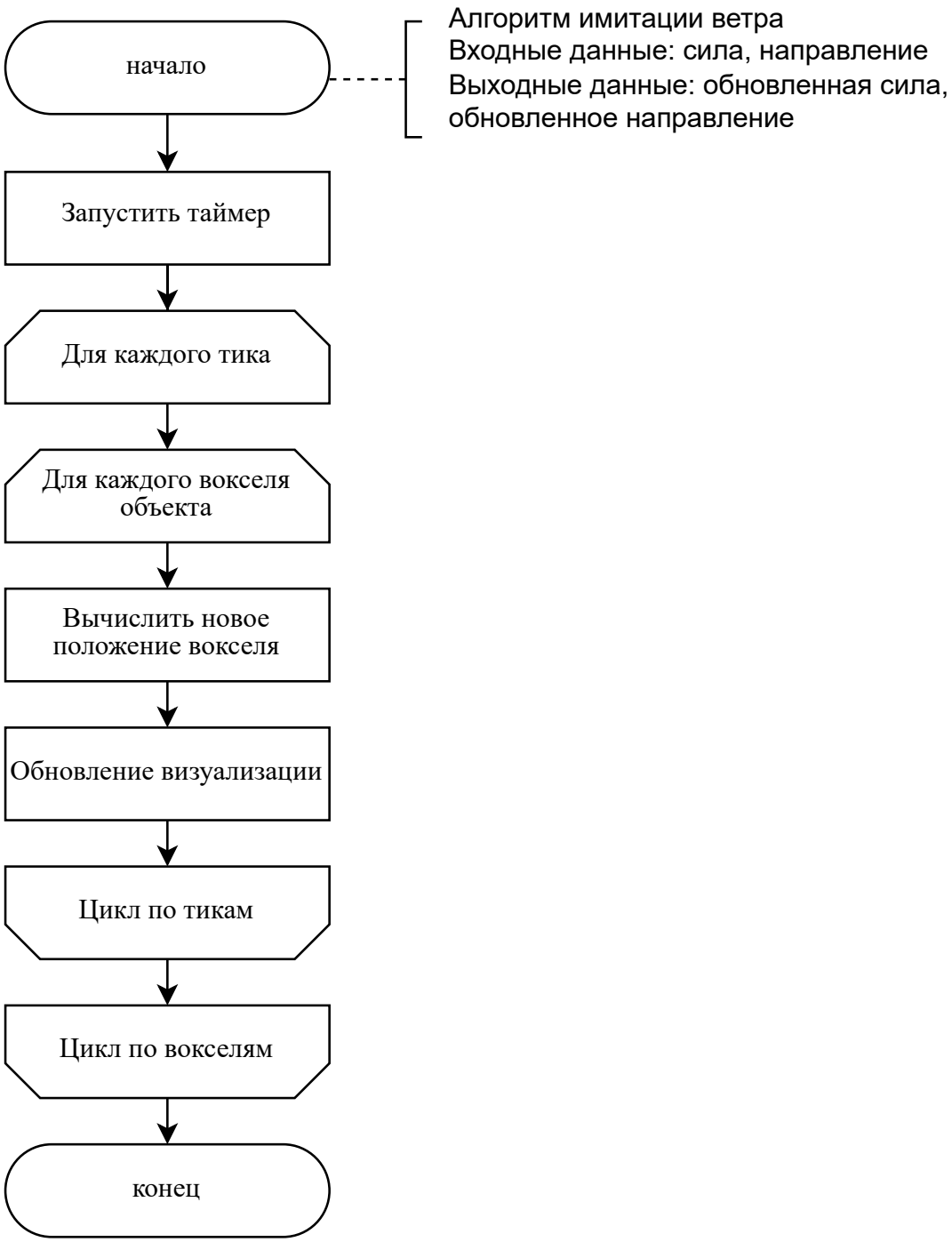

Рисунок 2.3 — Схема алгоритма имитации ветра.

Вывод

В разделе спроектировано разрабатываемое программное обеспечение для визуализации сакуры, качающейся на ветру, приведены схемы алгоритмов.

3 Технологический раздел

В технологическом разделе выбраны и описаны средства реализации программного обеспечения и представлены детали его реализации. В качестве языка программирования был выбран C++, т.к он как он позволяет реализовать все алгоритмы, выбранные в результате проектирования, а также поддерживает все требуемые структуры данных. Для создания пользовательского интерфейса использовался Qt, для визуализации модели SFML.

3.1 Диаграмма классов

На рисунке 3.1 представлена структура программы.

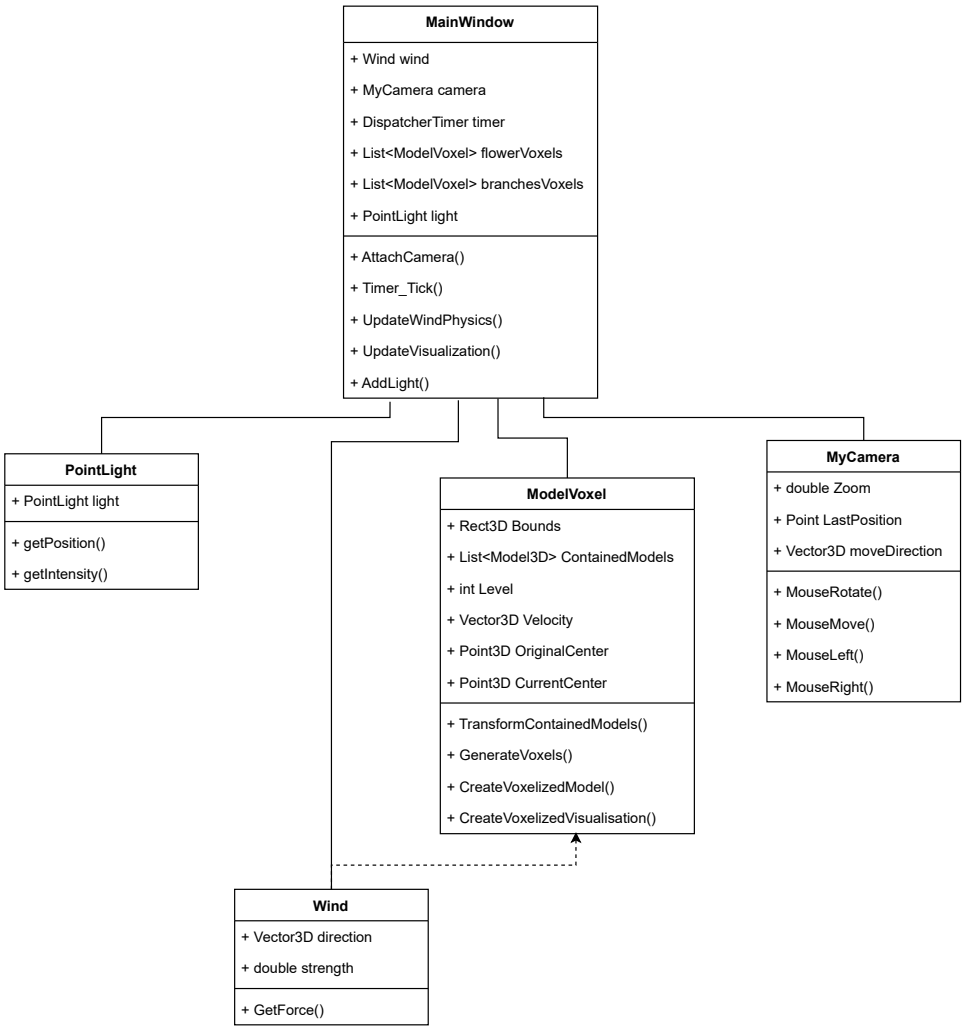

Рисунок 3.1 — Структура программы.

3.2 Графический интерфейс программы

На рисунке 3.2 представлен интерфейс программы. + описание

Рисунок 3.2 — Интерфейс программы.

3.3 Реализация алгоритмов

В листинге 3.1 представлен листинг функции изменения модели под влиянием ветра.

Листинг 3.1 — Фунция изменения модели под влиянием ветра

```
public static void UpdateVoxelPhysics(
List<ModelVoxel> voxels,
Vector3D windForce,
double deltaTime,
bool isLowerLevelFixed)
{
        foreach (var voxel in voxels)
        {
                if (isLowerLevelFixed && (voxel.Level
                  == 0)) continue;

                Vector3D displacement = voxel.
                  CurrentCenter - voxel.OriginalCenter
                  ;
                Vector3D springForce = -displacement *
                  voxel.SpringStiffness;
                Vector3D effectiveWind = windForce;

                voxel.Velocity += (springForce +
                  effectiveWind) * deltaTime;
                voxel.Velocity *= voxel.SpringDamping;

                Point3D newCenter = voxel.CurrentCenter
                    + voxel.Velocity * deltaTime;

                Vector3D totalDisplacement = newCenter
                  - voxel.OriginalCenter;
                if (totalDisplacement.Length >
                  MAX_DISPLACEMENT)
                {
                        totalDisplacement.Normalize();
                        totalDisplacement *=
```

```
                        MAX_DISPLACEMENT;
                    newCenter = voxel.
                        OriginalCenter +
                        totalDisplacement;
                    voxel.Velocity *= 0.5;
                }

                Vector3D movement = newCenter - voxel.
                    CurrentCenter;
                voxel.CurrentCenter = newCenter;

                voxel.Bounds = new Rect3D(
                voxel.Bounds.X + movement.X,
                voxel.Bounds.Y + movement.Y,
                voxel.Bounds.Z + movement.Z,
                voxel.Bounds.SizeX,
                voxel.Bounds.SizeY,
                voxel.Bounds.SizeZ
                );
        }
}
```

3.4 Модульное тестирование

Для модульного тестирования был использован XUnit. Тестировались критически важные функции. Покрытие составило 20,4%. В листинге 3.2 приведен пример тестов для функции отрисовки вокселей.

Листинг 3.2 — Тест для функций отрисовки вокселей

```
public class VoxelGeneratorTests
{
[Fact]
public void ModelVoxel_Initialization_CorrectCenters()
{
        var bounds = new Rect3D(0, 0, 0, 10, 10, 10);
        var voxel = new VoxelGenerator.ModelVoxel(
          bounds, 1);

        Assert.Equal(new Point3D(5, 5, 5), voxel.
          OriginalCenter);
        Assert.Equal(new Point3D(5, 5, 5), voxel.
          CurrentCenter);
}
[Fact]
public void
  TransformContainedModels_AppliesCorrectTransformation
  ()
{
        var bounds = new Rect3D(0, 0, 0, 10, 10, 10);
        var voxel = new VoxelGenerator.ModelVoxel(
          bounds, 1);
        var model = new GeometryModel3D();
        voxel.ContainedModels.Add(model);

        voxel.CurrentCenter = new Point3D(10, 10, 10);
        voxel.TransformContainedModels();

        Assert.IsType<TranslateTransform3D>(model.
          Transform);
}
```

```
[Fact]
public void UpdateVoxelPhysics_ChangesVoxelVelocity()
{
        var voxel = new VoxelGenerator.ModelVoxel(new
          Rect3D(0, 0, 0, 10, 10, 10), 1)
        {
                SpringStiffness = 0.1,
                SpringDamping = 0.9
        };
        List<VoxelGenerator.ModelVoxel> voxels = new()
          { voxel };
        Vector3D windForce = new(1, 0, 0);

        VoxelGenerator.UpdateVoxelPhysics(voxels,
          windForce, 0.1, false);

        Assert.NotEqual(new Vector3D(0, 0, 0), voxel.
          Velocity);
}
```

Вывод

В данном разделе были выбраны средства реализации программного обеспечения, были проведены модульные тесты, все тесты пройдены успешно. Покрытие программы модульными тестами составило 20.4%.

4 Исследовательский раздел

В данном разделе проведено исследование разработанной программы.

Технические характеристики устройства, на котором было проведено исследование

— Процессор: 12th Gen Intel(R) Core(TM) i5-12400F;

— Оперативная память: 32 Гб;

4.1 Цель исследования

Количество вокселей, используемых для разбиения кроны сакуры (листья и лепестки), влияет на качество визуализации дерева. Увеличение числа вокселей позволяет добиться большей детализации изображения, но одновременно повышает вычислительные затраты, увеличивает время рендеринга.

Цель исследования — определить оптимальное число вокселей для представления кроны сакуры. Для этого проведён анализ работы алгоритма визуализации при различном количестве вокселей.

4.2 Исследование

Проводилось исследование для количества вокселей от 500 до 20000. Каждый замер производился 10 раз, было взято среднеарифметическое времени рендеринга и обновления физики. В таблице 4.1 приведены результаты замеров

Таблица 4.1 — Среднее время визуализации и физики в зависимости от количества вокселей

Количество вокселей	$t_{\text{ср}}$(визуализация)	$t_{\text{ср}}$(физика)
500	446.00	19.25
1000	611.50	20.25
2000	722.50	20.00
3000	769.75	26.25
4000	729.75	21.50
5000	853.75	23.75
6000	883.00	24.00
7000	935.50	25.00
8000	990.00	24.67
9000	991.00	25.33
10000	1072.25	26.50
15000	1260.75	28.00
20000	1582.75	30.00

Также был произведен опрос для оценки реалистичности движения сакуры на ветру. Респонденты выбирали самый реалистичный, по их мнению, вариант. В опросе приняли участие 17 человек. Результат опроса приведен в таблице 4.2

Таблица 4.2 — Результаты опроса

Количество вокселей	**Количество проголосовавших**
500	2
1000	1
5000	5
10000	4
15000	2
20000	3

В результате выяснилось, что большинство респондентов проголосовало за вариант с 5000 вокселями, что обеспечивает хорошую реалистичность и относительно высокую скорость рендеринга.

Вывод

В данной части была описана цель исследования, технические характеристики устройства, были приведены результаты исследования.

Была произведена серия экспериментов, в результате которых было вычислено среднее время визуализации и обновление физики для разного количества вокселей. Также был проведен опрос для выяснения оптимального количества пикселей для реалистичности воздействия ветра на сакуру. Оказалось, что большинство респондентов выбрали вариант с 5000 вокселей.

ЗАКЛЮЧЕНИЕ

В ходе работы были решены задачи:

— формализована задача и модель сакуры и ветра;

— проведен анализ и выбраны алгоритмы решения основных задач компьютерной графики: задания трехмерных объектов, удаления невидимых линий и поверхностей, построения теней и освещения;

— спроектировано соответствующее программное обеспечение;

— выбраны средства реализации и реализовано спроектированное программное обеспечение;

— исследованы характеристики разработанного программного обеспечения: реалистичность изображения, скорость работы.

Все задачи были выполнены, цель работы была достигнута.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Starodubtsev I.A., Vasev P.P. Modeling and Visualization of Lava Flows // 2022

2. Д., Роджерс. Алгоритмические основы машинной графики. / Роджерс Д. — М.Мир, 1989. — С. 512.

3. Foster N., Metaxas D. Realistic Animation of Liquids // Graphical Models and Image Processing. - 1996. - №5. - С. 471-483.

4. Прилепко М.А Математические модели представления компьютерной графики // 2012. - №8

5. Емельянова Т. В., Аминов Л.А., Емельянов В. А. Реализация алгоритма удаления невидимых граней // Актуальные проблемы военно-научных исследований. - 2021. - №2. - С. 37-44.

6. Никулин, Е.А. Компьютерная геометрия и алгоритмы машинной графики. – С.Пб: БХВ–Петербург, 2003. – 560с.

7. Большая китайская энциклопедия // URL: https://www.zgbk.com/ecph/words?SiteID=1&ID=153502 (дата обращения: 19.02.2025).

Приложение А

Презентация к курсовой работе состоит из 12 слайдов.