СОДЕРЖАНИЕ

1 Аналитическая часть

1.1 Алгоритмы удаления невидимых поверхностей или линий

Алгоритм Робертса работает в объектном пространстве. В первую очередь он удаляет те грани и линии, которые экранируются самим объектом. После чего, оставшиеся грани выпуклого многранника сравниваются с оставщимися гранями другого выпуклого многогранника. Если же многогранник не является выпуклым, то его придется разбивать на выпуклые, что приводит к дополнительным затратам.

Метод Z-Буффера работает в пространстве изображений. В нем используется буффер для запоминания глубины каждого видимого пикселя. При обработке нового пикселя его глубина сравнивается с глубиной, записанной в буффере. Основной недостаток – потребление большого количества памяти.

В алгоритме, использующем трассировку лучей, из положения камеры испускается луч через каждый пиксель изображения. Каждый луч проверяется на пересечение с объектом 3D сцены. В точке пересечения определяется освещенность поверхности по параметрам. Для рассчета теней испускаются тестовые лучи из точки пересечения к источникам света, если тестовый луч блокируют другие объекты сцены, то точка затеняется по выбранным алгоритмам.

1.2 Отражение

Прямое освещение объекта равномерным количеством света, взаимодействующего с поверхностью. После того как свет падает на объект, он отражается в зависимости от свойств поверхности объекта, а также угла падения света. Свет, падающий перпендикулярно, создает более яркое освещение (Это явление описывается законом Ламберта). Матовые поверхности (дерево, камень, штукатурка) отражают больше диффузионного света, чем глянцевые, в результате чего они выглядят более мягкими. При

таком отражении положение наблюдателя не имеет значения, так как диффузно отраженный свет рассеивается равномерно по всем направлениям.

Окружающий свет не имеет направления. Его интенсивность определяется войствами материалов поверхностей объектов, а именно их коэффициентами отражения окружающего света. Так же окружающий свет не отбрасывает тени и не имеет какого-то конкретного источника. Окружающий свет действует на поверхности объектов независимо от их ориентации. Он добавляет базовый уровень освещенности, чтобы избежать полной темноты на теневых участках. Окружающий свет не требует сложных вычислений, таких как рассчет направления света.

Зеркальное отражение создает яркие пятна на объектах, исходя из интенсивности коэффициента зеркального отражения поверхности, а так же угла падения. Каждый материал имеет свой коэффициент отражения.

1.3 Модель освещения

Модели освещения используются для визуализации световых эффектов, где все рассчеты воспроизводятся на основе законов физики. Главной целью использования модели освещения является вычисления цвета каждого пикселя на основе освещенности объекта, представляемого в этом пикселе.

Локальные модели освещения учитывают световые эффекты, происходящие на объекте, исходя из источников света, находящихся рядом. В таких моделях освещённость пикселей определяется в зависимости от угла падения света, свойств поверхности и её взаимодействия с различными типами отражения (диффузного, зеркального, окружающего). Локальные модели упрощают расчёты, так как не учитывают влияние отражённых лучей от других объектов. На рисунке **??** представлено изображение, построенное с использованием локальной моделью освещения.

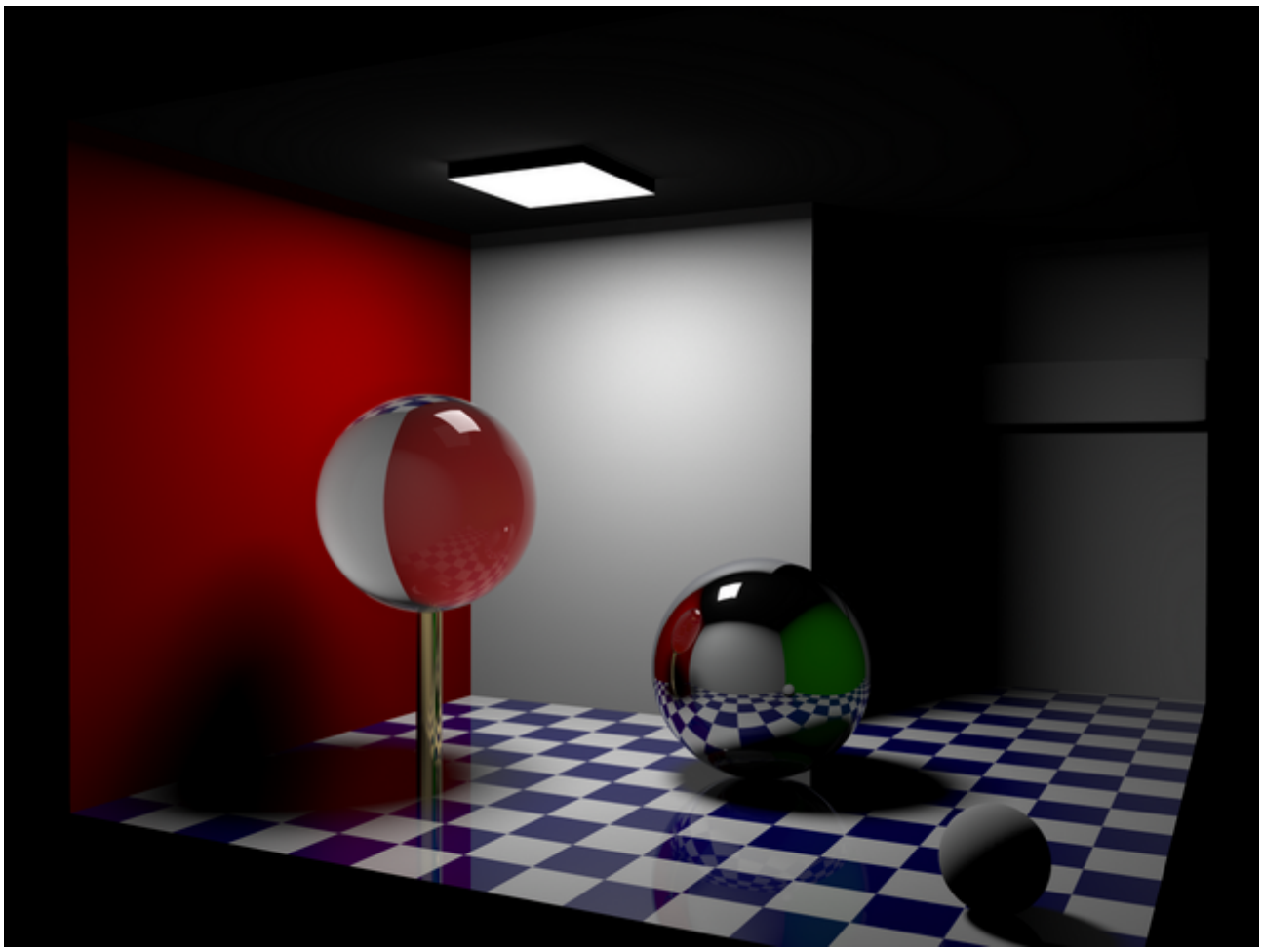

Рисунок 1.1 – Изображение, построенное с использованием локальной моделью освещения

Глобальная модель освещения учитывает лучи, не только выпущенные из источника, но и отраженные лучи от других объектов. В результате получаются более реалистичные изображения, создание которых требует больших затрат. На рисунке **??** представлено изображение, построенное с использованием глобальной модели освещения.

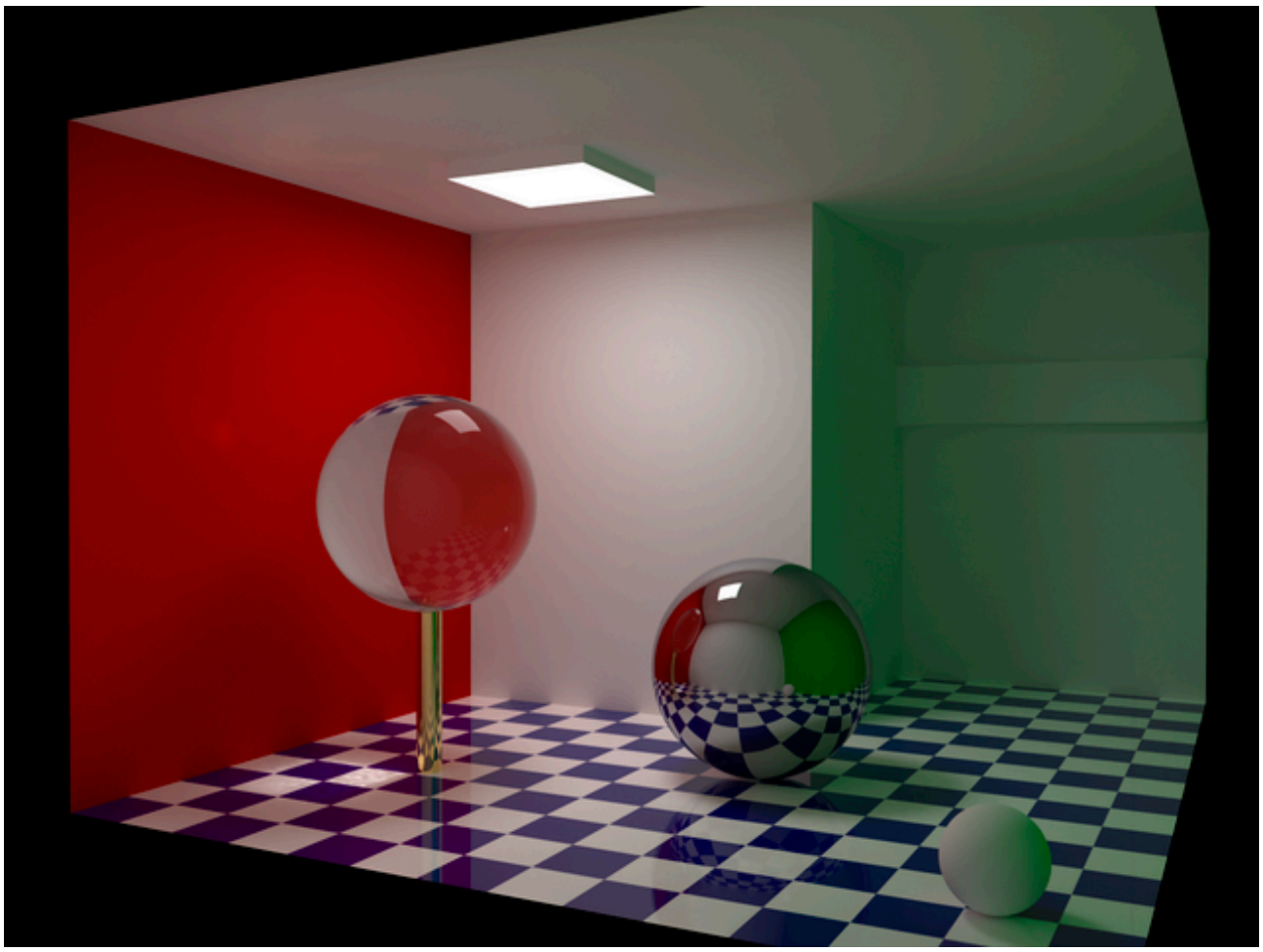

Рисунок 1.2 – Изображение, построенное с использованием глобальной модели освещения

1.4 Закраска

Закраска — это процесс определения цвета каждого пикселя на основе его освещенности и свойств поверхности объекта. Существуют различные подходы к закраске, каждый из которых даёт разные результаты в зависимости от того, какие эффекты освещенности учитываются.

Простая закраска (или плоская закраска) — это самый базовый способ закраски объектов. В этом методе для каждого полигона (или поверхности) используется один фиксированный цвет, который зависит от угла падения света на этот полигон.

Закраска Гуро -— это метод, в котором цвета интерполируются между вершинами полигона. Для каждой вершины вычисляется цвет с учётом освещенности (диффузного, зеркального и окружающего света), а затем

этот цвет интерполируется по всей поверхности полигона, создавая плавный переход. Закраска Гуро позволяет получить более гладкие и реалистичные изображения по сравнению с простой закраской.

Закраска Фонга -— это более сложный метод, в котором цвет рассчитывается для каждого пикселя, а не только для вершин. В отличие от закраски Гуро, где цвет интерполируется по поверхности, в закраске Фонга используется нормаль каждой точки на поверхности для вычисления угла между нормалью и направлением света. Этот метод позволяет точно моделировать освещенность, учитывая не только диффузное и зеркальное отражение, но и создание бликов на поверхности.

Закраска Фонга подходит для построения реалистичных изображений объектов с гладкими гранями, так как она учитывает нормали в каждой точке поверхности, что позволяет точно моделировать освещенность и плавные переходы между цветами. Однако этот метод плохо подходит для объектов с перпендикулярными гранями (например, кубов или объектов с резкими углами), поскольку в таких случаях интерполяция нормалей может привести к визуальным артефактам и неестественным переходам цвета, из-за чего объекты с резкими углами выглядят неприродно сглаженными.

1.5 Представление объектов сцены

Объекты сцены в 3D–графике часто представляют собой набор полигональных поверхностей, которые используются для аппроксимации сложных форм. Одним из наиболее распространённых типов полигонов является треугольник, который может быть задан тремя вершинами. Именно три вершины необходимо для того, чтобы задать плоскость.

Для того чтобы правильно вычислять освещенность и отображать световые эффекты на поверхности, каждому полигону требуется задвать вектор нормали, который используется для определения угла падения света и позволяет вычислять такие параметры, как диффузное и зеркальное отражение. Так как плоскость имеет две нормали, то необходимо задавать одну из нормалей вручную.

1.6 Сравнение алгоритмов удаления невидимых поверхностей и линий

В таблице 1.1. представлены результаты сравнения алгоритмов удаления невидимых поверхностей и линий по различным параметрам, где N - количество объектов на сцене, $SIZE$ - количество пикселей на экране, P - алгоритм Робертса, Z - алгоритм Z-буффер, T - алгоритм трассировки лучей.

Таблица 1.1 – Сравнение алгоритмов

Параметр	P	Z	T
Временная сложность	$O(N^2)$	$O(SIZE \cdot N)$	$O(SIZE \cdot N)$
Подготовка данных	Разбиение на выпуклые объекты	$-$	$-$
Дополнительная Память	$-$	$+$	$-$
Преимущества	Простота реализации	Эффективен для 3D сцен	Высокая реалистичность
Параллельные вычисления	$-$	$-$	$+$

1.7 Генерация горного ландшафта

Шум Перлина был предложен Кеном Перлином в 1983 году для генерации процедурных текстур и ландшафтов в компьютерной графике. Ландшафт выглядит естественным благодаря наличию в нем случайных элементов: выступов, неровностей. Шум Перлина добавляет эти случайные элементы в создаваемое изображение.

На генерируемой карте выбирается сетка, в узлах которой нужно сгенерировать векторы градиентов функции изменения высоты. Все векторы имеют единичную длину и случайное направление, как показано в формуле 1.1, где $\theta \in [0, 2\pi]$ — случайный угол.

$$g(x_i) = \cos\theta, \quad g(y_i) = \sin\theta \tag{1.1}$$

В пределах каждого промежутка сетки выбираются точки (x_p, y_p), для

которых рассчитывается значение шума.

Для каждой точки внутри ячейки сетки вычисляется смещение относительно ближайших узлов, как показано в формуле 1.2,где x_{int} и y_{int} — целочисленные координаты ближайшего узла (например, левого верхнего), а x_p и y_p — координаты точки внутри ячейки, для которой вычисляется шум.

$$u = x_p - x_{\text{int}}, \quad v = y_p - y_{\text{int}} \tag{1.2}$$

Для каждой из четырех ближайших точек сетки рассчитывается скалярное произведение между градиентом узла и вектором смещения, как показано в формуле 1.3, где g_x и g_y — компоненты градиента узла, а d_x и d_y — компоненты вектора смещения до точки.

$$\text{dot}(g, d) = g_x \cdot d_x + g_y \cdot d_y \tag{1.3}$$

где g_x и g_y — компоненты градиента узла, а d_x и d_y — компоненты вектора смещения до точки.

Для получения итогового значения шума в точке используется линейная интерполяция, как показано в формуле 1.4, где a и b — значения шума из двух ближайших узлов, а t — дробная часть координаты, сглаженная функцией fade.

$$\text{lerp}(a, b, t) = a + t \cdot (b - a) \tag{1.4}$$

Для плавного перехода между точками применяется функция сглаживания (fade function), как показано в формуле 1.5.

$$\text{fade}(t) = 6t^5 - 15t^4 + 10t^3 \tag{1.5}$$

Эта функция позволяет избежать резких изменений значений шума между узлами сетки.

На рисунке 1.3 показана визуализация шума Перлина. На рисунке 1.4 показана 3D визуализация шума Перлина.

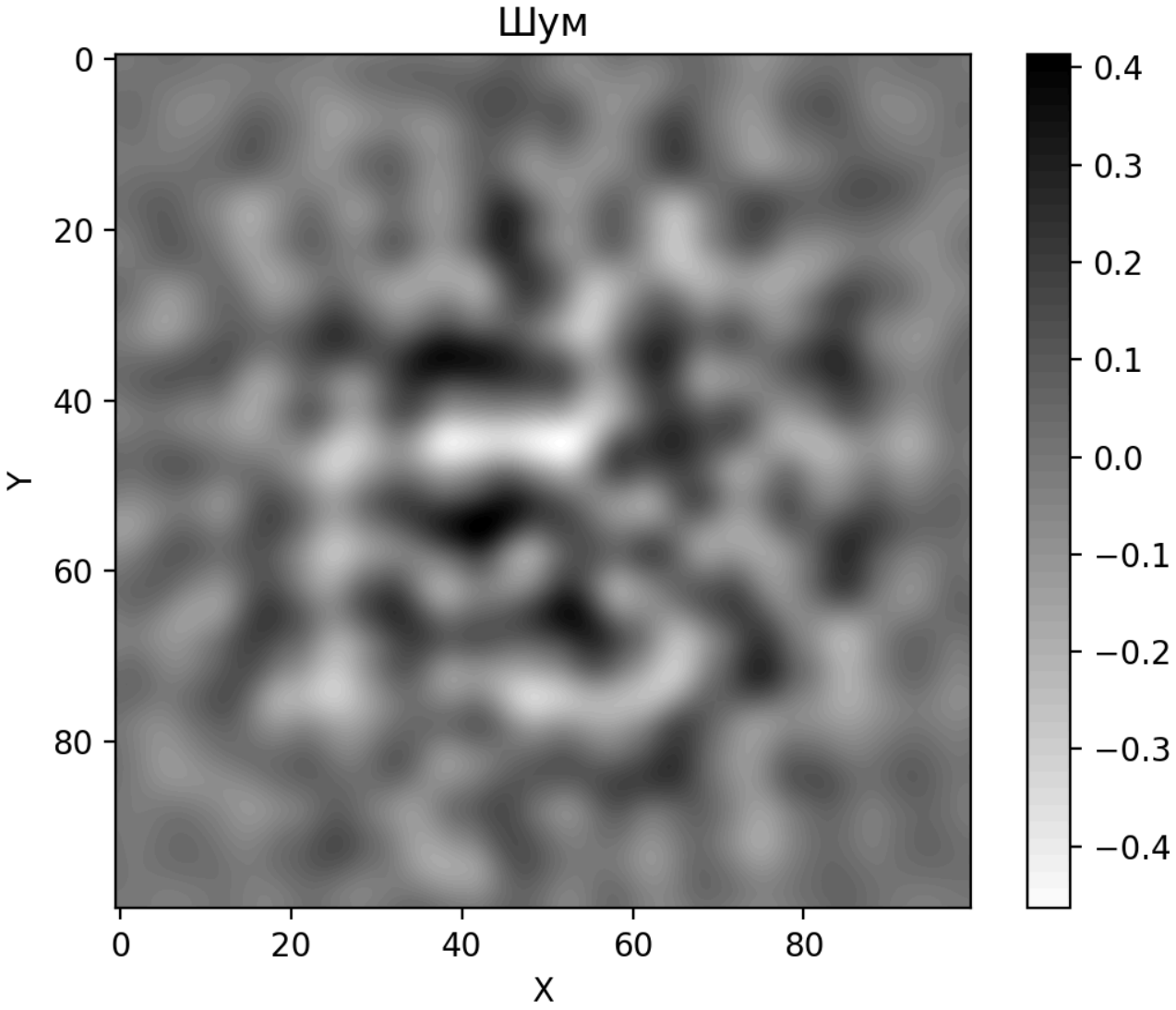

Рисунок 1.3 – Шум перлина

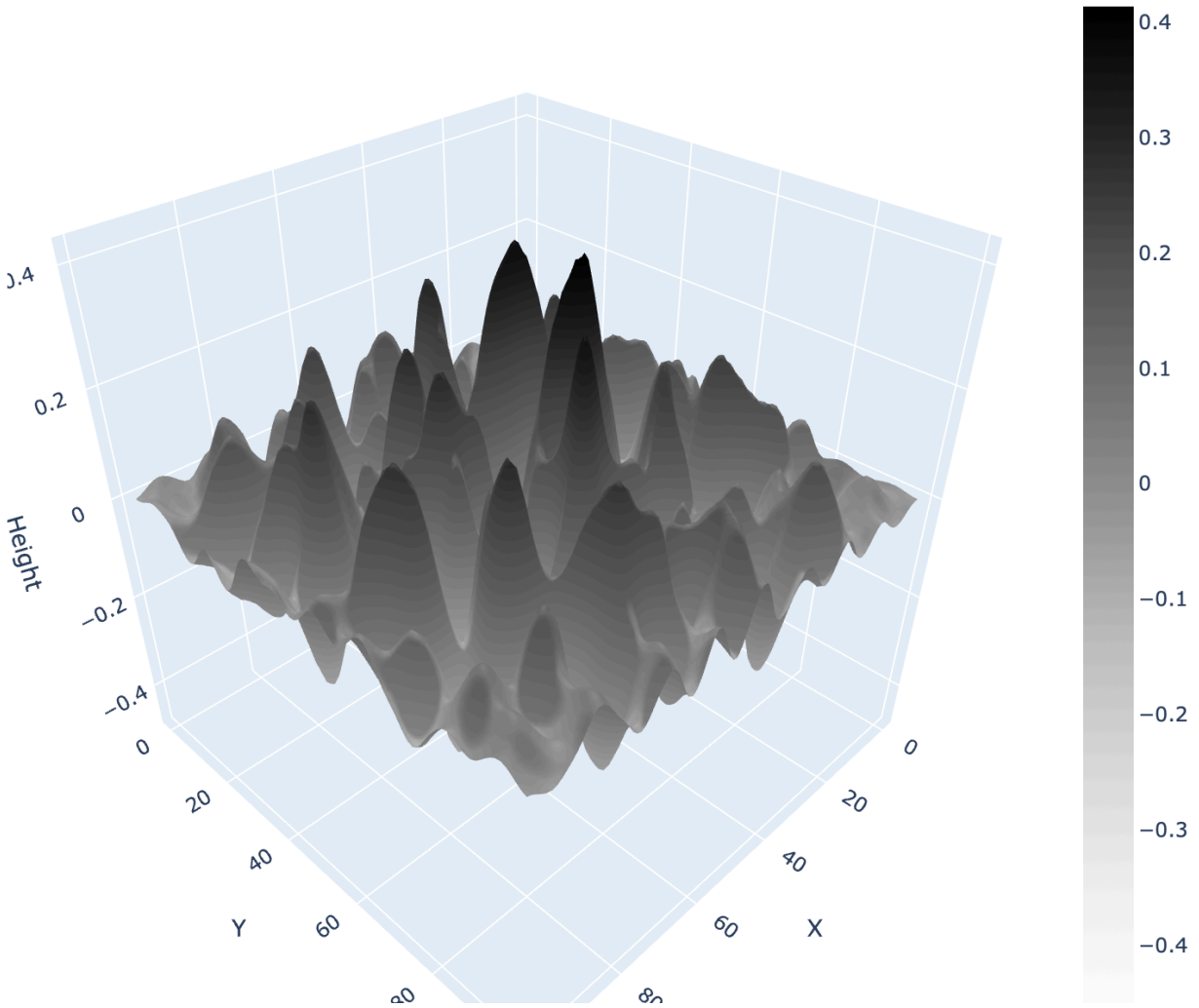

Рисунок 1.4 – 3D визуализация шума

1.8 Формализация задачи

На рисунке 1.5 представлена схема задачи построения реалистичного изображения воздушного шара на фоне горного ландшафта в формате idef0.

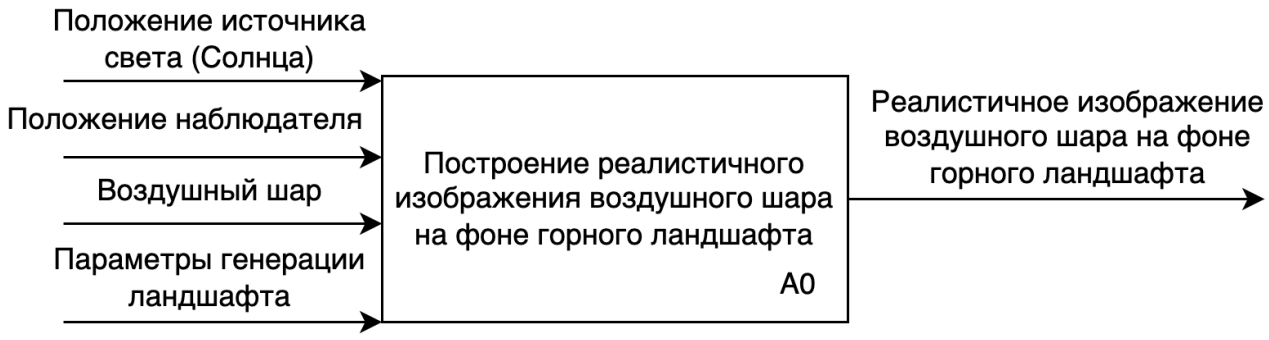

Рисунок 1.5 – Схема задачи построения реалистичного изображения воздушного шара на фоне горного ландшафта в формате idef0

1.9 Вывод

Можно сделать вывод, что для решения задачи построения реалистичного изображения воздушного шара на фоне горного ландшафта, использование алгоритма трассировки лучей будет обеспечивать большую реалистичность изображения. Также алгоритм трассировки лучей поддается распараллеливанию, в отличие от алгоритма, использующего Z-буффер. Поэтому было решено использовать алгоритм трассировки лучей.

Ввиду того, что глобальная модель освещения требует больших вычислительных затрат, было решено использовать локальную модель освещения. Для закраски был выбран алгоритм Фонга, так как почти все объекты сцены (воздушный шар, ландшафт) не имеют острых углов и являются гладкими объектами. Для генерации ландшафта был выбран алгоритм, использующий шум Перлина. Для представления объектов было решено использовать полигональную сетку, состоящую из треугольных полигонов, с заданной нормалью.

2 Конструкторская часть

2.1 Диаграмма классов

На рисунке 2.1 представленна диаграмма классов для разрабатываемого ПО в формате UML.

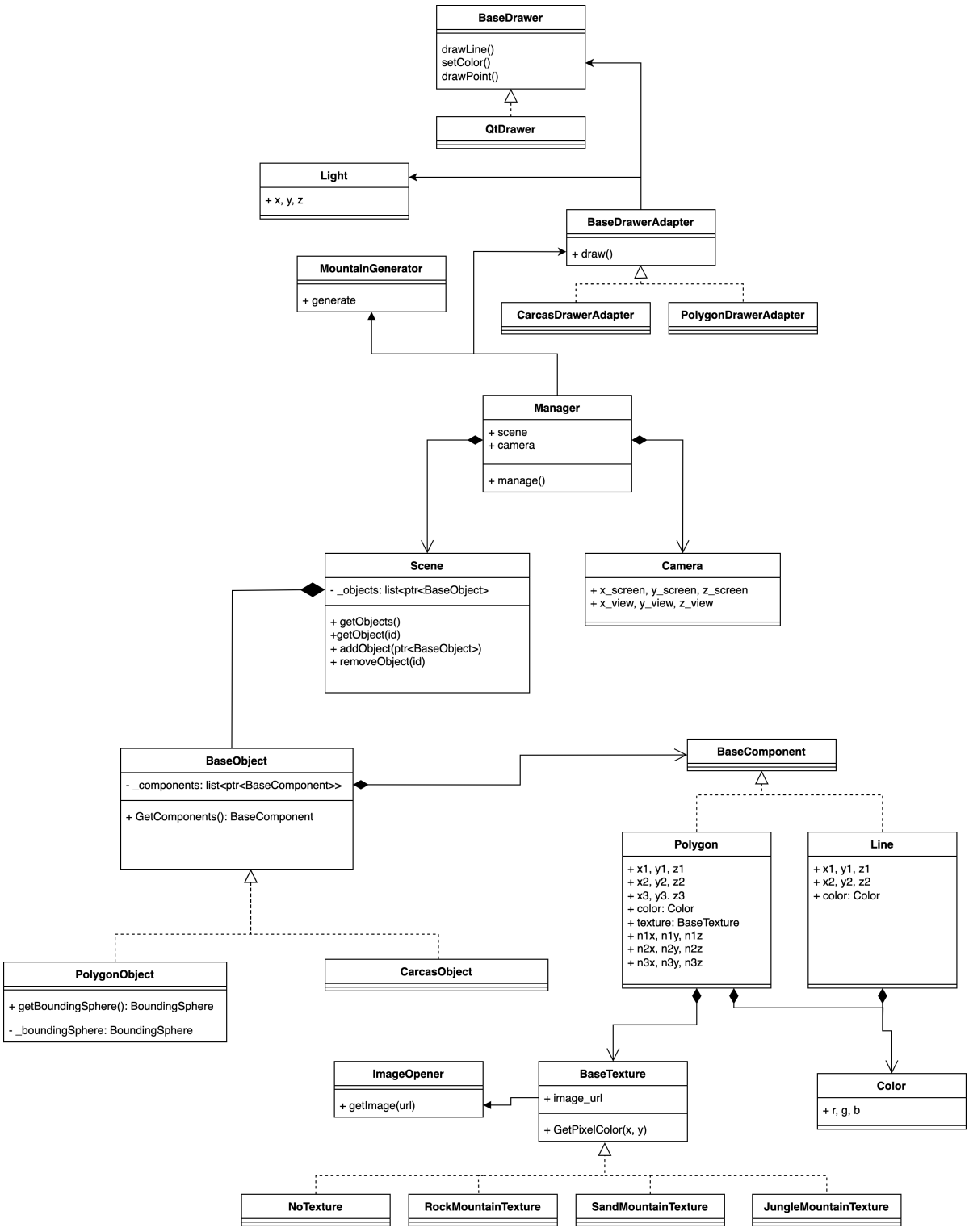

Рисунок 2.1 – Диаграмма классов в формате UML

2.2 Алгоритм построения изображения

На рисунке 2.2 изображено представление рендера изображения ввиде диаграммы idef0.

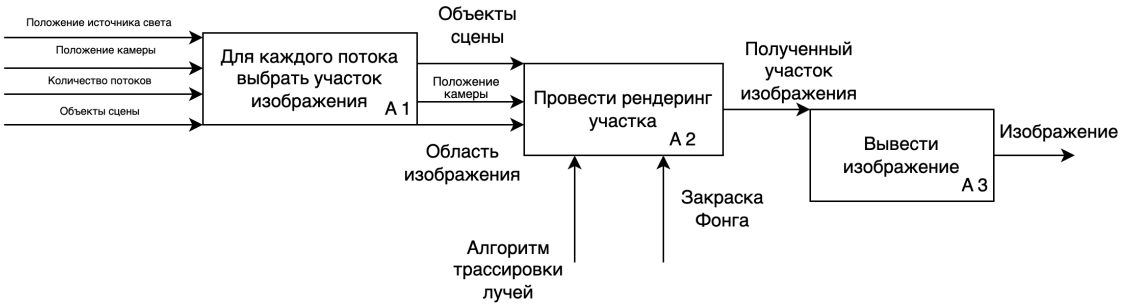

Рисунок 2.2 – Схема рендера изображения

На рисунке 2.3 изображено представление алгоритма, использующего трассировку лучей, в виде блок-схемы. На рисунке 2.4 изображено представление алгоритма, осуществляющего закраску Фонга, в виде блок-схемы.

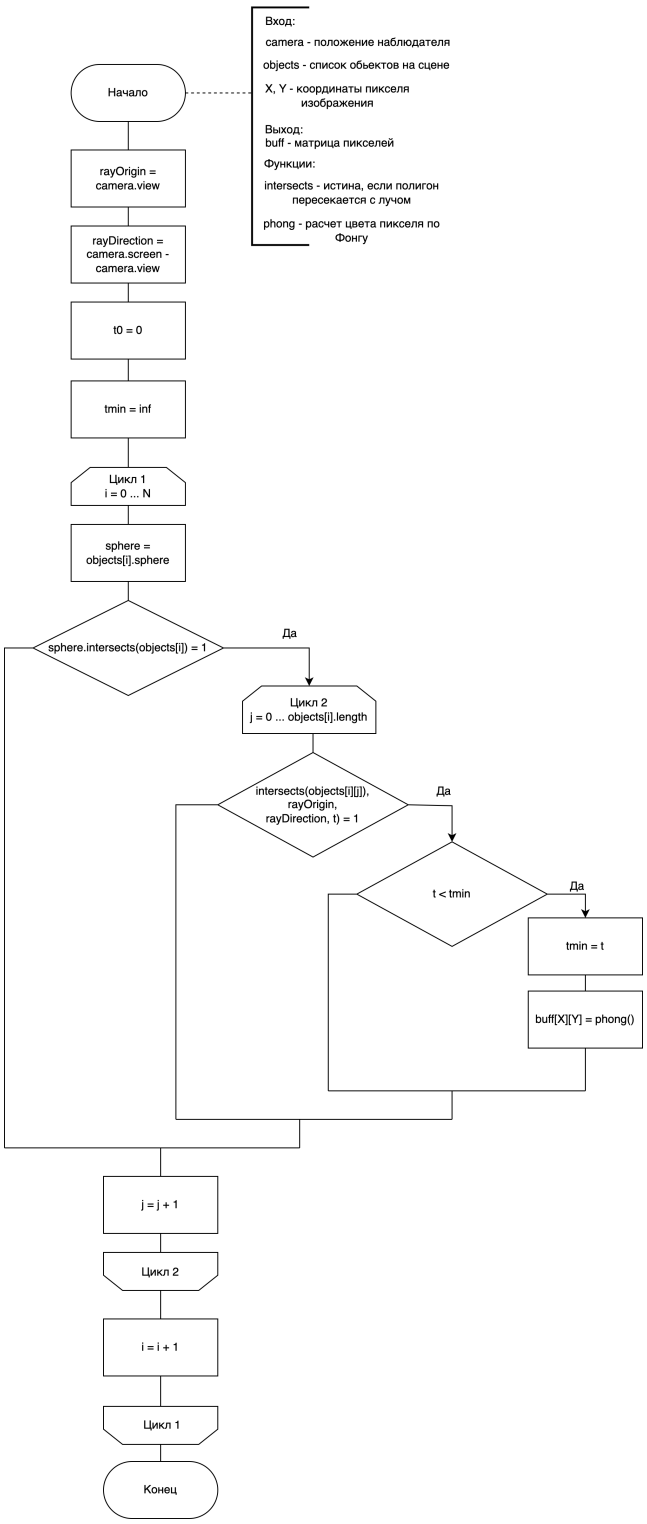

Рисунок 2.3 – Схема

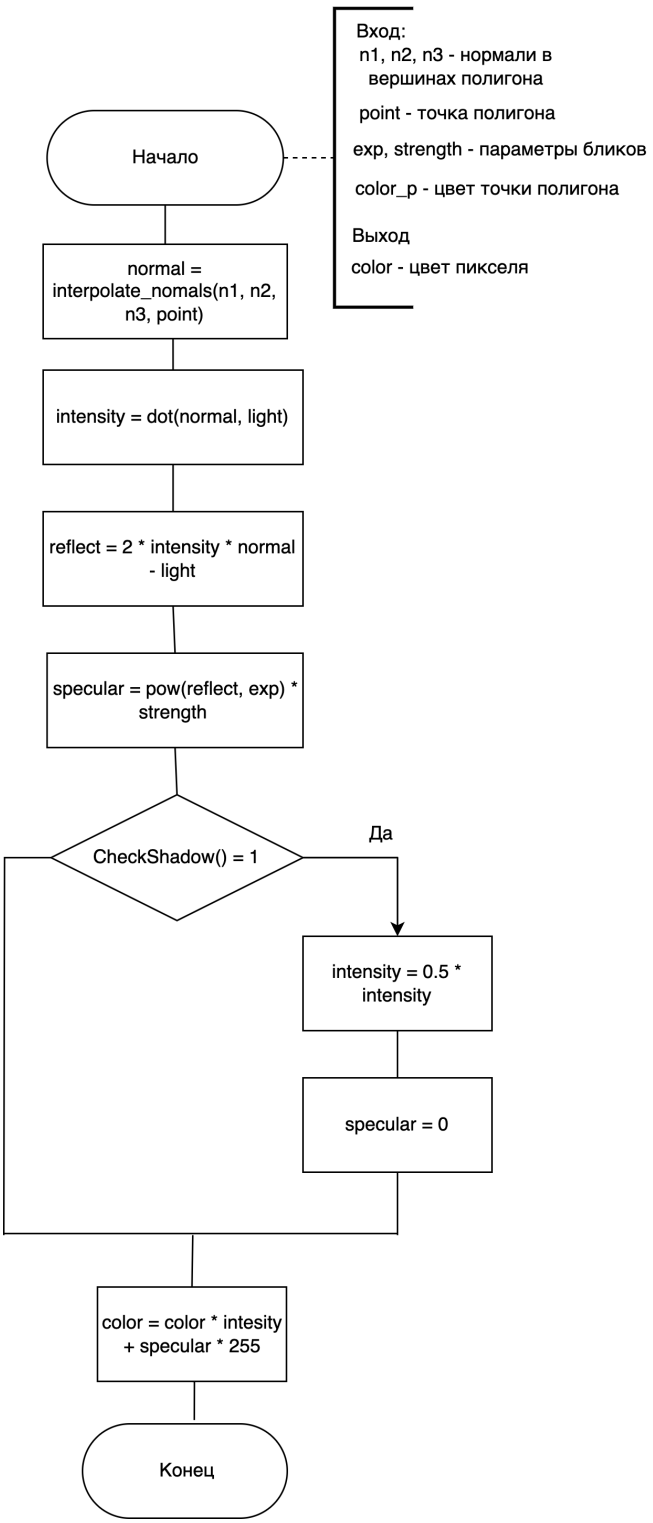

Рисунок 2.4 – Схема

2.3 Математические основы алгоритмов

Алгоритм трассировки лучей основан на поиске пересечение луча с полигоном. Для этого используется параметрическое представление луча и

треугольника. Параметрическое уравнение луча представлено в формуле 2.1, где O - начальная точка луча, D - направление луча.

$$P(t) = O + t \cdot D \tag{2.1}$$

Параметрическое уравнение треугольника представлено в формуле 2.2, где V_0, V_1, V_2 - вершины треугольника.

$$P(u, v) = (1 - u - v) \cdot V_0 + u \cdot V_1 + v \cdot V_2 \tag{2.2}$$

2.4 Математические основы алгоритмов

Алгоритм трассировки лучей основан на математике, которая описывает пересечение луча с поверхностью. Один из важных шагов этого процесса — нахождение пересечения луча с треугольной поверхностью, так как большинство объектов в 3D-графике моделируются с помощью треугольных полигонов.

В данной реализации используется метод, основанный на вычислениях с матрицами и векторами. Основной задачей является нахождение точки пересечения луча с треугольником в 3D-пространстве. Чтобы вычислить пересечение, используется параметрическое представление луча и треугольника.

2.4.1 Параметрическое уравнение луча

Луч можно выразить параметрически как:

$$\mathbf{P}(t) = \mathbf{O} + t \cdot \mathbf{D} \tag{2.3}$$

где:

- $\mathbf{O} = (O_x, O_y, O_z)$ — начальная точка луча (луч начинается в этой точке),
- $\mathbf{D} = (D_x, D_y, D_z)$ — направление луча,
- t — параметр, который определяет расстояние от начала луча до точки

пересечения.

2.4.2 Параметрическое уравнение треугольника

Треугольник определяется тремя вершинами $\mathbf{V}_0, \mathbf{V}_1, \mathbf{V}_2$ и представлен параметрически как комбинация этих точек:

$$\mathbf{P}(u, v) = (1 - u - v) \cdot \mathbf{V}_0 + u \cdot \mathbf{V}_1 + v \cdot \mathbf{V}_2 \tag{2.4}$$

где u и v — это параметры, которые описывают положение точки внутри треугольника.

2.4.3 Алгоритм пересечения

Для нахождения пересечения луча с треугольником необходимо решить систему уравнений, которая выражает пересечение двух параметрических уравнений: луча и треугольника. Для этого используется метод матричного детерминанта для вычисления коэффициентов t, γ и β, которые определяют положение точки пересечения на луче и в пределах треугольника.

Для вычисления пересечения вычисляется детерминант матрицы, которая зависит от разности координат вершин треугольника и направлений луча. Если детерминант слишком мал, то пересечения нет, так как луч и плоскость треугольника не пересекаются.

Детерминант **A** для трех векторов $\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3$ вычисляется по формуле 2.5, где:

- $a = v_0[X] - v_1[X]$,
- $b = v_0[Y] - v_1[Y]$,
- $c = v_0[Z] - v_1[Z]$,
- $d = v_0[X] - v_2[X]$,
- $e = v_0[Y] - v_2[Y]$,
- $f = v_0[Z] - v_2[Z]$,

- $g = \text{rayDir}[X]$,
- $h = \text{rayDir}[Y]$,
- $i = \text{rayDir}[Z]$.

$$\det(A)) = \begin{vmatrix} a & b & c \\ d & e & f \\ g & h & i \end{vmatrix} = a(ei - fh) - b(di - fg) + c(dh - eg) \tag{2.5}$$

Если детерминант матрицы $\det(A)$ не равен нулю, то вычисляется параметр t, который определяет расстояние от начала луча до точки пересечения по формуле 2.6, где:

- $j = v_0[X] - \text{rayOrigin}[X]$,
- $k = v_0[Y] - \text{rayOrigin}[Y]$,
- $l = v_0[Z] - \text{rayOrigin}[Z]$.

$$t = -\frac{(f \cdot (a \cdot k - j \cdot b) + e \cdot (j \cdot c - a \cdot l) + d \cdot (b \cdot l - k \cdot c))}{\det(A)} \tag{2.6}$$

2.5 Вывод

В данном разделе были представлены схемы алгоритмов рендеринга изображения, диаграмма классов, а так же математические основы алгоритма поиска пересечения луча с полигоном.

3 Технологическая часть

3.1 Средства реализации

В качестве языка программирования для реализации программного обеспечения был выбран язык программирования C++, так как он позволяет реализовать все выбранные алгоритмы и имеет поддержку нативных потоков.

Для визуализации изображения и реализации интерфейса был выбран фреймворк QT так как в нем есть поддержка вывода изображений.

3.2 Пример работы программы

На рисунках **??**, **??**, **??**, **??**, **??**, **??**, **??** представлены примеры работы программы.

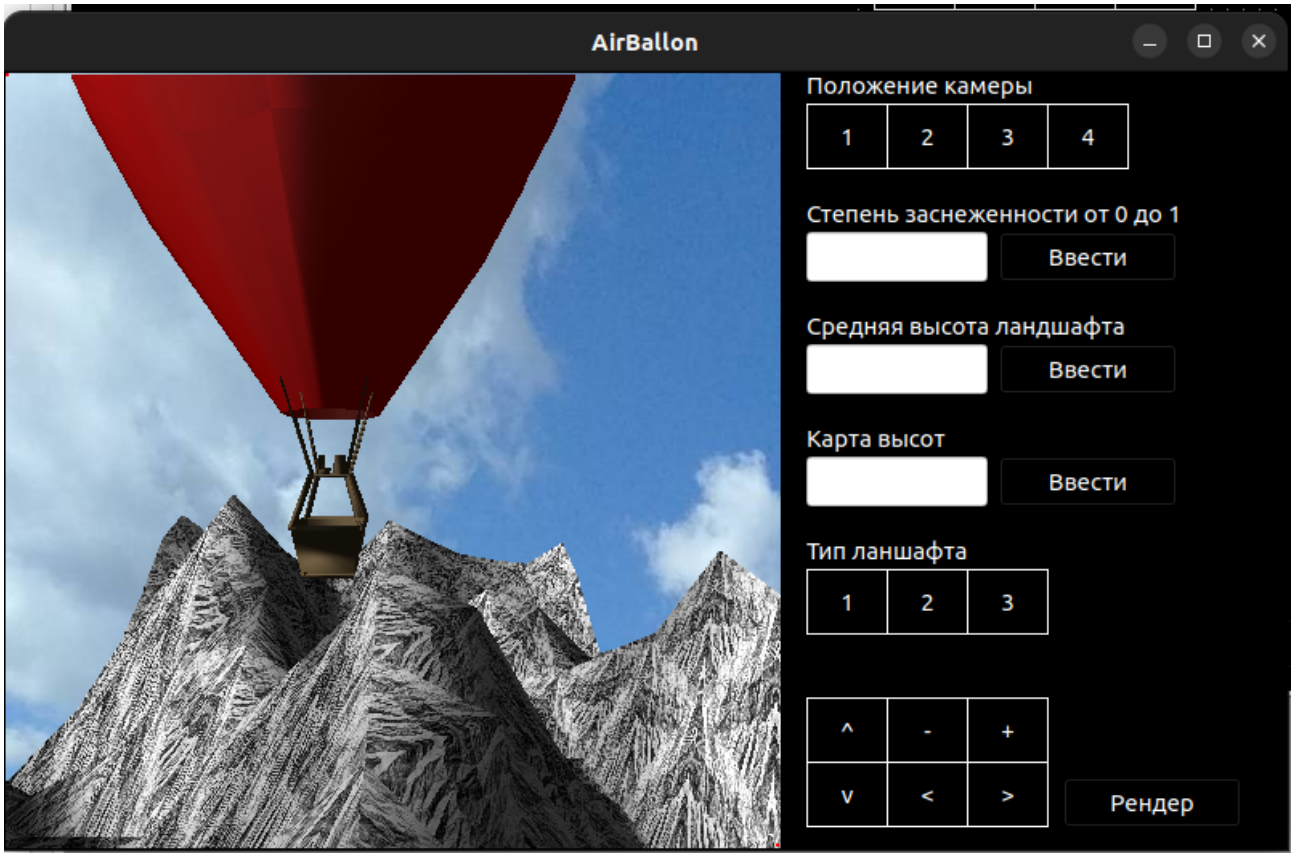

Рисунок 3.1 – Интерфейс программы

Рисунок 3.2 – Пример созданного изображения снежного ландшафта с видом из корзины

Рисунок 3.3 – Пример созданного изображения тропического ландшафта

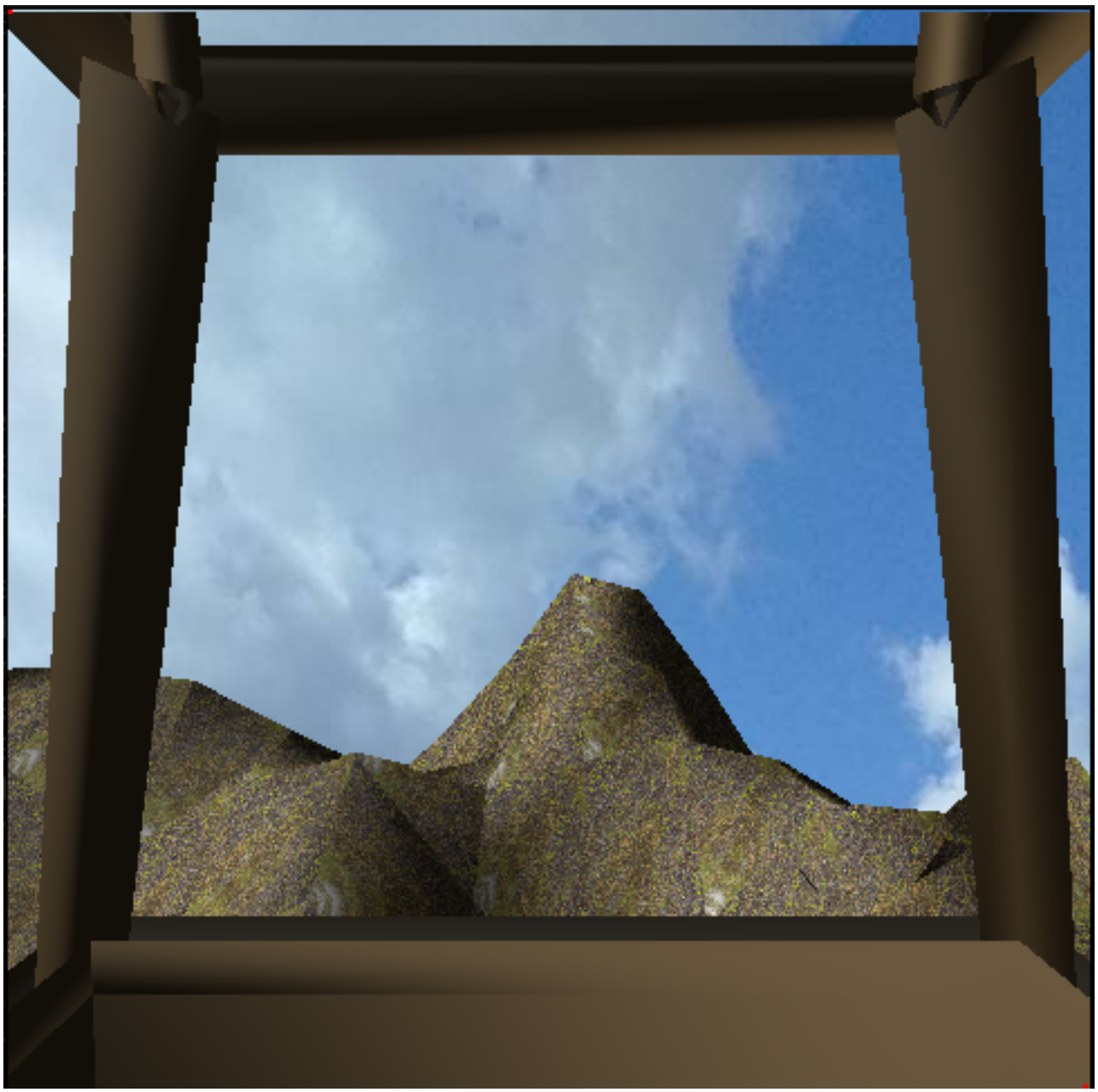

Рисунок 3.4 – Пример созданного изображения тропического ландшафта с видом из корзины

Рисунок 3.5 – Пример созданного изображения песчаного ландшафта

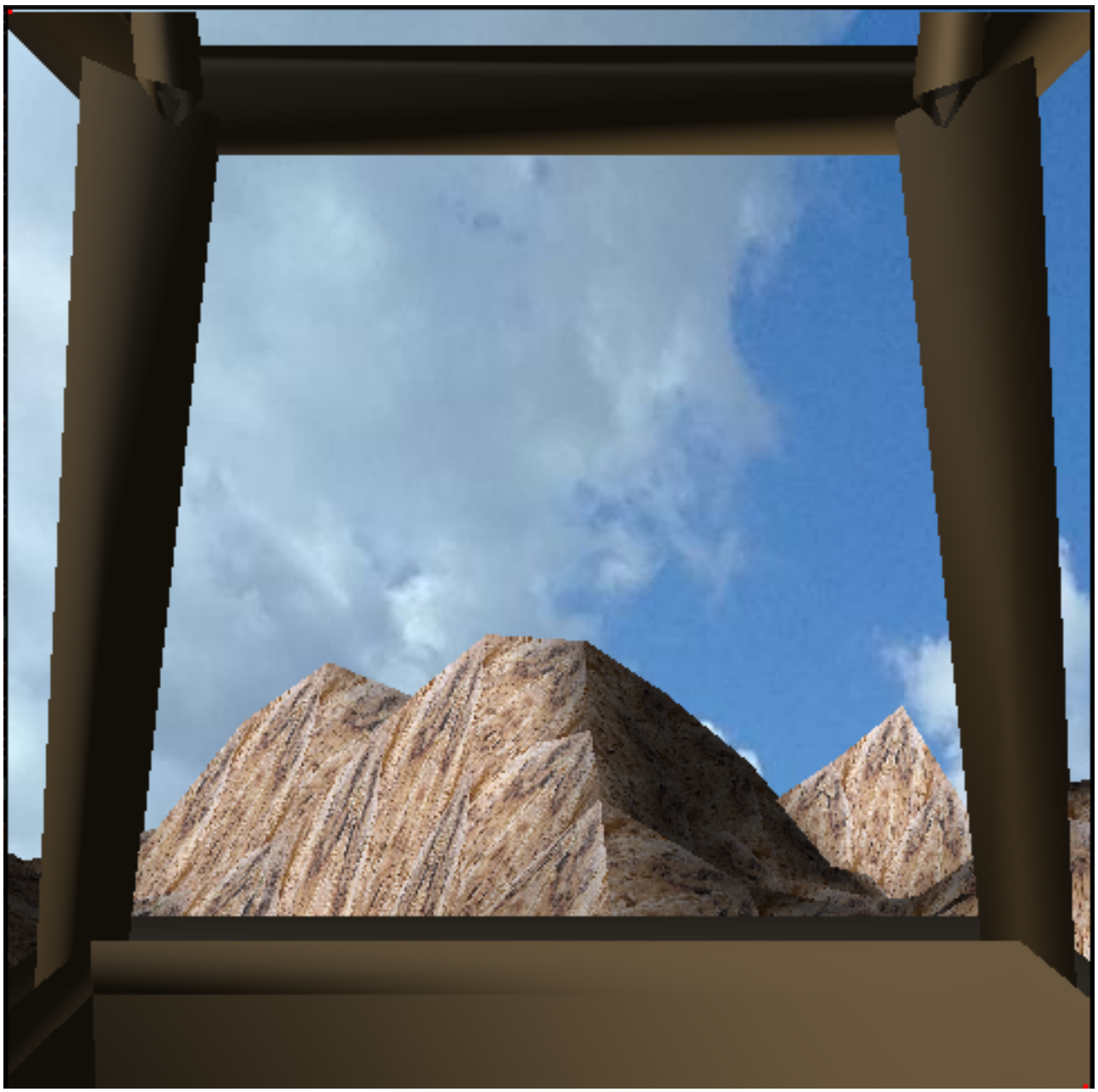

Рисунок 3.6 – Пример созданного изображения песчаного ландшафта с видом из корзины

Рисунок 3.7 – Пример созданного изображения песчаного ландшафта без воздушного шара