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

ФАКУЛЬТЕТ __«Информатика и системы управления»_____

КАФЕДРА _«Программное обеспечение ЭВМ и информационные технологии»_____

РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

К КУРСОВОЙ РАБОТЕ

НА ТЕМУ:

«Разработка программного обеспечения для моделирования водопада»

Студент ___ИУ7-51Б___ _____ ____Н.П. Артюхин_____
(Группа) (Подпись, дата) (И.О.Фамилия)

Руководитель курсовой работы _____ ___М.Ю. Барышникова__
(Подпись, дата) (И.О.Фамилия)

2022 г.

Содержание

Введение

Компьютерная графика (машинная графика) – это область деятельности, где компьютеры и специальное программное обеспечение используются в качестве инструмента для синтеза (создания), анализа и обработки изображений. В современном мире область применения компьютерной графики довольно широка. Машинная графика чаще всего используется при создании компьютерных игр и в кинематографе.

Методы отрисовки постоянно развиваются, также появляются новые способы. Сегодня наибольшее внимание уделяется алгоритмам получения реалистических изображений, это одна из основных задач компьютерной графики. Данные алгоритмы являются одними из самых затратных по времени и памяти, потому что они должны учитывать физические явления: отражение, преломление, рассеивание, поглощение света. Чем выше точность алгоритмов, то есть чем качественнее полученное на выходе алгоритма изображение, тем выше их сложность, что обычно приводит к увеличению затрат по времени и по памяти.

При создании динамической сцены на каждом временном промежутке необходимо производить расчеты заново, что приводит к большому числу вычислений и временным затратам.

Одна из самых сложных тем для моделирования – жидкости. На сегодняшний день существует серьезная необходимость в эффективной и качественной отрисовке морей, океанов, озер, рек и множества других водоемов [1].

Цель работы – разработать программное обеспечение, обеспечивающее динамическую визуализацию модели искусственного водопада.

Чтобы достигнуть поставленной цели, необходимо решить следующие задачи:

1) описать структуру трехмерной сцены, включая объекты, из которых она состоит;

2) проанализировать существующие алгоритмы, которые можно использовать для моделирования водопада, выбрать наиболее подходящий из них;

3) проанализировать алгоритмы удаления невидимых линий и поверхностей и выбрать наиболее подходящий из них;

4) реализовать выбранные алгоритмы;

5) разработать структуру классов программного обеспечения;

6) провести эксперимент по замеру производительности программного обеспечения в зависимости от используемых языков программирования.

1 Аналитический раздел

1.1 Модель классического водопада

Водопад – это падение воды в реке с уступа, пересекающего речное русло. В отличие от речных порогов, для водопадов характерны резкий перепад высоты речного дна и отвесность падения. Угол падения воды в классических водопадах составляет примерно 90 градусов. Высота водопада должна быть более 1 метра.

Образование водопадов в природе на реке – это очень долгий временной процесс. Обычно водопады возникают при резком перепаде высот, который появился еще до того, как там потекла река, или из-за того, что водяной поток размывает мягкие осадочные породы, вследствие чего происходит обрушение почвы и на границе мягкой и твердой пород образуется вертикальный уступ, с которого и начинает падать вода (рисунок 1.1).

Рисунок 1.1 – Модель водопада

Для более качественного и реалистичного изображения при построении

модели водопада нужно учитывать не только сам поток воды, но и брызги, которые образуется при ударе потока об воду и брызги от самого потока воды (рисунок 1.2).

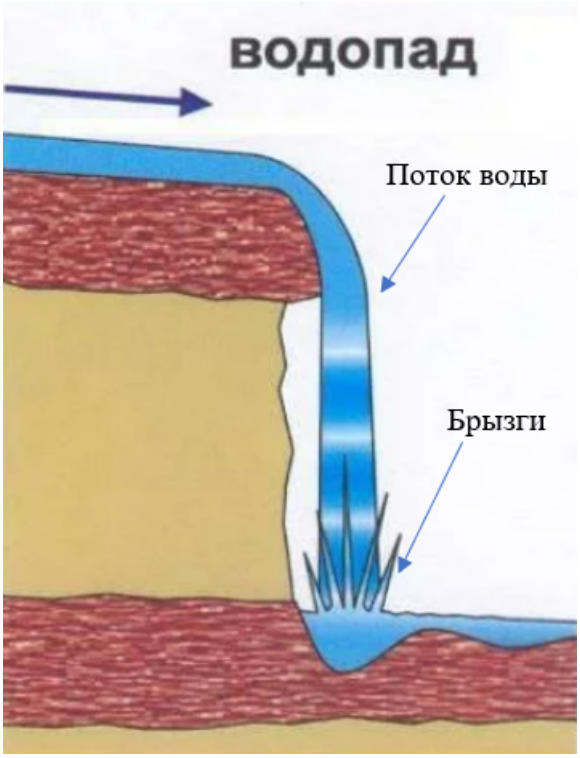


Рисунок 1.2 – Схема водопада

1.2 Формализация модели водопада

Модель водопада будет задаваться следующими характеристиками:

- высота – высота уступа, с которого падает вода (число типа float в интервале от 10% до 80% от высоты экрана);
- ширина – ширина уступа, по которому течет вода (число типа float);
- длина – длина уступа, по которому течет вода (число типа float);
- скорость – скорость падения воды с уступа (число типа float).

Частью сцены будет являться скала, играющая роль уступа, с которого падает вода. Она будет задаваться геометрическим объектом – прямоугольным параллелепипедом, высоту которого сможет изменять пользователь.

1.3 Анализ и выбор метода визуализации водопада

Водопад – это поток воды с частицами брызг. Для моделирования текучей воды существует несколько основных методов, которые будут приведены далее.

Перед выбором алгоритма для моделирования водопада необходимо выделить несколько свойств, которыми должен обладать выбранный алгоритм:

- алгоритм должен быть достаточно эффективным по времени;
- алгоритм должен быть достаточно эффективным по памяти;
- алгоритм должен иметь высокую реалистичность изображения;
- алгоритм не должен иметь очень высокую трудоемкость.

1.3.1 Метод, основанный на уравнении Навье-Стокса

Уравнение Навье-Стокса – это тип нелинейных дифференциальных уравнений в частных производных, которые описывают движение жидких и газообразных сред [2].

$$\frac{d\vec{v}}{dt} = -(\vec{v}\nabla)\vec{v} + \nu \triangle \vec{v} - \frac{1}{\rho}\nabla p + \vec{f}, \tag{1.1}$$

где $\vec{v}$ – векторное поле скорости, t – время, ∇ – оператор набла, ν – коэффициент вязкости, $\triangle$ – векторный оператор Лапласа, ρ – коэффициент плотности жидкости, p – давление, $\vec{f}$ – векторное поле массовых сил (внешние силы, применяемые к жидкости).

Данная формула позволяет вычислить скорость изменения скорости жидкости в точке.

Данное уравнение пока не имеет решения в общем виде, поэтому при разработке алгоритмов используются частные решения. Уравнение Навье-Стокса обычно применяется для математического моделирования сложных моделей природных явлений.

Методы, основанные на этом уравнении, трудно реализуются из-за сложной математики и большого количества трудоемких вычислений, которые придется производить компьютеру при обработке формул. Таким образом, данные методы являются очень неэффективными по времени, и подробное их рассмотрение не имеет смысла.

1.3.2 Сеточные методы

Сетки довольно часто используются при моделировании жидкостей. В каждой ячейке регулярной кубической сетки хранится информация о точке поверхности, задающей картину.

Качество получаемого изображения зависит от количества ячеек, чем больше ячеек, тем выше качество изображения.

Одни из самых распространенных сеточных мпетодов – метод, основанный на уравнении Эйлера [3] и метод моделирования пузырьков воздуха в воде [4].

Преимущества сеточных методов:

- простота визуализации результата симуляции;
- точно известны границы целевой области (возможность встраивания водяного потока в нужное место).

Однако сеточные методы неэффективны по памяти и имеют не очень высокую реалистичность изображения [3, 4], поэтому более подробно рассматриваться они не будут.

1.3.3 Методы, основанные на частицах

Метод частиц основывается на использовании трилинейной интерполяции для каждой частицы для определения ее скорости движения. Каждая частица перемещается в соответствии с определенным инерциальным физическим уравнением.

Преимущество данного метода в том, что он имеет довольно небольшие затраты по ресурсам компьютера. Большое количество частиц позволяет

достичь максимальной точности изображения, но чем больше частиц, тем больше нагрузка на компьютер.

Главным недостатком данного подхода является невозможность точно определить границы, в которых должен находиться водяной поток, а это приводит к появлению дополнительных проверок; также нет возможности создавать извивающиеся водяные потоки.

Наиболее распространенные методы, основанные на частицах, будут рассмотрены далее.

Метод, использующий диаграммы Вороного

Основным компонентом данного метода является аппроксимация геометрии пены путем обработки частиц пузырьков как участков взвешенной диаграммы Вороного. Информация о связности, предоставляемая диаграммой Вороного, позволяет довольно точно моделировать различные эффекты взаимодействия между пузырьками.

Используя ячейки Вороного и веса (взвешенную диаграмму Вороного), можно решить проблему потери объема при моделировании пены, что является общей проблемой во многих подходах [5].

Метод полу-Лагранжа

Метод полу-Лагранжа, объединенный с новым подходом расчета жидкости вокруг объектов, позволяет эффективно решать уравнения движения жидкости, сохраняя при этом достаточно деталей, чтобы получить реалистичное изображение.

Высококачественная поверхность получается из результирующего поля скоростей с использованием нового адаптивного метода для создания неявной поверхности [6].

Метод, основанный на уравнениях движения частиц по криволинейной траектории

В данном методе при моделировании каждая частица считается независимой и для каждой вычисляется ее место в соответствии с физическим уравнением. Частицы случайным образом распределяются по линии течения воды.

В системе частиц каждая отдельно взятая частица будет рассматриваться в независимости от остальных частиц. В качестве приоритетного способа реализации модели частицы, падающей с вершины уступа водопада, используется подход, основанный на криволинейном равноускоренном движении.

Таким образом, можно получить качественный результат, используя известные физические законы для движения частицы.

Для моделирования потока воды в качестве основного уравнения берется закон равноускоренного движения [7]:

$$y = y_0 + v_0 t + \frac{at^2}{2}, \tag{1.2}$$

где y_0 – начальное положение частицы, v_0 – начальная скорость движения частицы, t – время, a – ускорение движения частицы.

При этом скорость частицы при равноускоренном движении вычисляется следующим образом:

$$v = v_0 + at, \text{ где a} = \text{const} \tag{1.3}$$

Ускорение вычисляется следующим образом:

$$a = \frac{dv}{dt} \tag{1.4}$$

Нахождение скорости, в зависимости от направления, которое было задано частице.

Вектор направления в каждый момент времени вычисляется, путем сложения вектора направления движения и вектора гравитации. При этом по-

лучается новый вектор направления (результирующий вектор), тогда скорость будет вычисляться следующим образом:

$$\vec{v} = v * \vec{R}, \tag{1.5}$$

где $\vec{v}$ – вектор скорости частицы, v – скалярная скорость частицы, $\vec{R}$ – результирующий вектор вектора направления и вектора гравитации.

1.3.4 Комбинированные методы

Комбинированные методы – это методы, полученные в результате объединения методов, основанных на частицах и на сетках. Совместное использование двух методов помогает одновременно с потоком воды моделировать и другие виды водяных потоков. Данный метод является предпочтительным при создании полноценной системы.

Наиболее распространённые комбинированные методы – метод, напрямую использующий методы на основе сеток и частиц, в котором все частицы рассматриваются как простые невзаимодействующие точечные массы, которые обмениваются массой и моментом с жидкостью поля высоты [8] и метод, моделирующий пузырьки воздуха и пену [9].

Однако комбинированные методы являются трудоемкими [8, 9], поэтому подробнее они рассматриваться не будут.

1.3.5 Вывод

Для моделирования водопада выбран метод, основанный на частицах.

Данный метод позволяет смоделировать не только сам водопад (основной поток воды), который состоит из большого количества частиц, но и брызги и пену, которые тоже состоят из частиц воды. Также у данного метода довольно небольшие затраты по времени и по памяти по сравнению с другими рассмотренными выше методами. Качество изображения можно повысить путем увеличения количества частиц.

1.4 Анализ и выбор формы задания трехмерной модели

1.4.1 Основные формы задания моделей

1) Каркасная (проволочная) модель

   Хранится информация только о вершинах и ребрах объекта.

   Преимущество – простая.

   Недостаток – далеко не всегда передает правильную информацию о моделируемом объекте (например, все ребра видимы, при наличии отверстий не понятно какие грани связывает отверстие).

2) Поверхностная модель

   Поверхностная модель объекта является оболочкой объекта, она пустая внутри. Хранятся только информация о внешних геометрических параметрах объекта. Данный тип модели чаще всего используется в компьютерной графике. При этом могут использоваться различные типы поверхностей, ограничивающих объект, такие как полигональные модели, поверхности второго порядка и другие.

   Преимущество – довольно точно передает представление о форме моделируемого объекта.

   Недостаток – не подходит для создания изделий, так как нет информации о материале.

3) Объемная (твердотельная) модель

   Твердотельная модель содержит информацию о том, где находится материал, а где пустота. Это делается с помощью указания направления внутренней нормали.

   Преимущество – наиболее полное представление о моделируемом объекте.

Недостаток – довольно сложное представление объекта, для большинства задач компьютерной графики данная модель является избыточной.

Вывод

Для представления скалы (уступа водопада) была выбрана поверхностная модель, так как каркасные модели могут привести к неправильному восприятию формы моделируемого объекта, а реализация твердотельной модели потребует большого количества ресурсов на воспроизведение деталей, которые не являются необходимыми для решения поставленной задачи.

1.4.2 Способы задания поверхностной модели

1) Аналитический способ

   Данный способ характеризуется описанием модели объекта, доступной в неявной форме, то есть для получения визуальных характеристик необходимо дополнительно вычислять некоторую функцию, которая зависит от некого параметра.

2) Полигональная сетка

   Данный способ характеризуется совокупностью вершин, граней и ребер, которые определяют форму многогранного объекта в трехмерной компьютерной графике.

Вывод

В качестве способа задания поверхностной модели была выбрана полигональная сетка, чтобы избежать дополнительных вычислительных затрат, которые есть в аналитическом способе. Также с помощью полигональной сетки могут быть описаны сложные модели, и данный способ задания

1.5 Анализ и выбор алгоритма удаления невидимых ребер и поверхностей

Перед выбором алгоритма удаления невидимых ребер выделим несколько свойств, которыми должен обладать выбранный алгоритм, чтобы обеспечить оптимальную работу и реалистичное изображение.

Свойства:

- алгоритм должен быть эффективным по времени;
- алгоритм должен быть довольно эффективным по памяти;
- алгоритм может работать в пространстве изображения;
- алгоритм должен иметь высокую реалистичность изображения.

1.5.1 Алгоритм, использующий Z-буфер

Суть алгоритма – это использование двух буферов: буфера кадра, в котором хранятся атрибуты каждого пикселя, и Z-буфера, в котором хранятся информация о координате Z для каждого пикселя.

Начальная инициализация буферов:

- z-буфер заполняется минимально возможным значением Z;
- буфер кадра заполняется фоновым значением цвета.

Каждый многоугольник преобразуется в растровую форму и записывается в буфер кадра. В процессе подсчета глубины нового пикселя, он сравнивается с тем значением, которое уже лежит в z-буфере. Если новый пиксель расположен ближе к наблюдателю, чем предыдущий, то он заносится в буфер кадра и происходит корректировка z-буфера [10].

Для решения задачи вычисления глубины Z каждый многоугольник описывается уравнением $ax + by + cz + d = 0$. При $c = 0$ многоугольник для наблюдателя вырождается в линию.

Для некоторой сканирующей строки $y = const$ можно рекуррентно высчитывать z_1 (глубину пикселя) для каждого $x_1 = x + dx$:

$$z_1 = z - \frac{a}{c}, \text{ при } dx = 1 \text{ (шаг растра)} \tag{1.6}$$

Для невыпуклых многогранников предварительно потребуется удалить нелицевые грани.

Преимущества алгоритма:

- простота реализации;
- отсутствие сортировок;
- линейный рост времени работы алгоритма;
- работает в пространстве изображения;
- достаточно высокая реалистичность изображения.

Недостатки алгоритма:

- сложная реализация эффекта прозрачности;
- неэффективность по памяти (большой расход памяти).

Для экономии памяти можно использовать алгоритм построчного сканирования, использующий z-буфер (в z-буфере будет храниться одна сканирующая строка).

1.5.2 Алгоритм Робертса

Данный алгоритм работает в объектном пространстве, решая задачу только с выпуклыми телами в виде проволочной (каркасной) модели. Этапы алгоритма Робертса [10]:

- подготовка исходных данных (формирование матрицы тела);
- удаление ребер, экранируемых сами телом;
- удаление невидимых ребер, экранируемых другими телами сцены;

- удаление невидимых частей новых ребер, появившихся при протыкании тел.

Преимущества алгоритма:

- высокая точность вычислений.

Недостатки алгоритма:

- сложность алгоритма увеличивается нелинейно – $O(N^2)$, где N – число объектов;
- тела сцены должны быть выпуклыми (алгоритм усложняется, так как необходима проверка на выпуклость и если тело невыпуклое, то его необходимо разбить на выпуклые тела);
- сложность реализации алгоритма.

Таким образом, Алгоритм Робертса не подходит для решения поставленной задачи из-за высокой сложности реализации алгоритма, что приведет к низкой производительности.

1.5.3 Алгоритм, использующий список приоритетов (алгоритм художника)

Данный алгоритм работает аналогично тому, как художник рисует картину, то есть сцена строится, начиная с самых дальних объектов и заканчивая самыми близкими к наблюдателю.

Наиболее распространенная реализация алгоритма – сортировка по глубине (Zmin), которая заключается в том, что произвольное множество граней сортируется по ближнему расстоянию от наблюдателя, причем первым в списке окажется самый дальний от наблюдателя [10].

Затем отсортированные грани выводятся на экран в порядке от самой дальней до самой ближней. Данный метод работает лучше для построения сцен, в которых отсутствуют пересекающиеся грани.

Преимущества алгоритма:

- достаточно эффективен по памяти (затрачивает меньше памяти, чем алгоритм, использующий Z-буфер).
- работает в пространстве изображения.

Недостатки алгоритма:

- сложность реализации при пересечении граней на сцене (добавление проверок (тестов), разбиение многоугольников (граней) при циклическом экранировании);
- неэффективен по времени (из-за отрисовки невидимых граней);
- низкая реалистичность изображения.

Таким образом, данный алгоритм не подходит для решения поставленной задачи из-за низкой реалистичности изображения и неэффективности по времени, так как он отрисовывает все грани, включая невидимые.

1.5.4 Алгоритм Варнока

Алгоритм Варнока основан на разбиении картинной плоскости на части, для каждой из которых исходная задача может быть решена достаточно просто. Алгоритм Варнока работает в пространстве изображения.

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 фрагмента не станет достаточно простым для визуализации или его размер не достигнет требуемого предела разрешения (обычно в качестве такого предела выбирают окно размером в 1 пиксель) [10].

Преимущества алгоритма:

- эффективен по времени, если рассматриваемая область содержит мало информации (тогда картинная плоскость будет разбита на минимальное число частей и алгоритм справится со своей задачей довольно быстро);
- работает в пространстве изображения.

Недостатки алгоритма:

- неэффективен по времени, если область содержит много информации (придется разбивать картинную плоскость на большое число частей, и анализировать каждую из них).

Таким образом, данный алгоритм не подходит для решения поставленной задачи, так как будет неэффективен по времени: картинную плоскость с большой вероятностью придется разбивать на большое число частей из-за частиц воды.

1.5.5 Алгоритм обратной трассировки лучей

Суть алгоритма – наблюдатель видит объект с помощью испускаемого света, который согласно законам оптики доходит до наблюдателя некоторым путем. Алгоритм называется «обратной трассировкой лучей», так как более эффективно отслеживать пути лучей не от источника к наблюдателю, а в обратном направлении, то есть от наблюдателя к объекту [10]. Преимущества алгоритма:

- высокая реалистичность получаемого изображения;
- вычислительная сложность не сильно зависит от сложности сцены.

Недостатки алгоритма:

- неэффективность по времени из-за большого количества вычислений – низкая производительность.

Таким образом, данный алгоритм не подходит для решения поставленной задачи, так он неэффективен по времени, а данный критерий очень важен при моделировании водопада из-за большого числа динамических объектов (частиц воды).

1.5.6 Вывод

Для удаления невидимых линий выбран алгоритм, использующий Z-буфер. Данный алгоритм довольно просто реализуется, эффективен по

времени из-за отсутствия сортировок и позволяет получить достаточно высокую реалистичность изображения, что необходимо для моделирования водопада.

1.6 Выбор метода отрисовки изображения

Рендеринг (отрисовка) – это термин в компьютерной графике, обозначающий процесс получения изображения по модели с помощью компьютерной программы.

Чтобы моделируемый водопад был реалистичным, программа должно работать быстро, стандартная графическая библиотека не подойдет для отрисовки частиц потока воды из-за большой нагрузки, поэтому необходимо использовать API, которое будет использовать графический ускоритель для отрисовки изображения.

Основные API для рендеринга изображения будут рассмотрены далее.

1.6.1 Vulkan

Vulkan – кроссплатформенный API для 2D- и 3D-графики, представленный Khronos Group [11].

Vulkan API изначально был известен как «новое поколение OpenGL» или просто «glNext», но после анонса компания отказалась от этих названий в пользу названия Vulkan. Как и OpenGL, Vulkan позволяет с высокой производительностью отображать в реальном времени различные приложения с 3D-графикой, такие как игры или интерактивные книги на всех платформах, а также обеспечивает более высокую производительность и меньшую нагрузку на процессор.

Преимущества:

- высокая производительность;
- открытый код;
- кроссплатформенность (Linux, MacOS, Windows).

Недостатки:

- работа на глубоком уровнем GPU;
- библиотека для работы с API на различных языках программирования отличается полнотой функционала.

1.6.2 OpenGL

OpenGL – спецификация, определяющая платформонезависимый (независимый от языка программирования) программный интерфейс для написания приложений, использующих двумерную и трёхмерную компьютерную графику. Разрабатывается в США и Европе, имеет тип лицензий GNU-/EU/ [12].

Спецификация OpenGL определяет, каким должен быть результат/вывод каждой функции, и как она должна выполняться. Однако реализация этой спецификации уже зависит от конкретных разработчиков.

Если рассмотривать поддержку графической обработки для современных систем в целом, то видно, что она базируется на понятии конвейера: графические данные проходят последовательно несколько этапов обработки – выходные данные одного этапа сразу передаются на вход следующего.

Преимущества:

- высокая производительность;
- гибкая структура, которая позволяет изменять параметры выводимых объектов;
- кроссплатформенность (Linux, MacOS, Windows);
- гибкая структура, возможность изменять параметры выводимых объектов;
- библиотеки для работы с API на различных языках программирования.

Недостатки:

- сложная работа с новыми возможностями GPU;
- нельзя использовать для работы с мышью и клавиатурой (нужно отдельное API).

1.6.3 DirectX

DirectX – это набор API, разработанных для решения задач, связанных с программированием под Microsoft Windows. Он нацелен на создание менее сложного драйвера и API, более близкого к архитектуре современных графических процессоров. DirectX фокусируется на рендеринге в реальном времени, поэтому он предназначен для разработчиков игр и систем автоматизированного проектирования (CAD). Наиболее широко используется при написании компьютерных игр [13].

Преимущества:

- эффективность по времени (быстрая отрисовка изображения);
- высокое качество изображения.

Недостатки:

- новые версии только для самого современного оборудования (не поддерживается старыми видеокартами);
- некроссплатформенная (только для Windows).

1.6.4 Вывод

В качестве API для отрисовки изображения системы частиц водопада был выбран OpenGL. Он отличается кроссплатформенностью от DirectX, потому что последний используется исключительно для операционной системы Windows, а Vulkan не имеет библиотек для некоторых языков программирования, которые предоставляли бы весь функционал API.

1.7 Анализ и выбор модели освещения

Модели освещения используются для вычисления интенсивности света в данной точке на поверхности модели. Все модели освещения делятся на простые (локальные) и глобальные. В простых моделях освещения учитывается свет только от источников и ориентация поверхности, а в глобальных моделях учитывают еще и свет, отраженный от других поверхностей сцены (вторичные источники).

Простая модель представляет собой сумму трех составляющих света: рассеянная, диффузная и зеркальная.

Рассеянная составляющая света (фоновое освещение) присутствует в любой точке сцены, не зависит от пространственных координат освещаемой точки и источника и описывается следующей формулой:

$$I_p = i_p * k_p, \tag{1.7}$$

где i_p – интенсивность рассеянного света (обычно задается константой для всей сцены), k_p – коэффициент диффузного отражения рассеянного света (свойство материала воспринимать фоновое освещение).

1.7.1 Модель Ламберта

Модель Ламберта моделирует идеальное диффузное освещение, свет при попадании на поверхность рассеивается равномерно во все стороны [14]. При такой модели освещения учитывается только ориентация поверхности ($\vec{n}$ – вектор нормали в данной точке поверхности), направление источника света ($\vec{S}$ – вектор падения света), θ – угол между векторами $\vec{n}$ и $\vec{S}$ (рисунок 1.3).

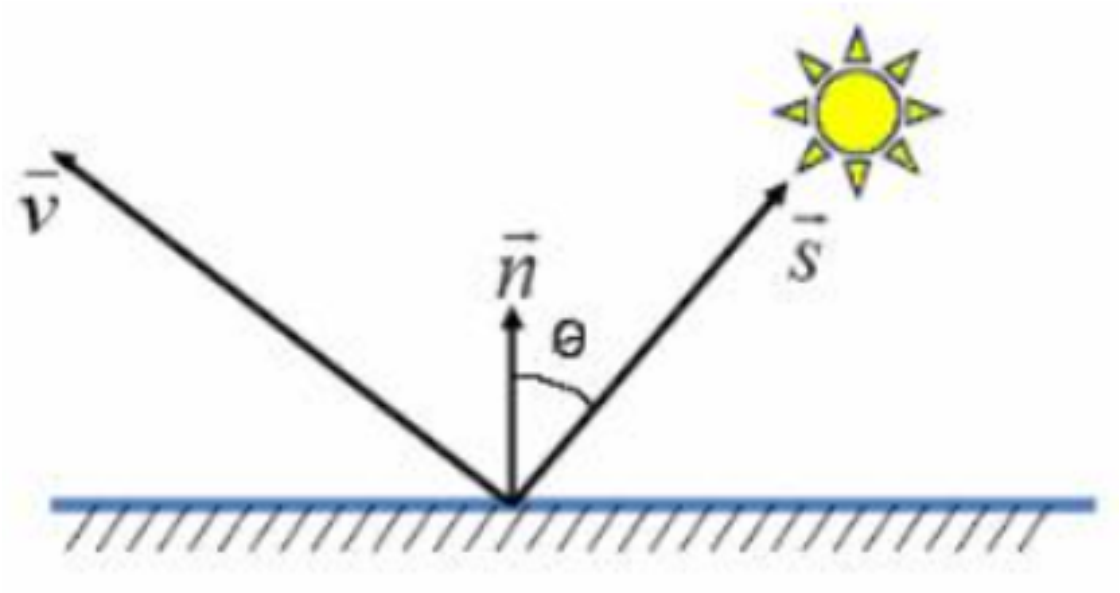


Рисунок 1.3 – модель Ламберта

Эта модель является одной из самых простых моделей освещения и часто является основой для большинства других моделей. Поверхность при использовании данной модели будет выглядеть одинаково яркой со всех направлений, без бликов на телах сцены.

1.7.2 Модель Фонга

Модель Фонга – классическая модель освещения, она добавляет в модель Ламберта зеркальную составляющую, то есть она представляет собой комбинацию диффузной и зеркальной составляющих. При использовании данной модели освещения кроме равномерного освещения на материале могут появляться блики. Местонахождение блика на объекте определяется из закона равенства углов падения и отражения. Чем ближе наблюдатель к углам отражения, тем выше яркость соответствующей точки [14].

Падающий луч ($\vec{S}$) и отраженный луч ($\vec{r}$) лежат в одной плоскости с нормалью ($\vec{n}$) к отражающей поверхности в точке падения, $\vec{v}$ - вектор, направленный на наблюдателя (рисунок 1.4). Нормаль делит угол между лучами на две равные части.

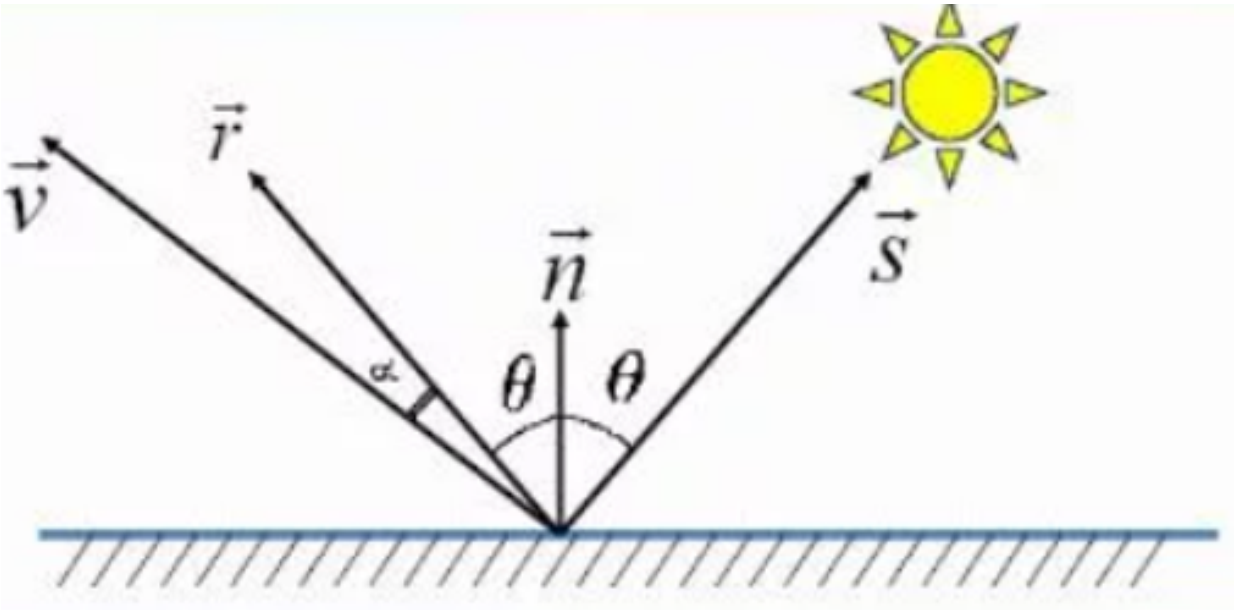


Рисунок 1.4 – модель Фонга

1.7.3 Вывод

В качестве модели освещения была выбрана модель Ламберта, так как она более проста в реализации, чем модель Фонга. Также для модели Ламберта необходимо производить меньше вычислений, чем для модели Фонга, следовательно использование модели Ламберта более эффективно по времени, чем модели Фонга.

1.8 Анализ существующего ПО

В данном разделе будет рассмотрено самое распространенное существующее программное обеспечение, которое можно использовать для моделирования водопада.

1.8.1 Autodesk 3ds Max

Autodesk 3dsMax – профессиональное программное обеспечение для 3D-моделирования, анимации и визуализации при создании игр и проек-

тировании [15]. В настоящее время разрабатывается и издается компанией Autodesk.

3ds Max располагает большим количеством средств для создания разнообразных по форме и сложности трёхмерных компьютерных моделей, реальных или фантастических объектов окружающего мира, с использованием разнообразных техник и механизмов.

Недостаток данного ПО – для коммерческих целей программа платная, но существует бесплатная лицензия для обучения.

Пример создания водопада в данной программе показан ниже на рисунке 1.5.

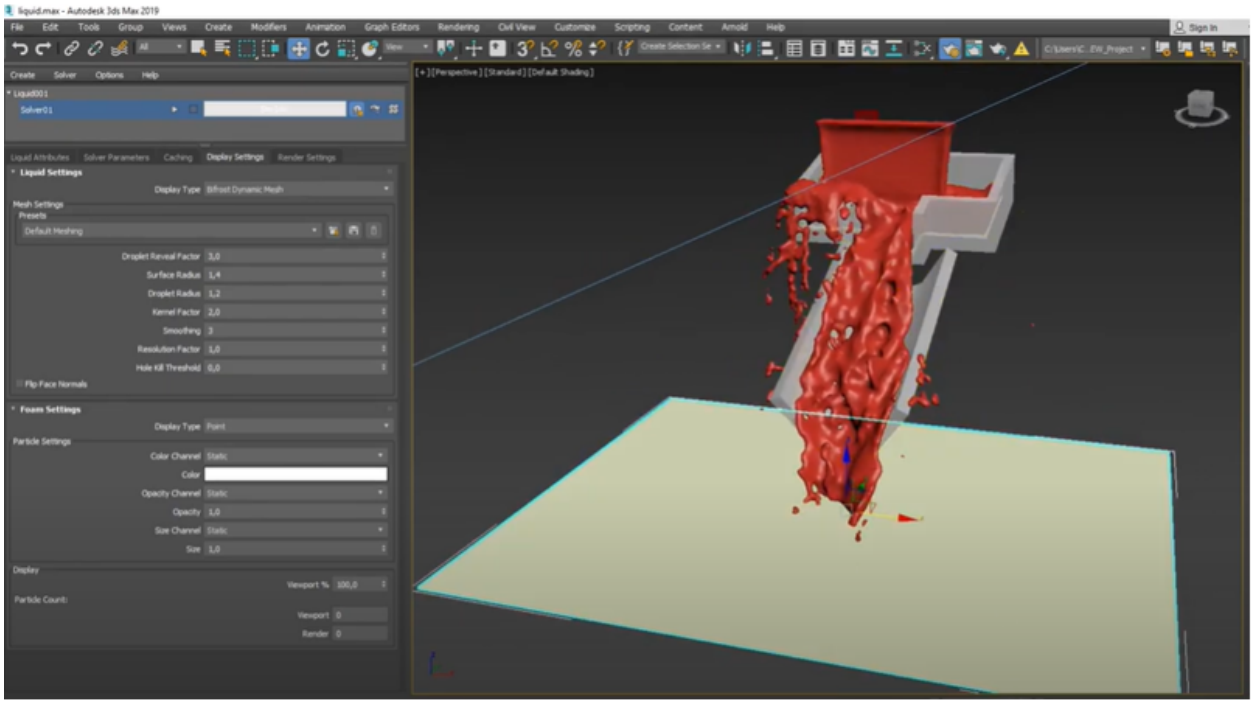


Рисунок 1.5 – пример реализации водопада в программе Autodesk 3ds Max

1.8.2 Blender

Blender – профессиональное свободное, кроссплатформенное и открытое программное обеспечение для создания трёхмерной компьютерной графики, включающее в себя средства моделирования, анимации, симуляции, рендеринга, монтажа видео со звуком [16].

С помощью данного ПО можно смоделировать водопад, причем он будет выглядеть достаточно реалистично (рисунок 1.6).

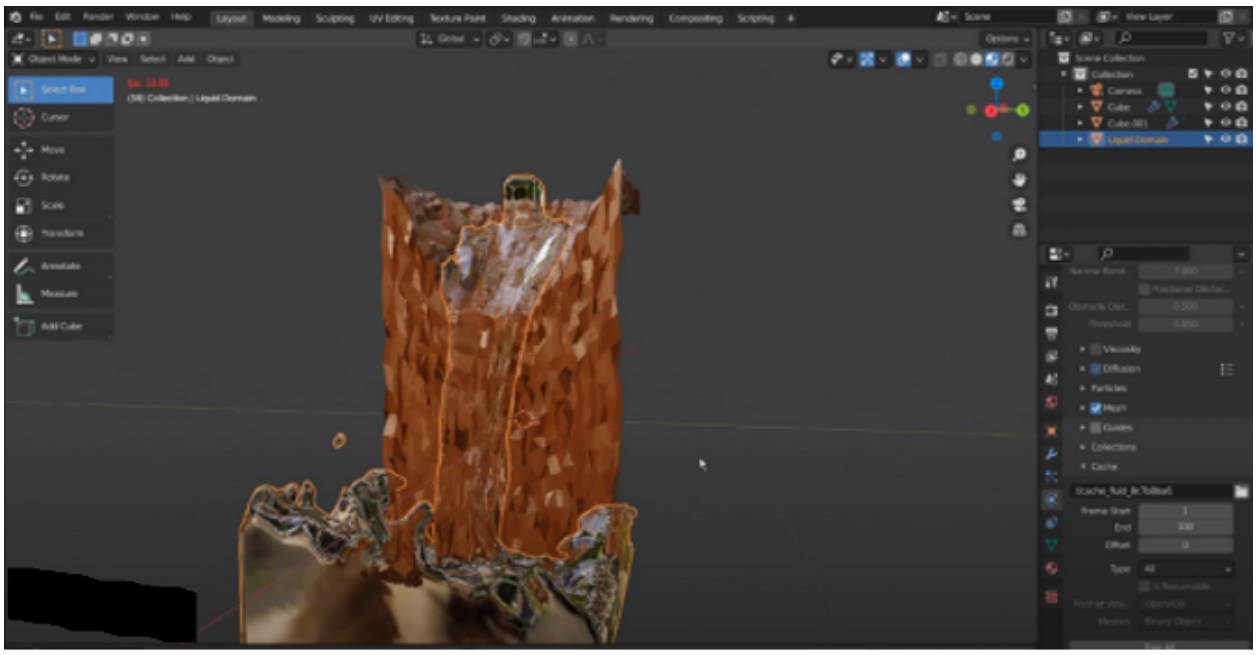


Рисунок 1.6 – пример реализации водопада в программе Blender

1.9 Вывод

В данном разделе были описаны и проанализированы методы визуализации текучей воды, с помощью которых можно смоделировать водопад, методы удаления невидимых линий и поверхностей, модели освещения, методы отрисовки изображения. В качестве алгоритма визуализации водопада выбран метод, основанный на системе частиц (уравнение движения частиц по криволинейной траектории), для удаления невидимых линий и поверхностей выбран алгоритм, использующий z-буфер, в качестве модели освещения выбрана модель Ламберта, в качестве API для отрисовки изображения – OpenGL. Также в данном разделе было рассмотрено существующее ПО, которое можно применять для моделирования водопада, и была формализована модель водопада.

2 Конструкторский раздел

2.1 Метод визуализации водопада, основанный на системе частиц

Алгоритм движения частиц

В качестве метода для моделирования основного потока водопада был выбран метод, основанный на уравнениях движения частиц по криволинейной траектории – рассмотрен в аналитическом разделе.

Моделирование водяного облака брызг, образующегося при ударе о воду

Так как при ударе воды о водяную поверхность образуется облако брызг, то его необходимо смоделировать для повышения реалистичности водопада. Полученная система частиц при реализации основного потока водопада позволяет довольно просто реализовать облако брызг. При достижении поверхности водоема, частица отражается (у нее изменяется направление движения, в векторе направления движения частицы координата z умножается на -1), теряет примерно 70% от своей скорости, и перекрашивается в белый цвет, чтобы отличаться от частиц основного потока водопада.

Моделирование водяных брызг от основного потока водопада

С помощью системы частиц довольно просто смоделировать частицы брызг, которые отходят от водопада. Чтобы показать, что частица стала брызгом, ее скорость уменьшается на 30% от текущей и изменяются значения вектора направления (с помощью умножения на определенные коэффициенты).

Каждая частица имеет определенное время жизни. Частицы, которые прошли стадии водопада и брызг должны быть удалены.

Входные данные:

- position — массив из 3 вещественных чисел (положение частицы в трехмерном пространстве – координаты x, y, z);
- speed — скорость частицы;
- acceleration — ускорение частицы;
- direction – массив из 3 вещественных чисел (направления движения частицы по осям 0x, 0y, 0z);
- gravitation – массив из 3 вещественных чисел (направление гравитации по осям 0x, 0y, 0z);
- color – цвет частицы, массив из 3 целых чисел (в формате RGB);
- max_time – максимальное время жизни частицы (при достижении максимального времени жизни частица должна исчезнуть с экрана);
- cur_time – текущее время жизни частицы.

Выходные данные:

- position (измененное положение частицы).

Схема алгоритма движения частиц водопада представлена на рисунках 2.1, 2.2.

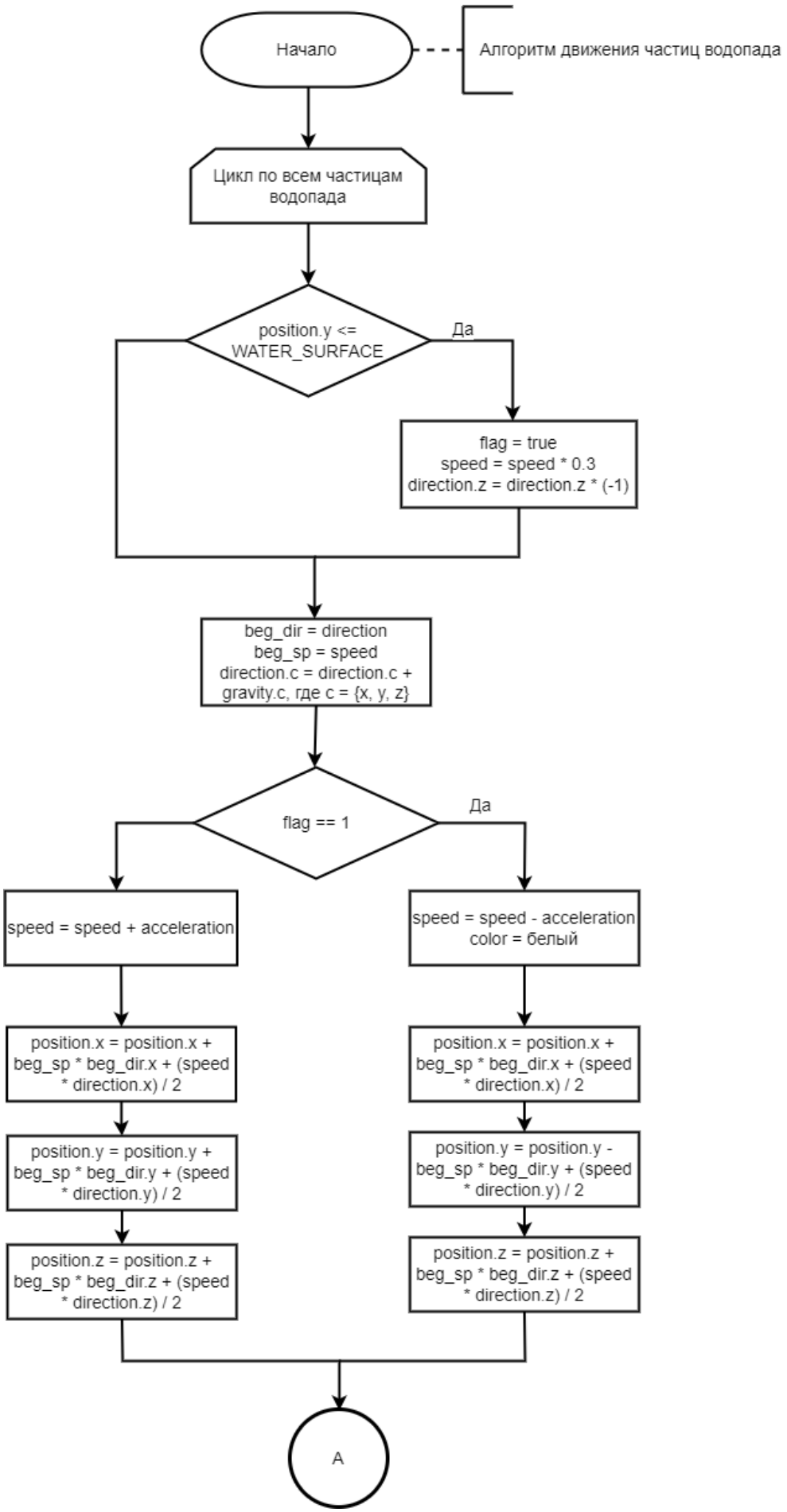


Рисунок 2.1 – Схема алгоритма движения частиц водопада

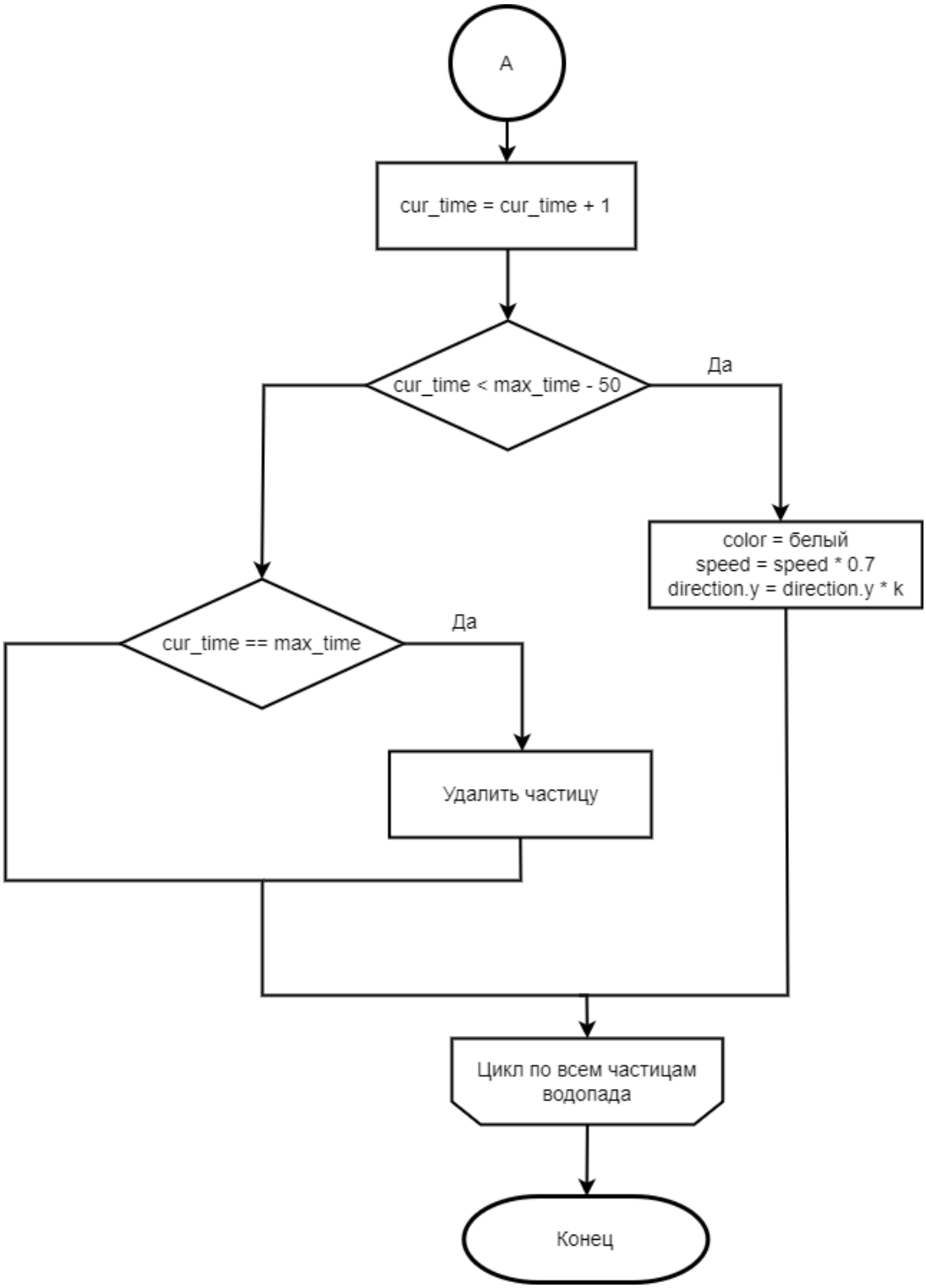


Рисунок 2.2 – Схема алгоритма движения частиц водопада

2.2 Алгоритм удаления невидимых ребер и поверхностей, использующий Z-буфер

Z-буфер – это отдельный буфер глубины, используемый для запоминания координаты z или глубины каждого видимого пиксела в пространстве изображения (размер z-буфера равен размеру экрана приложения).

Буфер кадра используется для запоминания атрибутов (цвета/интенсивности) каждого пиксела в пространстве изображения.

Вычислять глубину каждой частицы воды не придется, так как она будет вычислена в методе визуализации водопада (position.z).

Для вычисления глубины скалы можно использовать уравнения плоскостей, несущих грани скалы (параллелепипеда):

$$Ax + By + Cz + D = 0 \tag{2.1}$$

Если C = 0, то многоугольник находится параллельно оси z и линии взгляда наблюдателя, в этом случае изображается видимое ребро многоугольника.

Если ребро многоугольника не горизонтально, то z можно вычислить следующим образом:

$$\frac{z - z_1}{z_2 - z_1} = \frac{x - x_1}{x_2 - x_1} \tag{2.2}$$

Иначе, если ребро многоугольника не вертикально, то z можно вычислить следующим образом:

$$\frac{z - z_1}{z_2 - z_1} = \frac{y - y_1}{y_2 - y_1} \tag{2.3}$$

Иначе, произошло попадание в вершину многоугольника, глубину которой можно получить из исходных данных.

Вычисление глубины каждого пиксела на сканирующей строке ($y = const$) можно проделать пошаговым способом. Грань при этом рисуется последовательно, то есть строка за строкой.

$$\triangle z = z_1 - z = -\frac{Ax_1 + By + D}{C} + \frac{Ax + By + D}{C} = -\frac{A}{C} * \triangle x \tag{2.4}$$

В формуле выше $\triangle x = 1$ – шаг растра.

Общая схема алгоритма удаления невидимых линий и поверхностей, использующего z – буфер представлена на рисунке 2.3.

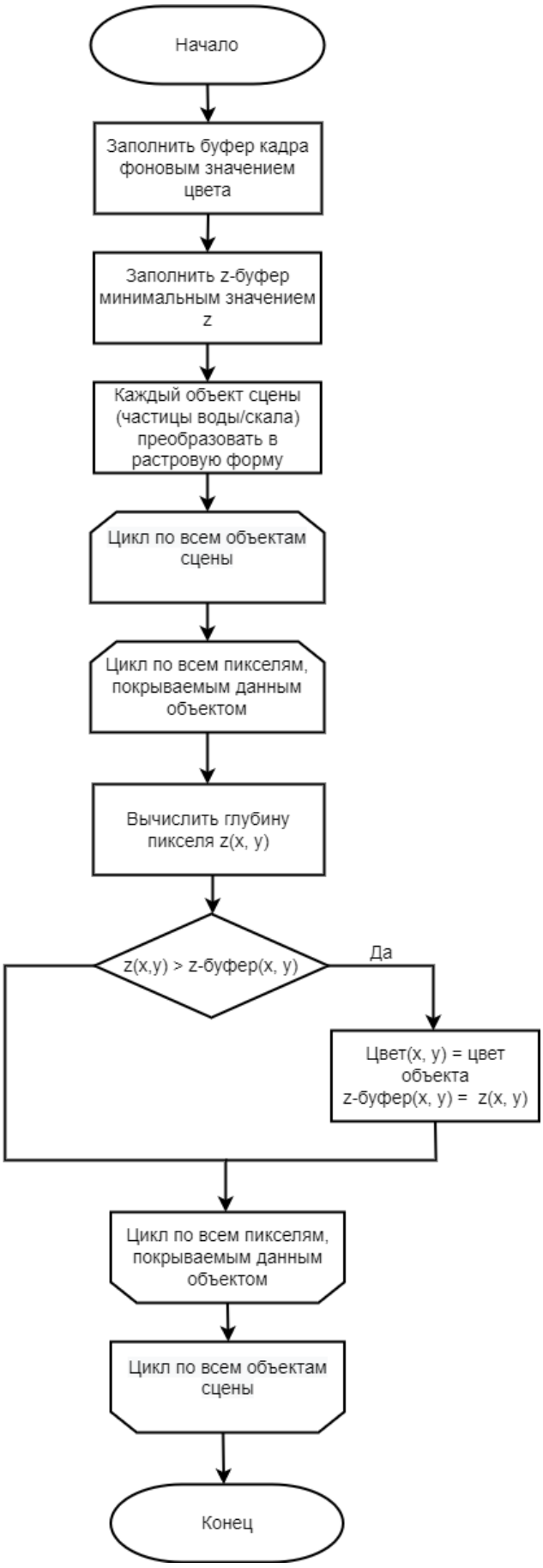


Рисунок 2.3 – Схема алгоритма удаления невидимых линий и поверхностей, использующего z-буфер

2.3 Модель освещения Ламберта

Данная модель вычисляет цвет поверхности в зависимости от того, как на нее светит источник света. Согласно данной модели, освещенность в точке вычисляется по следующей формуле:

$$I_d = k_d * i_d * cos(\vec{S}, \vec{n}), \tag{2.5}$$

где I_d – диффузная (рассеянная) составляющая света, k_d – коэффициент диффузной составляющей света (способность материала воспринимать диффузное освещение), i_d – интенсивность диффузного освещения, $\vec{S}$ – вектор падения света, $\vec{n}$ – вектор нормали в точке поверхности.

Изменение времени суток

При измении времени суток со дня на ночь интенсивность освещения сцены снижается на 30%.

2.4 Визуализация изображения скалы

Скалу удобно представить в виде массива точек, ее ограничивающих, и связей между ними.

Скала – прямоугольный параллелепипед (8 вершин). Каждая точка (вершина) задается тремя параметрами – координаты x, y, z.

Ширина и длина скалы задается константами, а высоту скалы может изменять пользователь.

2.5 Выбор используемых типов и структур данных

Для разрабатываемого программного обеспечения необходимо использовать следующие типы и структуры данных:

- точка (вершина) – массив из 3 вещественных чисел (float) – координаты x, y, z;
- параметры скалы (уступа водопада):
  - высота – вещественное число (float);
  - длина – вещественное число (float);
  - ширина – вещественное число (float);
- параметры водопада (потока воды):
  - скорость водопада – вещественное число (float);
  - количество частиц – целое число (int).
- источник света – естественный, то есть расположен в бесконечности (направление – массив из 3 вещественных чисел (координат) – константа, интенсивность – массив из 3 вещественных чисел – константа, но зависит от времени суток (день/ночь))
- частица – объект класса Particle (положение частицы (массив координат x, y, z – float), направление движения частицы (массив из направлений по осям 0x, 0y, 0z – float), скорость частицы (float), текущее и максимальное время жизни (int), цвет частицы (в формате RGB)
- водопад (поток воды) – массив частиц;
- скала (уступ водопада) – объект класса Rock (массив точек с координатами вершин – float и массив граней – связей (по индексам точек) – int, цвет (в формате RGB)).

2.6 Структура программы

Условное разбиение реализуемых классов на группы:

- трехмерные объекты:
  - Object — абстрактный класс.

  - Rock – скала (уступ водопада), реализует работу со скалой, трехмерные преобразования и генерацию.
  - Particle – частица водопада, реализует работу с частицей, ее движение, генерацию, хранит текущее положение частицы в пространстве, направление движения, скорость, текущее время жизни, максимальное время жизни, цвет (RGB).

- камера
  - Camera — класс для работы с камерой, хранит позицию камеры.

- сцена
  - SceneRender – класс для отрисовки всех объектов сцены.

- шейдер
  - Shader – класс инициализации шейдера.

- источник света
  - Light – класс, хранящий интенсивность и направление источника света.

Интерфейс – взаимодействие пользователя с интерфейсом производится через диалоговые окна приложения (модуль main), взаимодействующие с классом SceneRender.

2.7 Диаграмма классов

Диаграмма классов для данного программного обеспечения представлена на рисунке 2.4.

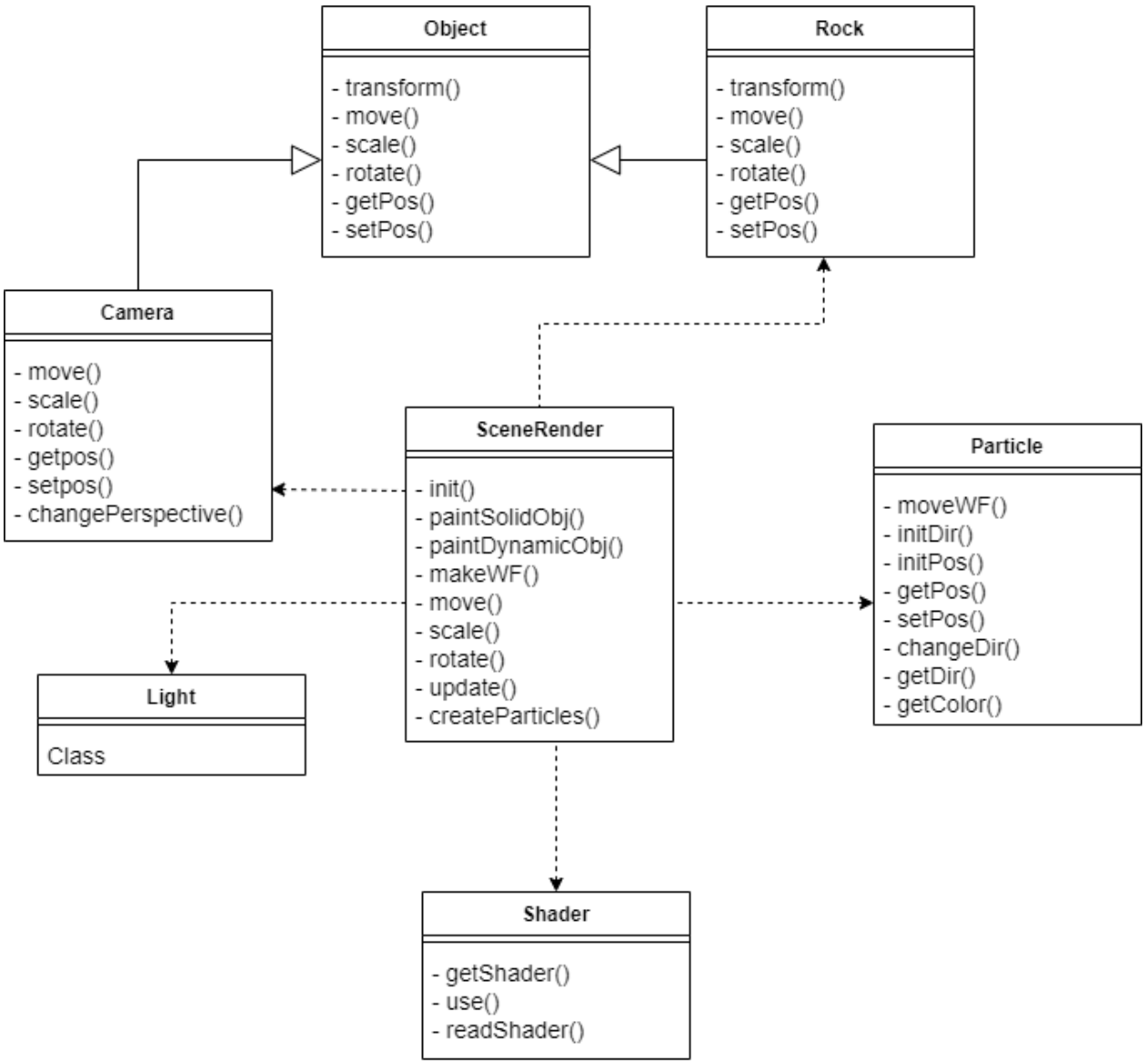


Рисунок 2.4 – Диаграмма классов

2.8 Вывод

В данном разделе были рассмотрены алгоритмы, которые будут использоваться для решения поставленной задачи, построены их схемы, также описаны используемые типы и структуры данных, описана структура основных классов программного обеспечения и построена их uml-диаграмма.

3 Технологический раздел

В данном разделе будут описаны требования к разрабатываемому ПО, обоснован выбор языка программирования и среды разработки, представлен интерфейс, предлагаемый пользователю.

3.1 Требования к программному обеспечению

Программа должна предоставлять пользователю следующий функционал:

- визуальное отображение сцены (модели водопада);
- изменение параметров водопада в интерактивном режиме – высоты водопада, скорости течения воды;
- изменение количества частиц воды в водопаде;
- зменение естественного источника света за счет выбора времени суток (день/ночь);
- перемещение, масштабирование, вращение сцены;
- перемещение, масштабирование, вращение камеры;
- вывод инструкции по использованию программы;
- запуск и остановка работы модели водопада.

Требования к работе программы:

- программа должна корректно реагировать на любые действия пользователя;
- время отклика программы на действия пользователя не должно превышать 1 секунды.

3.2 Выбор языков программирования и сред разработки

На данный момент существует очень большое количество языков и сред программирования, многие из которых обладают достаточно высокой эффективностью и удобны в использовании.

Для разработки данного программного обеспечения были выбраны языки *C++* [17] и *Python* [18].

Это обусловлено следующими факторами: C++ поддерживает объектно-ориентированное программирование; данный язык программирования является строго типизированным, что позволяет защититься от неконтролируемых ошибок; язык имеет богатую стандартную библиотеку, которая включает в себя распространённые контейнеры и алгоритмы, ввод-вывод, регулярные выражения, поддержку многопоточности; C++ имеет хорошую документацию.

Это обусловлено следующими факторами:

- *C++*
  - поддерживает объектно-ориентированное программирование;
  - является строго типизированным, что позволяет защититься от неконтролируемых ошибок;
  - имеет богатую стандартную библиотеку, которая включает в себя распространённые контейнеры и алгоритмы, ввод-вывод, регулярные выражения, поддержку многопоточности;
  - имеет хорошую документацию;
  - позволяет писать эффективные алгоритмы.
- *Python*
  - поддерживает объектно-ориентированное программирование;
  - предоставляет все необходимые графические библиотеки для решения поставленной задачи;
  - имеет хорошую документацию.

Язык C++ будет использоваться для написания алгоритма движения частиц водопада и алгоритма удаления невидимых линий и поверхностей, использующего z-буфер.

Язык Python будет использоваться для взаимодействия с графической библиотекой и для отрисовки сцены.

В качестве сред разработки были выбраны *Pycharm* [19] и Clion [20].

Преимущества данных сред разработки:

- высокая производительность, возможность эффективной работы над проектами любого размера и сложности;
- современные;
- поддерживают весь основной функционал: отладчик, точки останова, сборка, клиент для Git и т.д.;
- кроссплатформенные (работают под Linux, macOS и Windows).

В качестве среды для разработки интерфейса был выбран Qt Designer [21]. Он предоставляет возможность для создания качественного интерфейса и удобен в использовании, так как имеет встроенный редактор выводимого окна приложения.

3.3 Интерфейс программного обеспечения

Кнопки приложения и их функции:

- кнопка запуск/остановка – запускает или останавливает моделирование водопада, то есть течение потока воды;
- кнопки переключения день/ночь – задают текущее время суток;
- слайдер «высота водопада» – изменяет высоту водопада, чем правее слайдер, тем больше высота;
- слайдер «скорость течения воды» – изменяет скорость течения воды, чем правее слайдер, тем больше скорость;

- слайдер «количество частиц» – изменяет количество частиц воды в основном потоке воды, чем правее слайдер, тем больше частиц;
- стрелки вверх/вниз/влево/вправо и кнопки вперед/назад в блоке «камера/перемещение» – перемещают камеру;
- кнопки +/- в блоке «камера/масштабирование» – приближают/уменьшают изображение;
- стрелки вверх/вниз/влево/вправо/против часовой/по часовой – осуществляют поворот камеры;
- кнопка «управление сценой» – выводит инструкцию по управлению сценой;
- кнопка «выход» – осуществляет выход из программы.

Корректность ввода данных проверяется автоматически за счет средств, предоставляемых интерфейсом Qt Designer.

Также есть возможность взаимодействовать с программой при помощи клавиатуры (кнопки «1», «2», «3», «4», «5», «6», «W», «A», «S», «D», «F», «C», «H», «B») для поворотов, перемещения и масштабирования сцены и при помощи мыши (нажать на сцену и плавно ввести курсор вправо/влево для поворота камеры, для отключения режима поворота камеры нужно повторно нажать на сцену).

Интерфейс программы представлен на рисунках 3.1, 3.2.

Рисунок 3.1 – Интерфейс программы часть 1

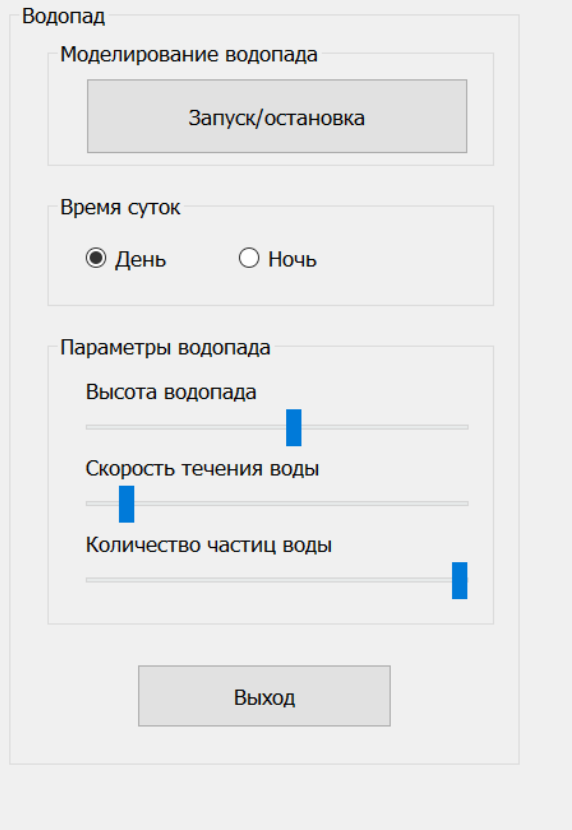


Рисунок 3.2 – Интерфейс программы часть 2

3.4 Примеры работы программного обеспечения

На рисунке 3.3 приведен результат генерации сцены в дневное время.

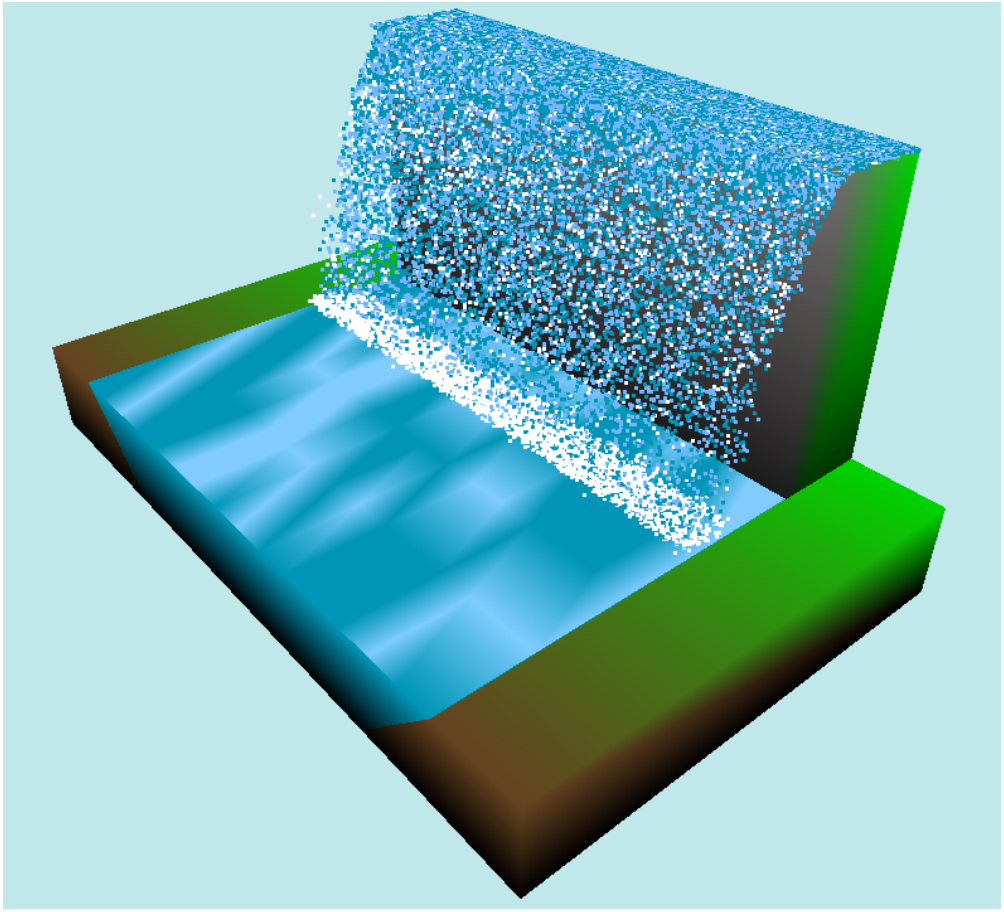


Рисунок 3.3 – Результат генерации сцены в дневное время

На рисунке 3.4 приведен результат генерации сцены в ночное время.

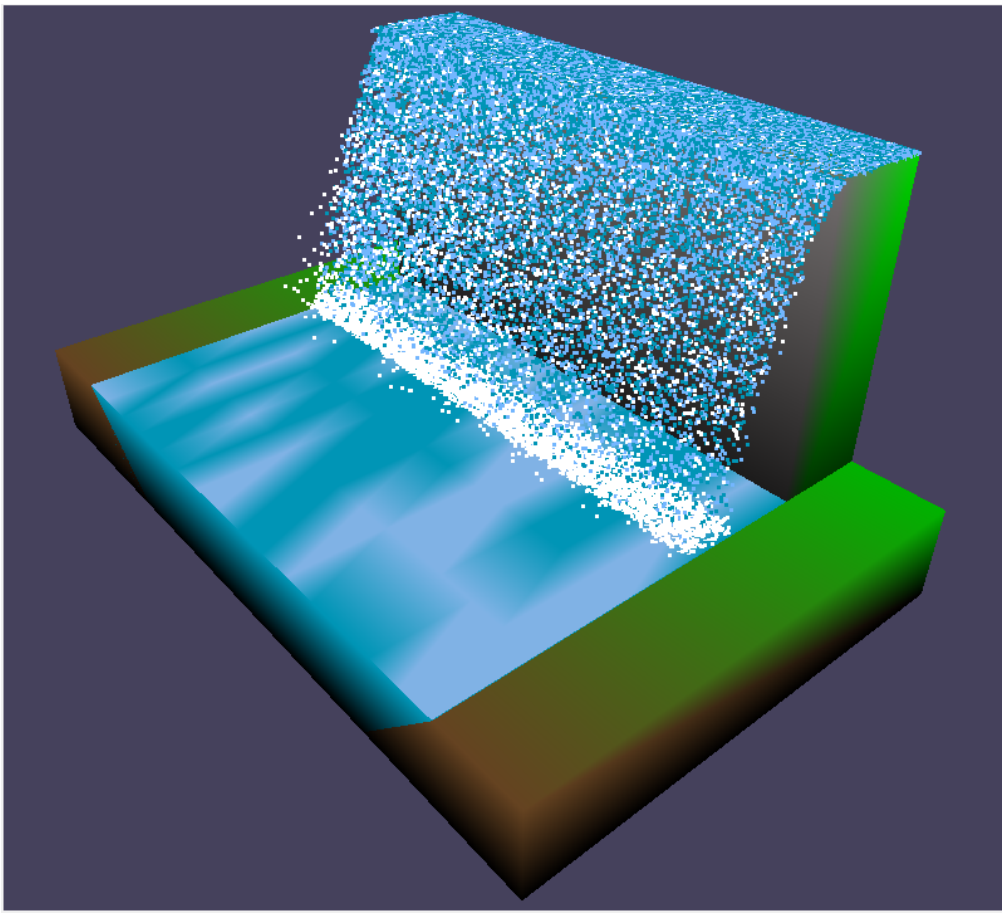


Рисунок 3.4 – Результат генерации сцены в ночное время

На рисунке 3.5 приведен результат генерации сцены с измененной высотой скалы.

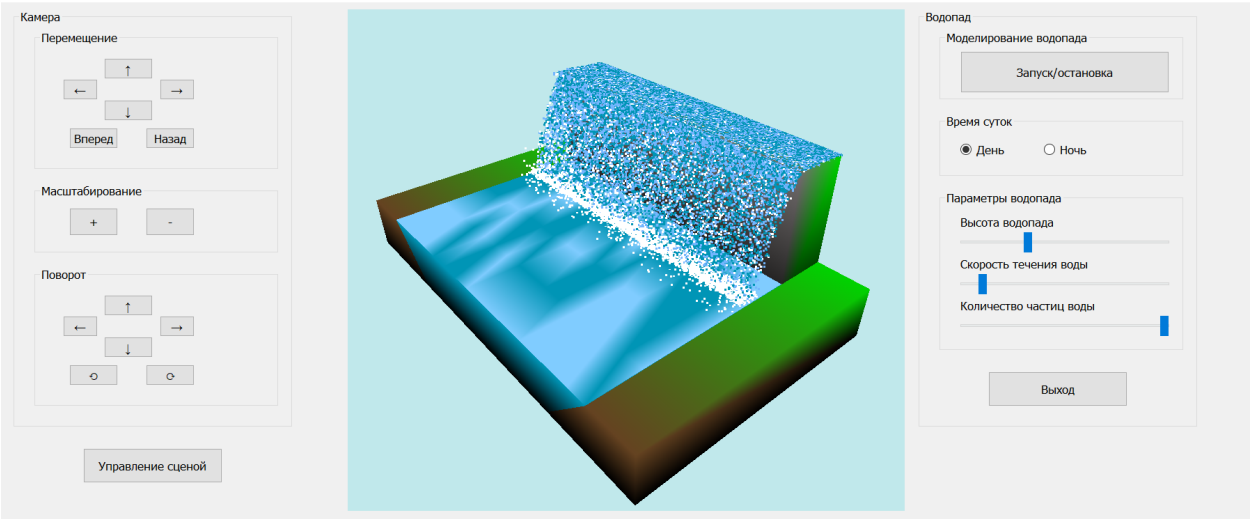


Рисунок 3.5 – Результат генерации сцены с измененной высотой скалы

На рисунке 3.6 приведен результат поворота сцены.

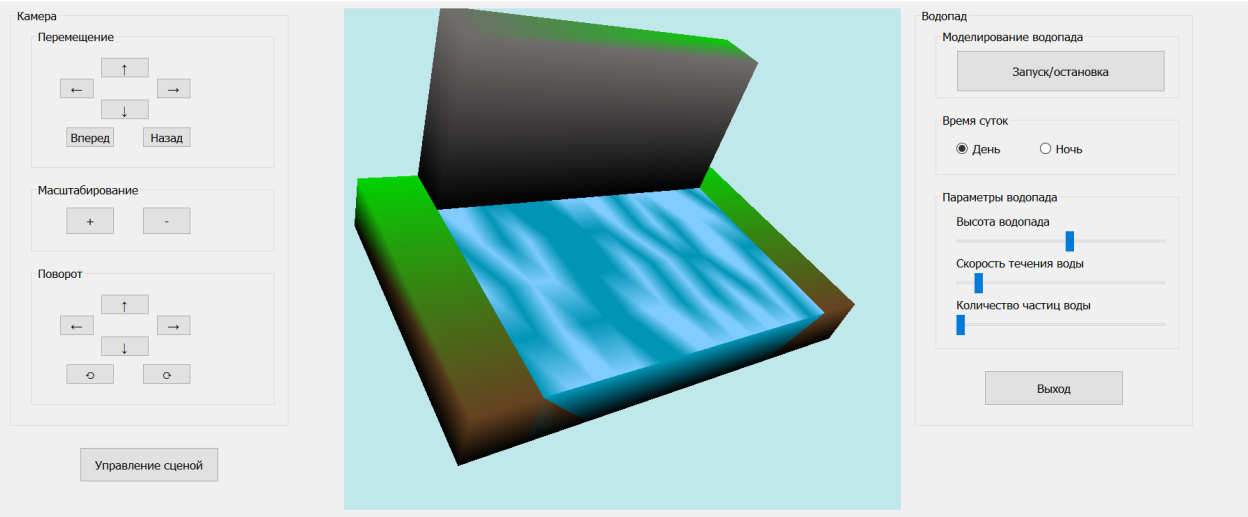


Рисунок 3.6 – Результат поворота сцены

3.5 Вывод

В данном разделе были описаны требования к разрабатываемому ПО, выбраны языки программирования и среды разработки, описан и представлен интерфейс приложения и приведены результаты работы програмы.

4 Экспериментальный раздел

В текущем разделе будут представлены примеры работы разработанного программного обеспечения, постановка эксперимента и сравнительный анализ реализованных алгоритмов.

4.1 Технические характеристики

Технические характеристики устройства, на котором выполнись замеры времени:

- операционная система: Windows 10 [22];
- оперативная память: 16 Гб;
- процессор: Intel® Core™ i5 10300H 2.5 ГГц [23];
- видеокарта: NVIDIA GeForce GTX 1650 Ti, 4 ГБ GDDR6 [24];
- 4 физических ядра, 4 логических ядра.

Во время проведения эксперимента ноутбук был включен в сеть питания и нагружен только встроенными приложениями окружения и системой тестирования.

4.2 Постановка эксперимента

4.2.1 Цель эксперимента

Цель эксперимента – проведение тестирования производительности при создании сцен с различной степенью нагрузки на программное обеспечение, полностью написанное на языке программирования *Python*, и программное обеспечение, алгоритм движения частиц воды водопада и алгоритм удаления невидимых линий и поверхностей которого написаны на

языке программирования *C++*, а взаимодействие с графической библиотекой и отрисовка сцены на *Python*. Нагрузка на ПО будет меняться в зависимости от количества частиц, из которых состоит водопад.

Производительность будет оцениваться мерой количества кадров в секунду (*Frames Per Second, FPS*), которое выдает разработанное ПО при данной загруженности.

4.2.2 Результаты эксперимента

Результаты эксперимента приведены в таблице 4.1.

Таблица 4.1 – Зависимость производительности ПО от количества частиц

	Производительность, кадров в секунду	
Количество частиц, штук	Python и C++	Python
500	71	52
1000	60	40
2000	43	35
3000	32	28
4000	27	24
5000	23	20
6000	19	17
7000	17	14
8000	15	12
9000	13	11
10000	12	9
15000	7	5
20000	6	3
25000	5	3
30000	4	2
35000	3	2
40000	3	2
50000	2	1
65000	2	1

На рисунке 4.1 представлен график сравнения изменения производительности ПО, написанных на *Python* и на *Python* и *C++*, в зависимости от количества частиц.

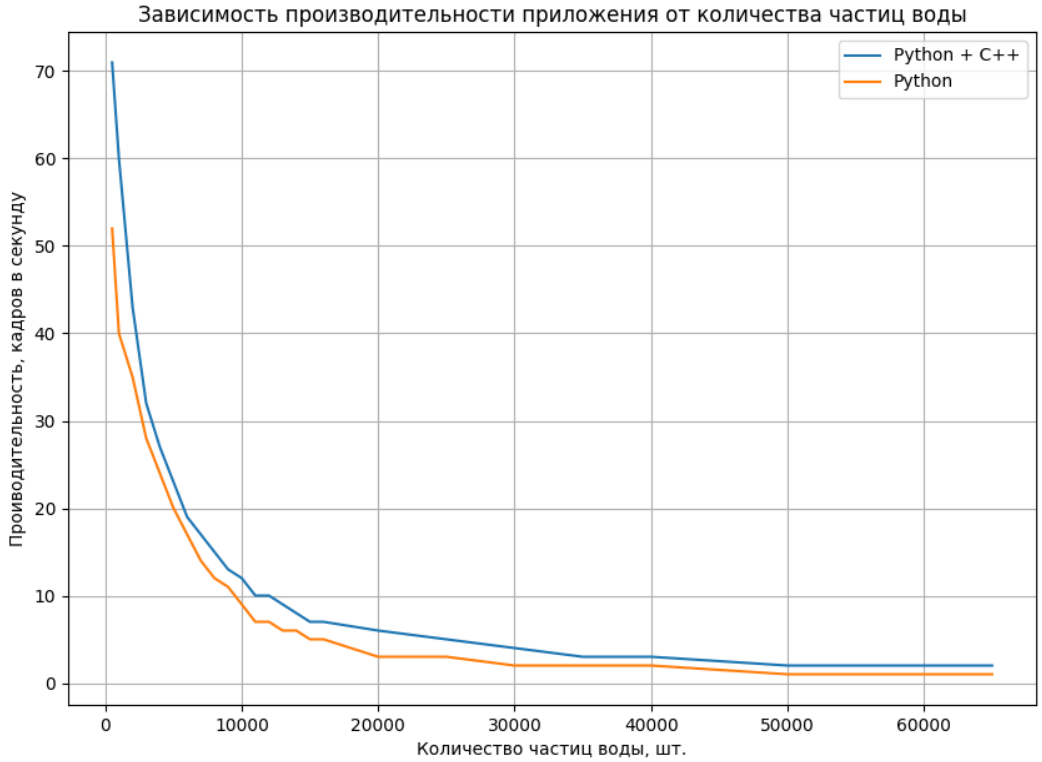


Рисунок 4.1 – График сравнения изменения производительности ПО, написанных на *Python* и на *Python* и *C++*, в зависимости от количества частиц

4.3 Вывод

По графику, полученному в результате эксперимента, видно, что производительность ПО (количество кадров в секунду) уменьшается экспоненциально при линейном увеличении количества частиц в водопаде. Также по результатам эксперимента получено, что ПО, написанное на *Python* с алгоритмами, написанными на *C++* и используемыми в коде на *Python* с помощью собранной динамической библиотеки, имеет более высокую производительность, чем ПО, полностью написанное на языке программирования *Python*.

Таким образом, задача отрисовки большого количества частиц является трудной задачей даже для компьютеров с достаточно хорошим современным программным обеспечением. Для комфортного восприятия динамического изображения человеку нужно хотя бы 10 кадров в секунду, что

достижимо при 10000 частиц в водопаде, но чтобы поток воды водопада был наиболее реалистичным и непрерывным нужно более 60000 частиц.

Заключение

В результате выполнения курсовой работы цель достигнута: разработано программное обеспечение, обеспечивающее динамическую визуализацию модели искусственного водопада.

В ходе выполнения данной работы были решены все задачи:

1) описана структура трехмерной сцены, включая объекты, из которых она состоит;

2) проанализированы существующие алгоритмы, которые можно использовать для моделирования водопада, выбран наиболее подходящий из них;

3) проанализированы алгоритмы удаления невидимых линий и поверхностей и выбран наиболее подходящий из них;

4) реализованы выбранные алгоритмы;

5) разработана структура классов программного обеспечения;

6) проведен эксперимент по замеру производительности программного обеспечения в зависимости от используемых языков программирования.

В результате провндения эксперимента было установлено, что производительность разработанного программного обеспечения уменьшается экспоненциально при линейном увеличении количества частиц, при этом программное обеспечение, написанное на языках программирования *Python* и *C++* имеет более высокую производительность, чем ПО, полностью написанное на *Python*.

Список использованных источников

[1] Куров А. В., Завалин А. А. Визуализация водной поверхности на основе спектрального метода [Электронный ресурс] // Символ науки. №1. 2017.

[2] Коптев А. В. Метод построения решений уравнений Навье-Стокса [Электронный ресурс] // Известия РГПУ им. А. И. Герцена. №154. 2013.

[3] Chentanez N., Muller M. Real-Time Eulerian Water Simulation Using a Restricted Tall Cell Grid [Электронный ресурс] // США, ACM Transactions on Graphics. 2011.

[4] Iwasaki K. Visual Simulation of Freezing Ice with Air Bubbles [Электронный ресурс] // Сингапур, SIGGRAPH Asia 2012 Technical Briefs. 2012.

[5] Busaryev O. Animating Bubble Interactions in a Liquid Foam [Электронный ресурс] // США, ACM Transactions on Graphics. 2012.

[6] Foster N., Fedkiw R. Practical Animation of Liquids [Электронный ресурс] // США, SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques. 2001.

[7] Сивухин Д.В. Общий курс физики. Физматлит, 2005. — Т. I. Механика. — С. 37. — 560 с.

[8] Chentanez N. Real-time Simulation of Large Bodies of Waterwith Small Scale Details [Электронный ресурс] // Мадрид, ACM SIGGRAPH Symposium on Computer Animation. 2010.

[9] Ihmsen M. ANIMATION OF AIR BUBBLES WITH SPH [Электронный ресурс] // Фрайбург, Computer Science Department – University of Freiburg. 2011.

[10] Роджерс Д. Алгоритмические основы машинной графики: пер. с англ. — М.: Мир, 1989.— 512 с.: ил.

[11] Vulkan Specification [Электронный ресурс]. Режим доступа: https://www.vulkan.org/learn#key-resources (дата обращения: 08.07.2022).

[12] The Industry's Foundation for High Performance Graphics [Электронный ресурс]. Режим доступа: https://www.khronos.org/opengl/ (дата обращения: 08.07.2022).

[13] Программирование для DirectX [Электронный ресурс]. Режим доступа: https://learn.microsoft.com/ru-ru/windows/uwp/gaming/directx-programming (дата обращения: 08.07.2022).

[14] Модели освещения. [Электронный ресурс]. Режим доступа: https://devburn.ru/2015/09/ (дата обращения: 06.07.2022).

[15] Документация по Autodesk 3ds Max [Электронный ресурс]. Режим доступа: https://ru.wikibooks.org/wiki/3ds_Max (дата обращения: 07.07.2022).

[16] Справочное руководство Blender 3.5 [Электронный ресурс]. Режим доступа: https://docs.blender.org/manual/ru/dev/ (дата обращения: 07.07.2022).

[17] Документация по C++ [Электронный ресурс]. Режим доступа: https://docs.microsoft.com/ru-ru/cpp/cpp/ (дата обращения: 13.07.2022).

[18] Лутц, Марк. Изучаем Python, том 1, 5-е изд. Пер. с англ. — СПб.: ООО "Диалектика", 2019 — С. 832.

[19] Узнайте все о PyCharm [Электронный ресурс]. Режим доступа: https://www.jetbrains.com/ru-ru/pycharm/learn/ (дата обращения: 20.09.2022).

[20] Документация по Clion [Электронный ресурс]. Режим доступа: https://www.jetbrains.com/help/clion/viewing-inline-documentation.html (дата обращения: 13.07.2022).

[21] Документация по Qt Designer [Электронный ресурс]. Режим доступа: https://doc.qt.io/qt-5/qtdesigner-manual.html (дата обращения: 13.07.2022).

[22] Windows 10 [Электронный ресурс]. Режим доступа: https://learn.microsoft.com/ru-ru/windows/ (дата обращения: 20.09.2022).

[23] Intel® Core™ i5-10300H Processor [Электронный ресурс]. Режим доступа: https://www.intel.co.uk/content/www/uk/en/products/sku/201839/intel-core-i510300h-processor-8m-cache-up-to-4-50-ghz/specifications.html (дата обращения: 20.09.2022).

[24] GeForce GTX 16 Series [Электронный ресурс]. Режим доступа: https://www.nvidia.com/en-in/geforce/graphics-cards/16-series/ (дата обращения: 20.09.2022).

Разработка программного обеспечения для моделирования водопада

Студент: Артюхин Николай Павлович ИУ7-51Б
Научный руководитель: Барышникова Марина Юрьевна

Москва, 2022 г.

Цель и задачи

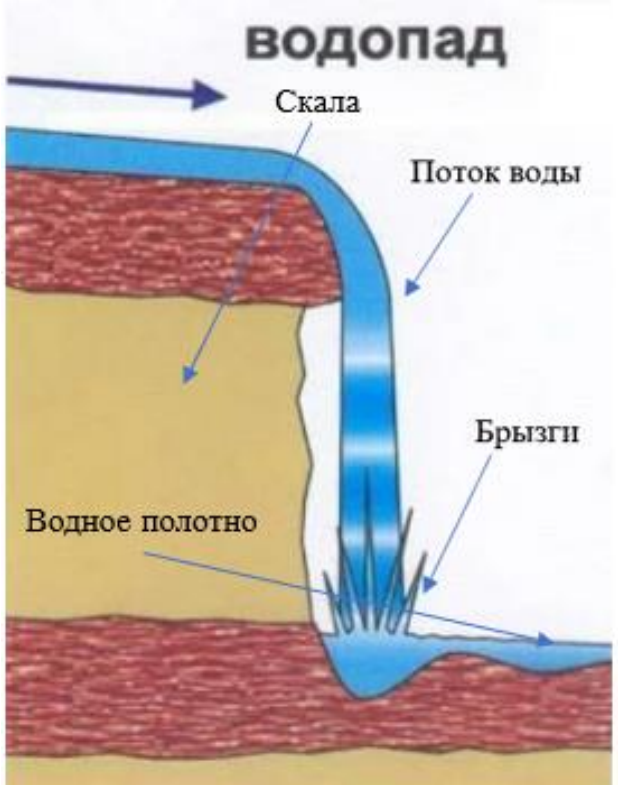
Цель работы – разработать программное обеспечение, обеспечивающее динамическую визуализацию модели искусственного водопада.

Задачи:

- описать структуру трехмерной сцены, включая объекты, из которых она состоит;
- проанализировать существующие алгоритмы, которые можно использовать для моделирования водопада, выбрать наиболее подходящий из них;
- проанализировать алгоритмы удаления невидимых линий и поверхностей и выбрать наиболее подходящий из них;
- реализовать выбранные алгоритмы;
- разработать структуру классов программного обеспечения;
- провести эксперимент по замеру производительности программного обеспечения в зависимости от используемых языков программирования.

Описание объектов сцены

- Скала (уступ водопада)
- Поток воды
- Брызги
- Водное полотно
- Берега
- Камера

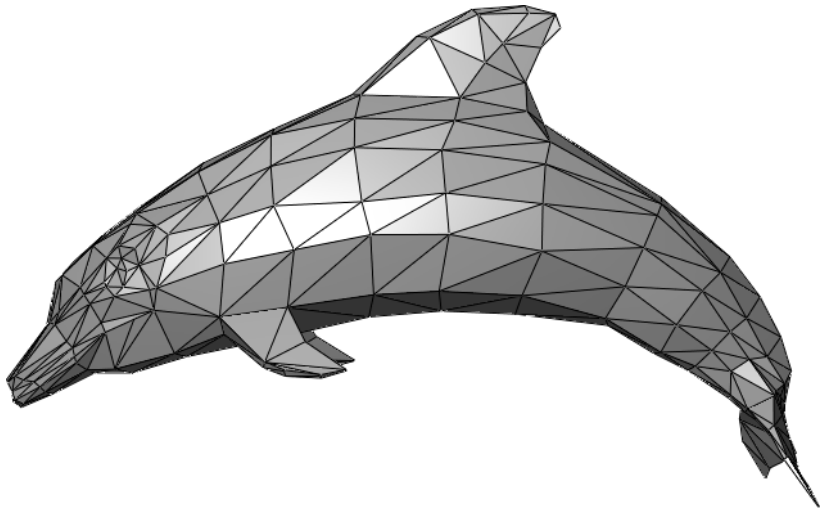


Анализ методов моделирования водопада

	Уравнение Навье-Стокса	Сеточные методы	**Система частиц**	Комбинированные методы
Эффективность по времени	Низкая	Высокая	**Высокая**	Высокая
Реалистичность	Высокая	Низкая	**Средняя**	Высокая
Сложность реализации	Высокая	Низкая	**Низкая**	Высокая
Возможность реализации брызг	Нет	Нет	**Да**	Да

Способ представления трехмерной модели

- В рамках данного проекта в качестве представления модели была выбрана полигональная сетка.
- **Полигональная сетка -** это совокупность вершин, рёбер и граней, которые определяют форму многогранного объекта в трехмерной компьютерной графике и объемном моделировании.



Анализ алгоритмов удаления невидимых линий и поверхностей

	Алгоритм Z-буфера	Алгоритм Робертса	Алгоритм художника	Алгоритм Варнока	Алгоритм обратной трассировки лучей
Сложность N – число граней, m – число пикселей	**O(mN)**	O(N^2)	O(mN)	O(mN)	O(mN)
Реалистичность	**Средняя**	Средняя	Низкая	Средняя	Высокая
Сложность реализации	**Низкая**	Высокая	Высокая	Средняя	Высокая
Эффективность по памяти	**Низкая**	Средняя	Высокая	Средняя	Высокая
Эффективность по времени	**Высокая**	Средняя	Низкая	Средняя	Низкая

Выбор API для отрисовки изображения

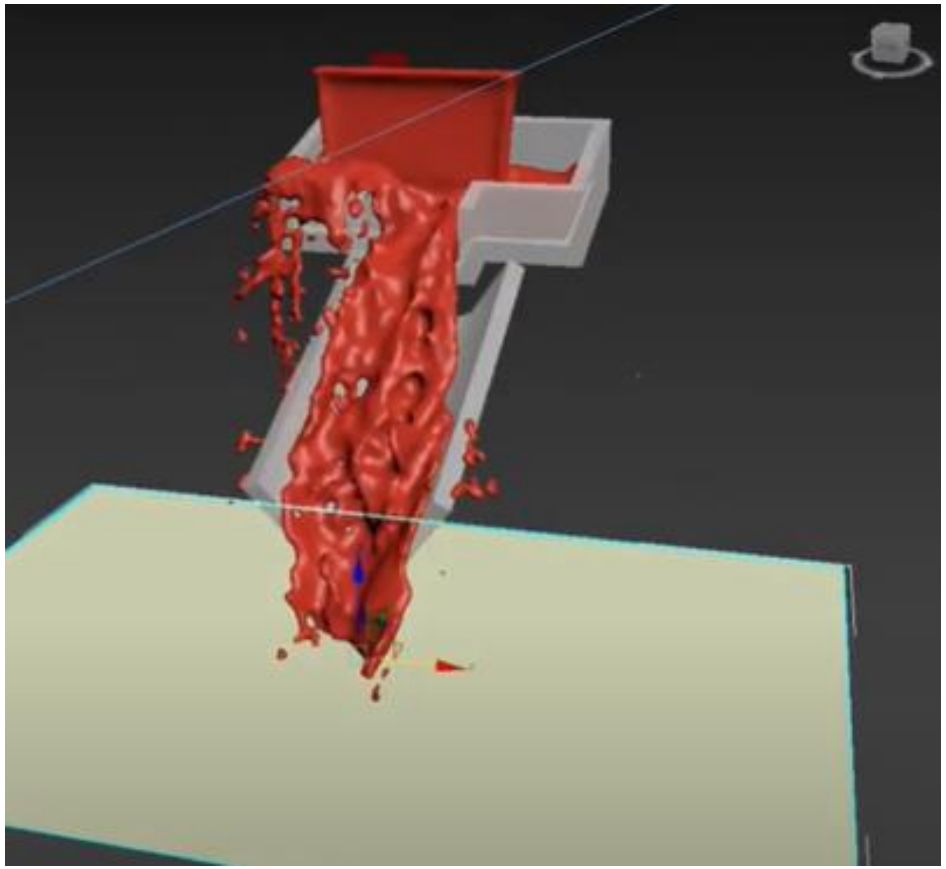
	Vulkan	**OpenGL**	DirectX
Кроссплатформенность	Да	**Да**	Нет (только Windows)
Открытый код	Да	**Да**	Нет
Библиотека для ЯП Python	Нет	**Да**	Да
Поддержка видеокарт предыдущего поколения	Да	**Да**	Нет

Выбор модели освещения

	Модель Ламберта	Модель Фонга
Реалистичность	**Средняя**	Высокая
Сложность реализации	**Низкая**	Высокая

Существующее ПО

Autodesk 3ds Max



Blender

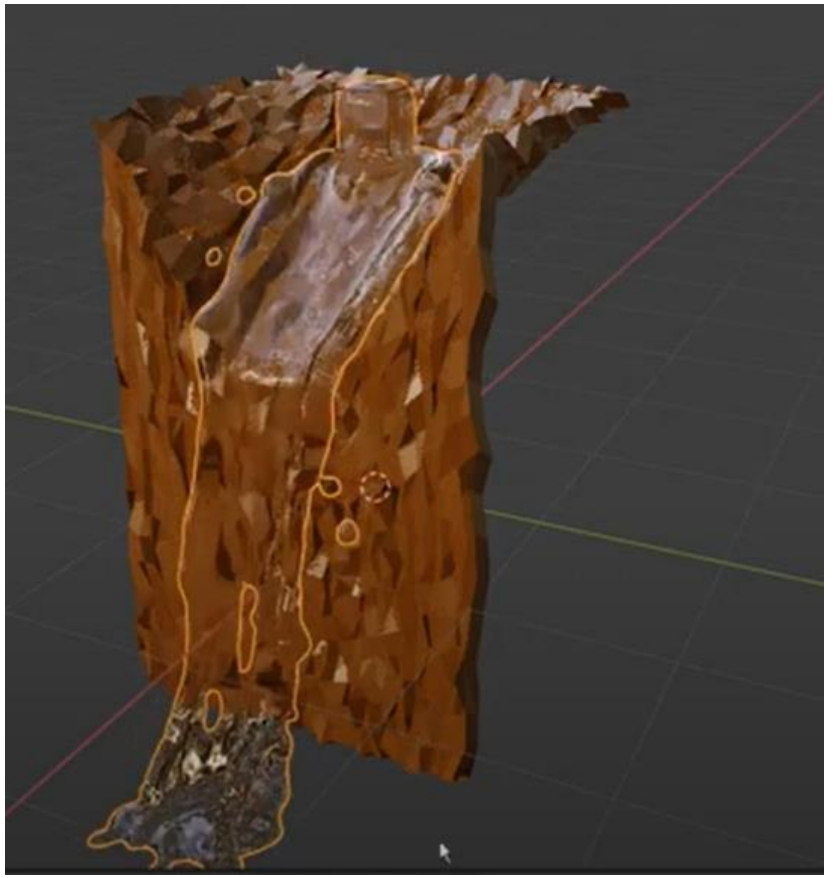


Схема алгоритма движения частиц водопада

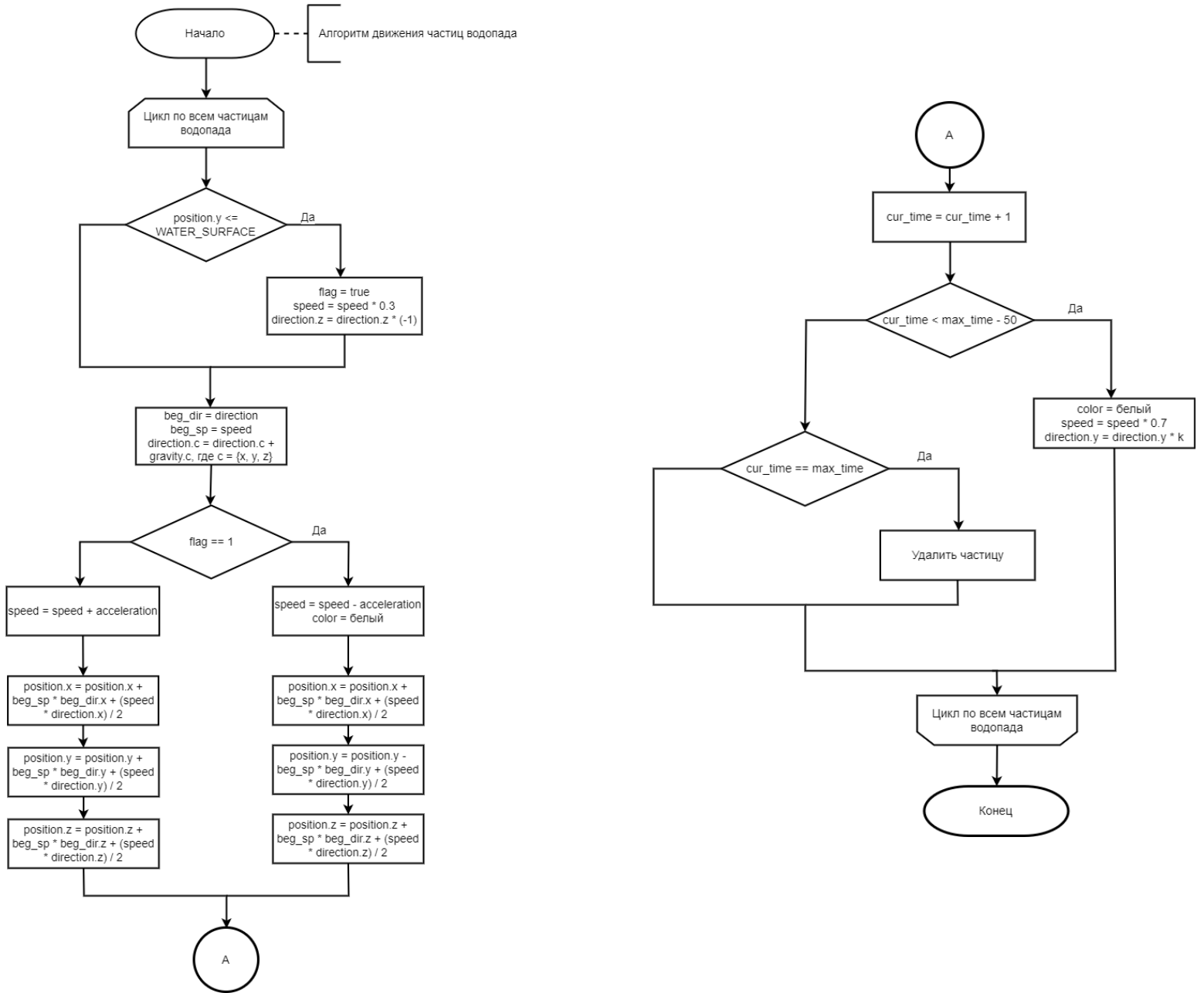


Схема алгоритма z-буфера

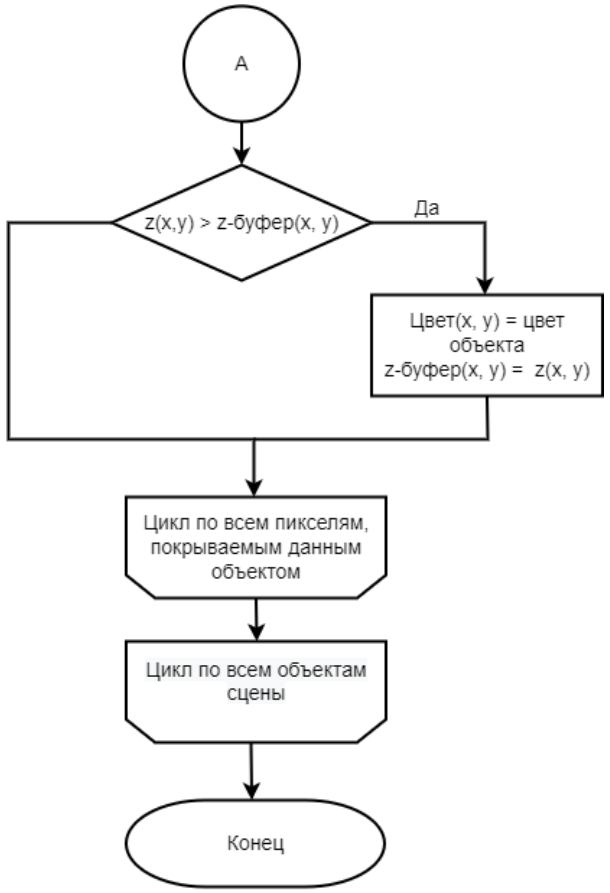
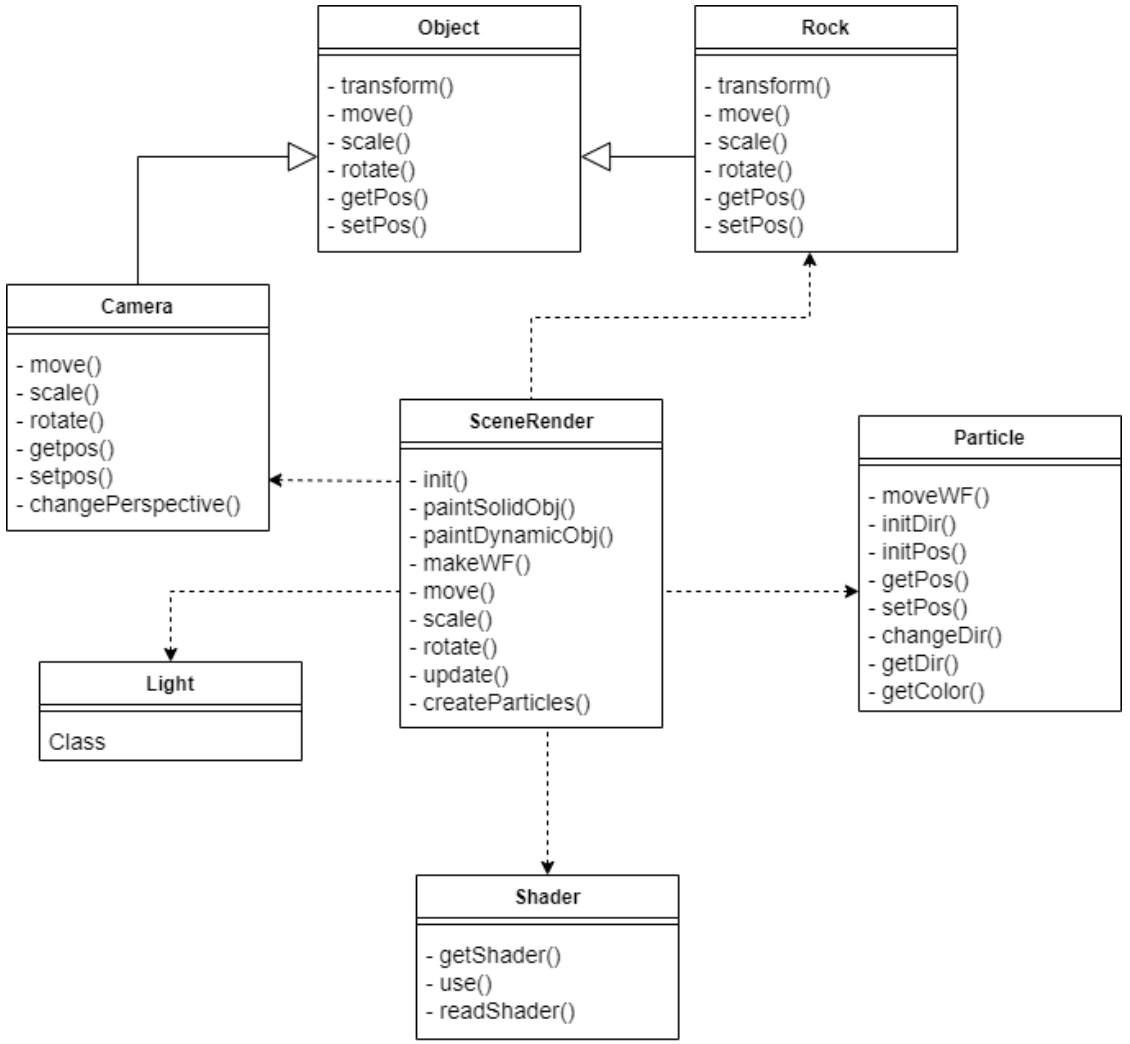
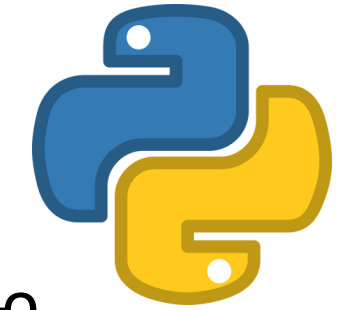


Диаграмма классов



Выбор языков программирования и сред разработки



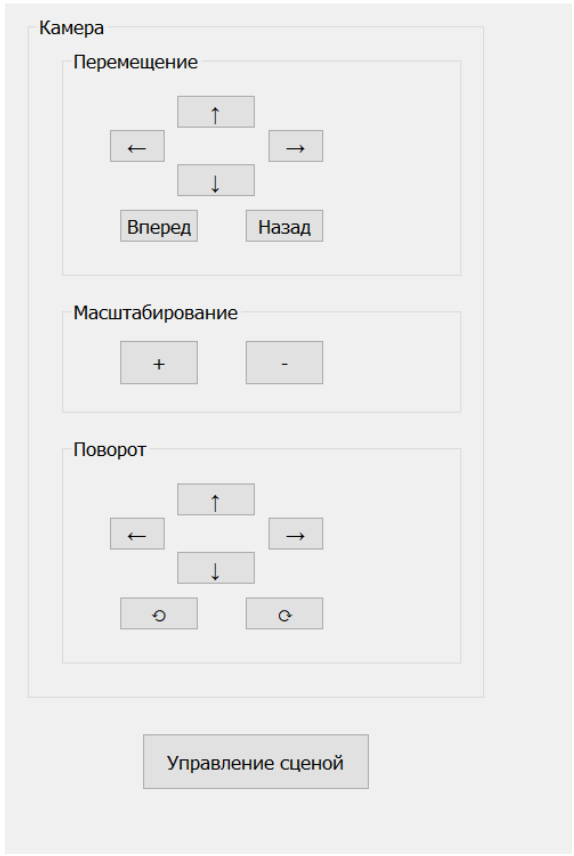
- Языки программирования: Python, C++
- C++ поддерживает объектно-ориентированное программирование, имеет хорошую документацию, строго типизированный (защита от неконтролируемых ошибок).
- Python поддерживает объектно-ориентированное программирование, предоставляет все необходимые графические библиотеки для решения поставленной задачи, имеет хорошую документацию.
- Разработка интерфейса: Qt Designer
- Среды разработки: PyCharm, Clion

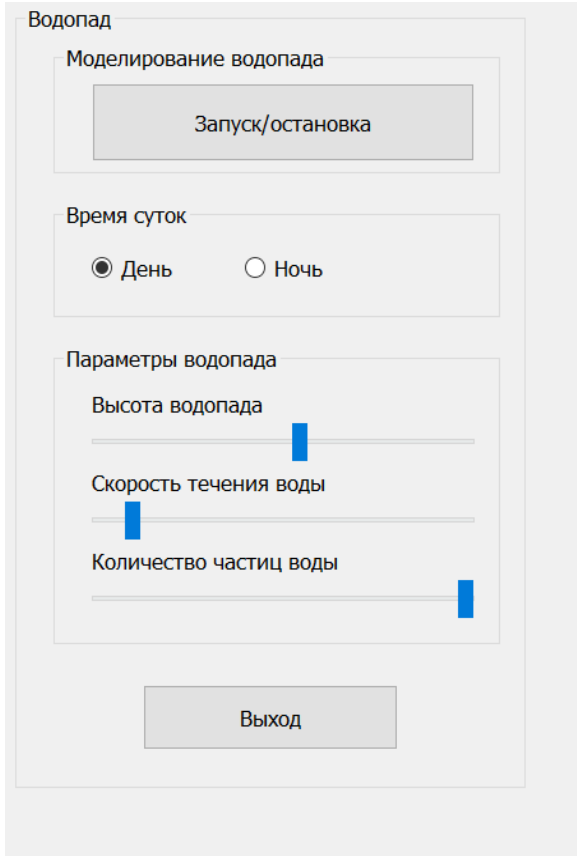
Интерфейс программы

Пользователь может:

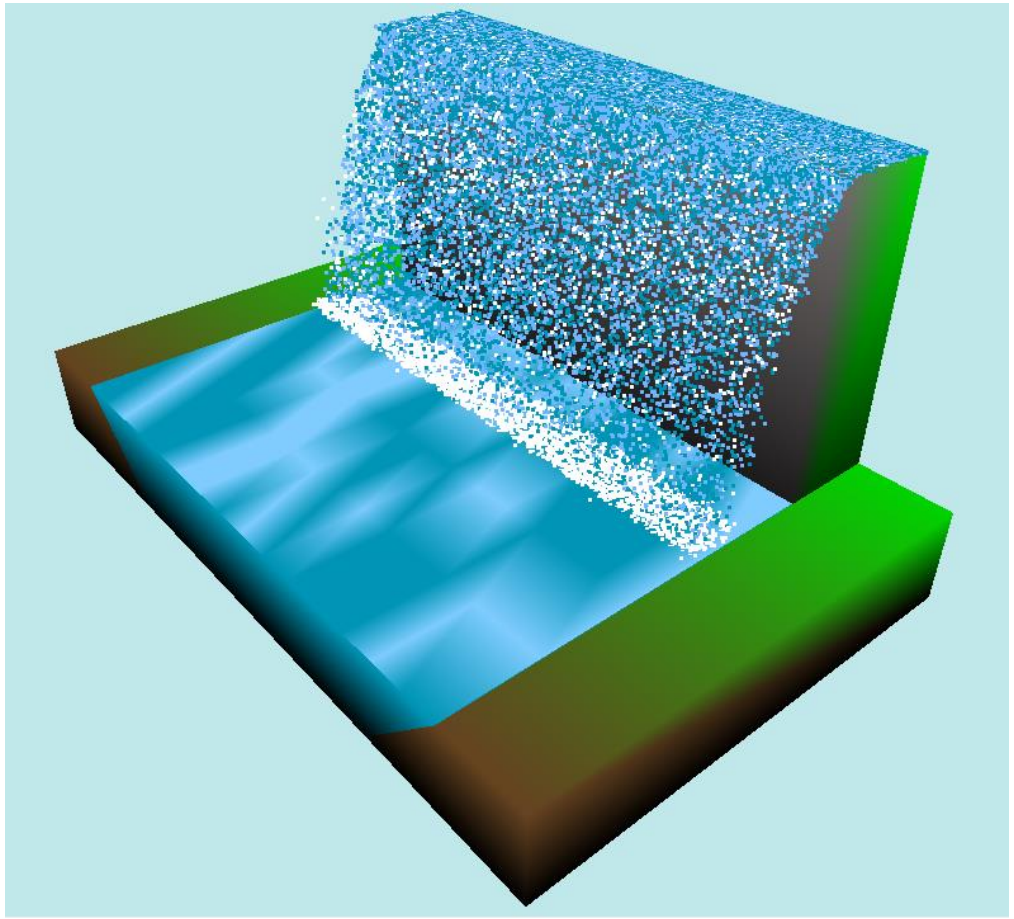
- запустить/остановить моделирование водопада;
- выбрать дневное или ночное освещение;
- изменить высоту водопада;
- изменить скорость течения воды;
- изменить количество частиц воды;
- управлять камерой/сценой.

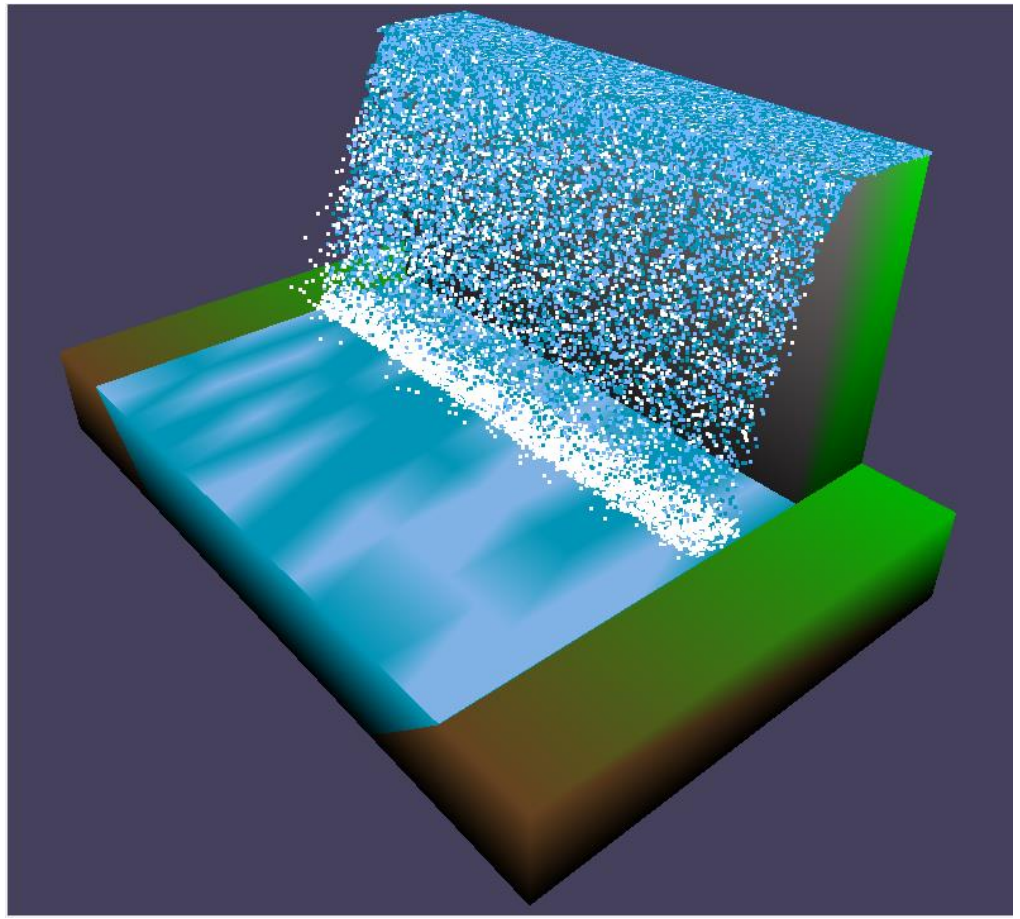
Корректность ввода данных проверяется автоматически за счет средств, предоставляемых интерфейсом Qt Designer.





Примеры работы программы





Эксперимент

- **Цель эксперимента** – проведение тестирования производительности при создании сцен с различной степенью нагрузки на программное обеспечение, полностью написанное на языке программирования Python, и программное обеспечение, написанное на Python и C++.

Нагрузка на ПО будет меняться в зависимости от количества частиц, из которых состоит водопад.

- **Результаты эксперимента**

Справа представлена таблица зависимости производительности ПО, написанного полностью на Python, и ПО, написанного на Python (графика) и C++ (алгоритм движения частиц, алгоритм z-буфера), в зависимости от количества частиц в водопаде.

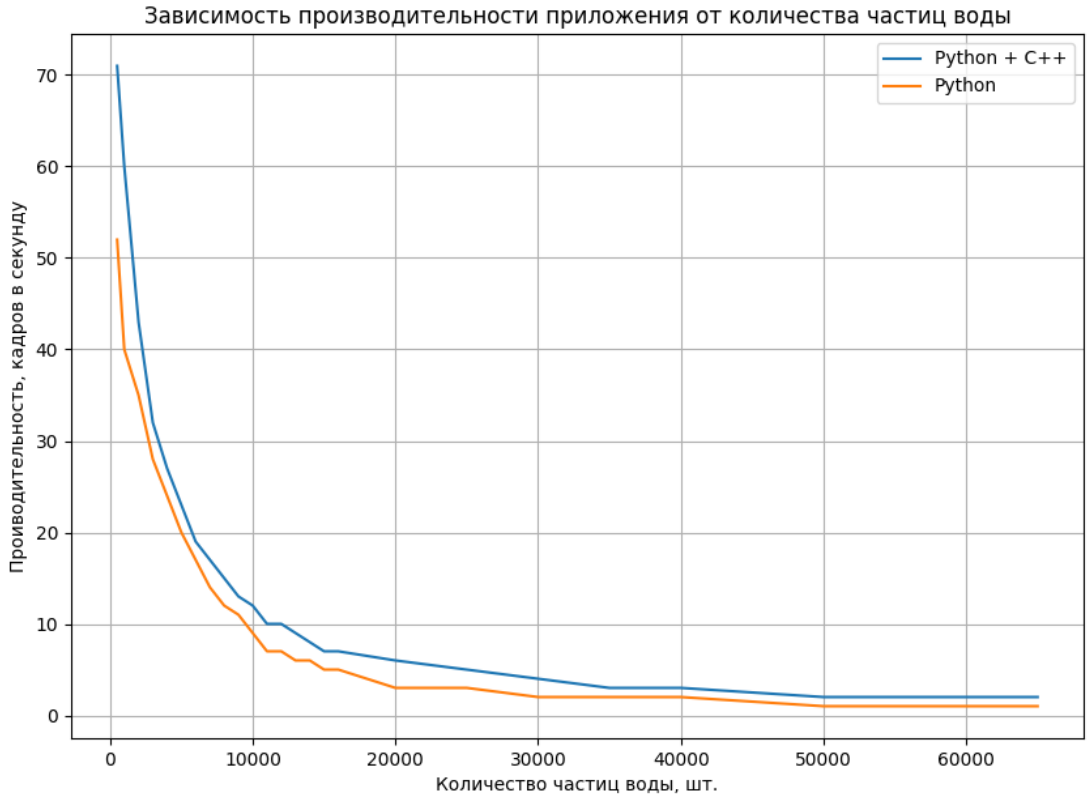
	Производительность, кадров в секунду	
Количество частиц, штук	Python и C++	Python
500	71	52
1000	60	40
2000	43	35
3000	32	28
4000	27	24
5000	23	20
6000	19	17
7000	17	14
8000	15	12
9000	13	11
10000	12	9
15000	7	5
20000	6	3
25000	5	3
30000	4	2
35000	3	2
40000	3	2
50000	2	1
65000	2	1

Результаты эксперимента

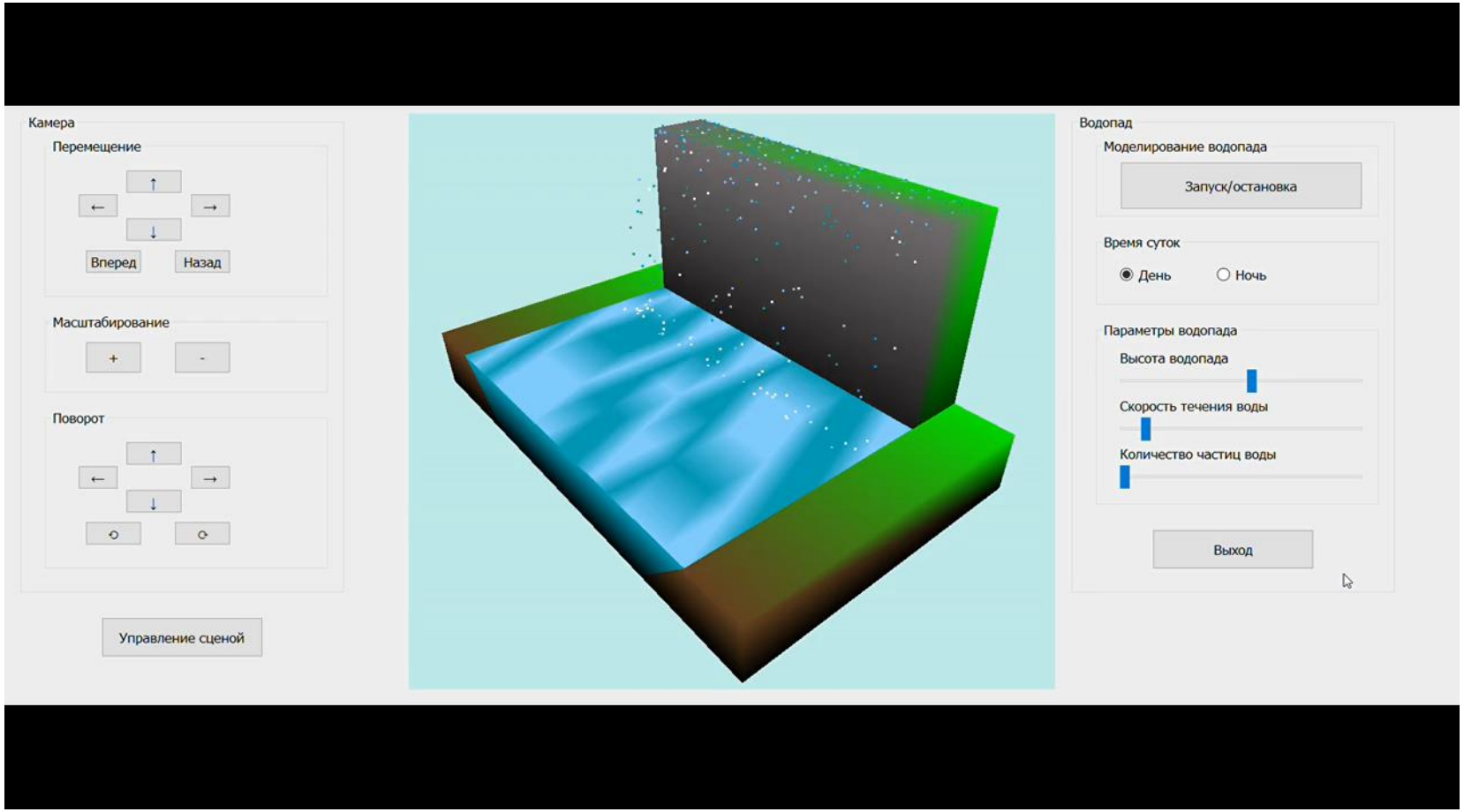
- Справа представлен график зависимости производительности ПО, написанного полностью на Python, и ПО, написанного на Python (графика) и C++ (алгоритм движения частиц, алгоритм z-буфера), в зависимости от количества частиц в водопаде.

- По графику, полученному в результате эксперимента, видно, что производительность ПО (количество кадров в секунду) уменьшается экспоненциально при линейном увеличении количества частиц в водопаде.

- Также можно увидеть, что ПО, написанное на Python и C++ показало более высокую производительность, чем ПО, полностью написанное на Python.

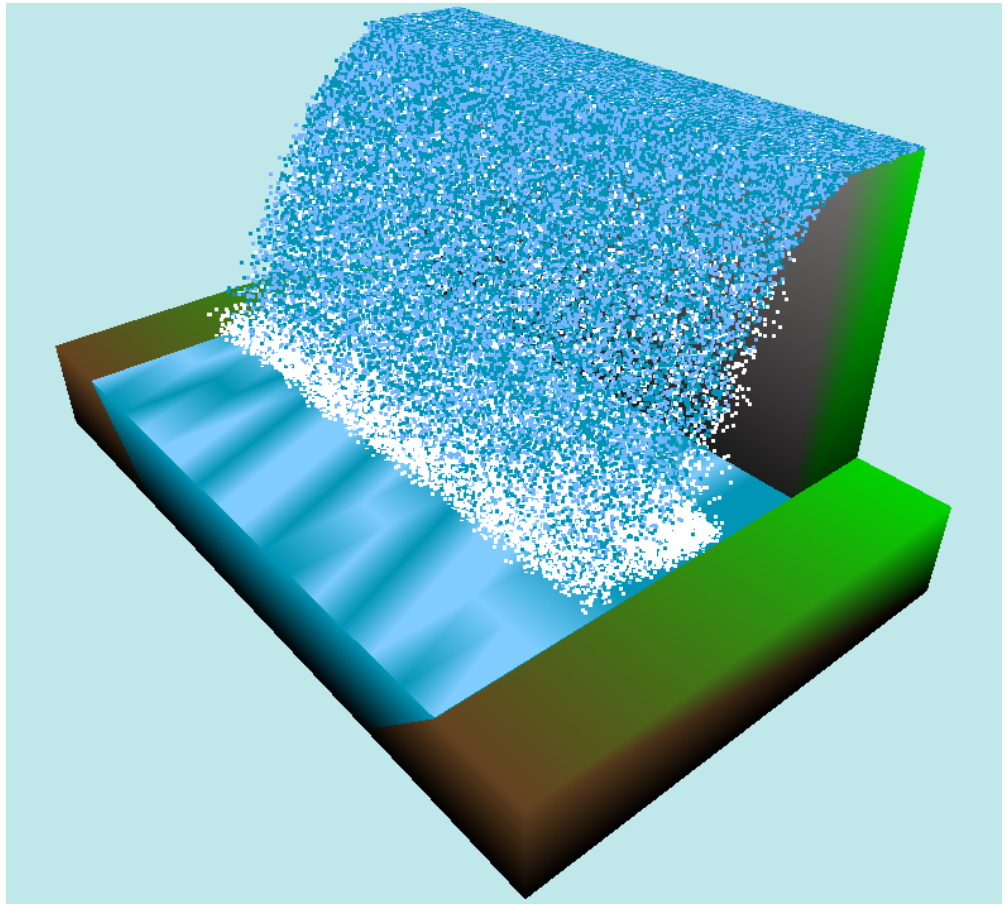


Реалистичность водопада, в зависимости от числа частиц



Реалистичность водопада, в зависимости от числа частиц

- На рисунке представлен водопад из 200000 частиц.
- Видно, что вода стала почти непрерывной.
- При этом за реалистичность изображения приходится платить производительностью приложения.



Заключение

Цель курсовой работы была достигнута: разработано программное обеспечение, обеспечивающее динамическую визуализацию модели искусственного водопада.

Все поставленные **задачи** были решены:

- описана структура трехмерной сцены;
- проанализированы алгоритмы необходимые для моделирования водопада;
- реализованы выбранные алгоритмы;
- разработана структура классов ПО;
- проведен эксперимент по замеру производительности программного обеспечения в зависимости от используемых языков программирования.