|  |  |
| --- | --- |
|  | **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 КУРСОВОМУ ПРОЕКТУ***

***НА ТЕМУ:***

Моделирование реалистичных изображений облаков

Студент \_ИУ7-51б **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_**Сушина А.Д.**\_\_**

(Группа) (Подпись, дата) (И.О.Фамилия)

Руководитель курсового проекта **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

(Подпись, дата) (И.О.Фамилия)

Консультант **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

(Подпись, дата) (И.О.Фамилия)

Аналит часть рпз:

1. Более подроьная формулировка тз

2. Формализация (составлениее 3хмерной математической модели) объеетов синтезируемой сцены

3. Анализ возможных методов (алгоритмов) решаемой задачи + выбор метода/вывод о необходимости модификации известных методов

Конструкторская часть:

Технологическая часть:

7. Выбор среды, языка программирования

8. Разработка структуры и состава классов (если используется ооп)

9. Сведенья о разработке отдельных модулей

10. Разработка интерфейса

11. Разраюотка тестовых наборов данных

12. Проведение тестирования и отладки программы

13. Проведение экспериментов, построение графиков, исследования зависимостей (исследовательский раздел)

14. Выводы

15. Список источников

15. Приложения

ПРИМЕРНОЕ СОДЕРЖАНИЕ СОСТАВНЫХ ЧАСТЕЙ ПРОЕКТА:

## 1.1. Техническое задание

Для реализации цели было составлено следующее техническое задание:

Разработать программу моделирования реалистичного изображения облаков. Пользователем должен быть выбран размер облака, а также степень покрытия.

Объем и содержание проекта:

Пользователь вводит необходимую информацию в диалоговые окна, затем создается изображение по заданным параметрам.

Описание предметной области.

# ВВЕДЕНИЕ

С появлением систем виртуальной реальности, в которых наблюдатель погружается в мир модели, возникает потребность визуализации виртуальной среды, максимально приближенной к естественным условиям. Такие объекты как ландшафт, водная поверхность, растительность, небо и облака являются неотъемлемой частью практически любой естественной сцены. Моделирование облаков как природного феномена является одной из наиболее важных и трудных задач, решаемых при рендеринге реалистичного неба. Алгоритмы визуализации облаков используются во всех приложениях, где необходимо создать максимально естественную среду, а также в приложениях, в которых облака играют основную роль — в авиасимуляторах.

Целью данной работы является изучение различных алгоритмов визуализации облаков и реализация одного из них. При этом необходимо обеспечить возможность настройки параметров облака(размеров, плотности).

Для достижения этой цели поставлены следующие задачи.

1. Изучить предметную область.
2. Изучить существующие алгоритмы визуализации облаков и провести их сравнительный анализ.
3. Разработать программу на основе одного из существующих алгоритмов. Программа должна соответствовать техническому заданию и создавать реалистичные изображения облаков.

# 1. АНАЛИТИЧЕСКИЙ РАЗДЕЛ

В данном разделе будут рассмотренны существующие алгоритмы и их особенности, а так же будут представленные выбранные методы и обоснован их выбор.

## 1.1 Анализ предметной области

При решении проблемы отображения больших открытых пространств качественная визуализация облаков оказывается мощным инструментом повышения реализма виртуальной обстановки.

Метеорологи классифицировали облака в зависимости от их внешнего вида. Каждый из этих типов имеет латинское название, такое как «stratocumulus (слоисто-кучевые)». Эти названия формируются из следующих корней: кучевые облака (cumulus) – плотные, днем ярко-белые облака со значительным вертикальным развитием. Высота нижней границы обычно от 800 до 1500 м, иногда – 2–3 км и более. Толщина составляет 1–2 км, иногда – 3–5 км. Верхние части кучевых облаков имеют вид куполов или башен с округлыми очертаниями. Слоистые облака (stratus) образуют однородный слой, сходный с туманом, но расположенный на некоторой высоте (чаще всего от 100 до 400 м, иногда 30–90 м). Обычно они закрывают все небо, но иногда могут наблюдаться в виде разорванных облачных масс. Перистые (cirrus) – раздельные, тонкие, нитеобразные облака в виде белых тонких волокон или чуть сероватых вытянутых гряд и клочьев, часто имеющие вид бородки пера, обыкновенно белого цвета; иногда располагаются полосами, пересекающими небесный свод подобно меридианам.

Основным источником света в дневное время служит солнце, как результат – цвет облаков в большей степени зависит от солнца. Вот почему мы видим белые облака в полдень и красные на закате. Однако не только солнце влияет на цвета облаков. Они так же зависят от цвета неба и земли. Ночью основным источником света становиться луна, когда она видима.

Облака представляют собой динамический объект, структура которого сложна и неоднородна: плотность, с которой распределены по облаку капли воды неравномерна как на границах, так и внутри облака. Она меняется постоянно, в особенности при изменении погоды и при появлении осадков. Но при визуализации внимание уделяется в основном частицам на поверхности облака, тогда как плотность частиц, находящихся внутри может считаться постоянной, поскольку она влияет лишь на преломление и рассеивание света, и при визуализации это можно отразить, придавая облакам белый цвет.

## 1.2 Алгоритмы моделирования облаков

Облака по своей структуре являются объектами, напрямую не представимыми в полигональной форме. На данный момент известно три способа аппроксимации неба и облаков:

1. Использование заранее нарисованных текстур неба и интерполяция между ними.
2. Генерация двумерных изображений облаков с помощью шума Перлина.
3. Представление облака как набор полигонов с текстурами, всегда обращенными к наблюдателю.

Рассмотрим каждый из способов отдельно.

### Использование заранее нарисованных текстур неба и интерполяция между ними

Если камера находится достаточно близко к поверхности земли, то можно использовать заранее нарисованные текстуры для отображения неба. Тогда небесную поверхность можно представить как плоскость, с нанесенной на нее текстурой. Для достижения большей реалистичности следует представить небесную поверхность как небесную сферу, т.е. вместо плоскости использовать некую поверхность, представляющую собой часть сферы. Эта поверхность должна быть пологой, чтобы создавался эффект удаленности горизонта. За счет использования готовых текстур высокого качества можно добиться практически фотореалистичного качества.

Для визуализации динамического неба можно использовать следующие подходы:

* Использовать несколько слоев текстур и сдвигать их отностительно друг друга.
* Использовать интерполяцию между несколькими картинами неба.

Для визуализации смены дня и ночи можно изменять гамму цветов в зависимости от времени суток и применять пост-эффекты свечения солнца и луны.

У данного метода есть существенные недостатки:

1. Построить изображение можно только в случае, если наблюдатель находится близко к поверхности земли. Невозможно построить изображения для произвольной позиции наблюдателя.
2. Для того, чтобы получить реалистичную картину неба необходим большой набор текстур высокого разрешения для интерполяции. Это приводит к большим затратам текстурной памяти.

Преимущество данного метода в возможности получения высоко качества изображения.

Данный метод может быть использован для систем, не предъявляющих высоких требований к динамичности неба и облаков, в которых наблюдатель расположен близко к земле. [1]

### Генерация двумерных изображений облаков с помощью шума Перлина

Использование шума Перлина для моделирования облаков является наиболее распространенным способом из-за его простоты и достаточной степени реалистичности. Алгоритм шума Перлина широко применяется для генерации туманов, облаков, огня и различных эффектов. Он основывается на том же представлении небесной поверхности как часть сферы, а облаков как двумерную текстуру.

Облачная поверхность обладает неравномерной структурой, но она не совсем хаотична. В одних местах возможно скопление "массы", а в других некая разреженность. Поэтому для визуализации облаков можно использовать хаотичную функцию, ядро которой подчиняется некоторому определенному закону. На роль такой функции отлично подходят функции Perlin Noise. [2]

Функции шума Перлина являются комбинацией шумовой функции и интерполяционной функции. Аргументами для шумовой функции выступают целые числа.

Чтобы реализовать шум Перлина, необходимо построить ряд функций, основными величинами для которых будут частота и амплитуда. Амплитуда - значения локальных экстремумов, а частота - величина, обратная длине волны. Сначала строится функция с минимальной выбранной частотой и максимальной амплитудой. Затем, для построения каждой следующей функции постепенно увеличивается частота и уменьшается амплитуда. Последняя функция будет иметь максимальную частоту и минимальную амплитуду. Каждая из функций называется октавой. Затем необходимо сложить все эти функции. В результате получается функция обладающая гладкостью и в то же время достаточной степенью хаотичности. [3] Алгоритм работы показан на рисунке 1.1.1

|  |
| --- |
| Рис 1.1.1 Алгоритм работы шума Перлина для реализации текстуры облаков |

Аналогично, можно реализовать алгоритм, который с помощью этих функций будет создавать изображение, соответствующее конечной функции.

С помощью шума Перлина можно создавать двумерные карты облаков. Для регулирования плотности облаков достаточно будет установить все пиксели меньше какого-то значения в цвет фона. На рисунке 1.1.2 можно увидеть результат такого отсечения.

|  |
| --- |
| Рис 1.1.2 Регулирование плотности облаков посредством отсечения части пикселей с минимальными значениями |

Преимущество данного метода перед предыдущим заключается в отсутствии необходимости выделения дополнительной памяти под текстуры. На реалистичность изображения влияет количество октав. Облака могут двигаться относительно наблюдателя, динамически исчезать и появляться. Однако это все также двумерная текстура и, как и в предыдущем случае, невозможно построить объемное изображение и визуализировать такие эффекты, как пролет через облако.

### Трехмерное моделирование облаков

Трехмерное моделирование облаков является более реалистичным и физически точным. Оно позволяет визуализировать гораздо большее число явлений, происходящих в облаках. В силу своей структуры облака не могут быть напрямую представлены полигональными моделями, что затрудняет их отображение как трехмерных объектов. Можно представить облако в виде набора частиц с текстурами прозрачности. В этом случае контур и изображение облаков определяется перекрытием полупрозрачных частиц, что соответствует физической структуре настоящего облака.

Модель облачной поверхности (то есть такой поверхности, которая покрывает все небо в определенном диапазоне) представляет из себя трехмерную решетку. В узлах этой решетки находятся воксели - минимальная графическая единица, операции над которой выполняются как над единым целом и которая визуализируется с помощью простейшего графического объекта.

Для создания частиц облака можно использовать возможности шума Перлина. В алгоритме шума Перлина нет никакого ограничения размерности, он может быть сколь угодно больших измерений. Результаты трехмерного шума Перлина можно с успехом использовать для создания формы объемным облакам. В этом случае яркость выступает размером частицы и может служить ключом к выбору текстуры для этой частицы, что увеличивает возможности влияния на форму облака, прозрачные места в текстуре означают отсутствие частиц. [1]

Существует также вариант реализации основанный на двумерной карте высот, полученной на основе двумерного шума Перлина. В таком случае в памяти компьютера не требуется хранить трехмерную решетку — для моделирования трехмерного облака используется только карта высот. Ее создание является наиболее важной задачей, поскольку именно она будет отвечать за форму и структуру облачности, а следовательно, влиять на ее реалистичность. При этом, все эффекты (изменение формы облака, ветер) реализуются за счет перемещения по карте высот, которая, соответственно, должна иметь достаточную площадь. На каждый пиксель карты приходится одна из градаций серого, представленная, например, числом от 0 до 255. На основе карты строится трехмерная сетка, длина и ширина которой совпадают с размерами текстуры, а высота – с некоторым заранее определенным числом. Для каждого вокселя предлагается использовать параметр, отвечающий за присутствие облака в соответствующей точке решетки. Это значение должно проставляться для всех элементов трехмерной сетки в соответствии с картой высот. Воздействие ветра на такую структуру, ее перемещение, изменение, можно реализовать за счет сдвига заполненных вокселей относительно трехмерной сетки. [2]

Использование шума Перлина привносит все свои возможности и достоинства в алгоритм генерации – высокая скорость просчета, возможность управлять плотностью облаков, эффекты клубящихся облаков, постепенного проявления и так далее. Для динамической картины неба с успехом можно использовать четырехмерный шум Перлина, где еще одной координатой выступает время.

Основная проблема этого метода визуализации облаков — завтра ресурсов на их просчет и отображение. Для физического моделирования приходится использовать большие массивы данных с существенным с точки зрения затрат вычислительных ресурсов количеством итераций для получения реалистичной картинки. Для получения динамической картины необходимо продолжать просчеты в реальном времени в масштабах видимости облаков, что может ощутимо повлиять на производительность.

Вторая проблема – затраты ресурсов на отображение облаков. В зависимости от необходимой детализации облако может включать от сотен до тысяч частиц, каждая из которых занимает существенный размер на экране, и каждая всегда рисуется с учетом полупрозрачности. Это огромные затраты заливки для графического акселератора, особенно в случае плотного облачного слоя. С другой стороны, метод является наиболее общим и универсальным методом представления облаков, и легко модифицируется для любых требований – таких, как полет сквозь облако, возможность рассмотреть облако под любым углом, полет над облачным слоем и т. д. [1]

## 1.3 Алгоритмы визуализации облаков

Из предыдущего раздела видно, что методы моделирования облаков сильно отличаются от методов моделирования других объектов. В методах, основанных на представлении объектов поверхностями, сначала создается промежуточная модель на базе плоских треугольников. Далее выполняется визуализация объектов. В то же время методы, основанные на воксельном представлении объемов, создают трехмерное изображение объекта непосредственно из объемных данных. Объемно-ориентированная технология визуализации отличается от традиционной растровой полигональной графики кардинальным образом. В полигональной растровой графике объекты задаются поверностями, представленными полигонами, накладываемыми на проволочный каркас модели. Такое задание объектов достаточно для игровых приложений, анимационных и синтетических объектов, но совсем недостаточно для отображения внутренней структуры естественных объектов или явлений. Например, в симуляторах полета необходимо отображать объемные облака, а также симулировать ситуацию пролета сквозь них. [4]

### Использование билбордов

Билборд – это полигон, все время направленный на наблюдателя. В данном подходе облачная поверхность представляет собой трехмерную решетку, в узлах которой находятся билборды. С помощью шума Перлина, путем складывания октав шума, генерируется трехмерная карта облачности. На основе это карты выставляются позиции билбордов. После чего к каждому билборду применяется небольшая текстура, соответствующая части облака. Для корректного отображения облачной поверхности, необходимо производить сортировку билбордов от дальнего к ближнему относительно позиции камеры. Так же возможно применение следующих оптимизаций: облачная поверхность делится на сектора, и выполняется проверка, какие из секторов попадают в область видимости камеры. Видимые сектора сортируются от дальнего к ближнему, а не входящие в область видимости отсекаются. Также необходимо отсекать сектора полностью скрытые за другими секторами. Так же для всех билбордов в видимых секторах необходимо рассчитать цвет с учеом источников света, плотности и рассеяния. Данный подход дает хорошие показатели реалистичности и обеспечивает работу в режиме реального времени. [5]

### Визуализация с помощью вокселей

Отличием данного подхода от предыдущего является использованием вокселей вместо билбордов. Таким образом, небо представляется трехмерной решеткой, в узлах которой находятся вокселы. С помощью шума Перлина генерируется трехмерная карта облачности, далее на основе этой карты идет выставление позиций вокселей. После чего, при помощи трассировки лучей света, рассчитывается цвет вокселей с учетом источников света, плотности облаков и рассеяния света. Данный подход дает более детализированный результат в сравнении с билбордами, но является более ресурсозатратным и сложен в оптимизации. [5]

В таком случае вводятся три битовых параметра для каждого вокселя, которые отражают его физическую природу как части облака:

**hum** - если равен 1, то это означает, что в данном вокселе собралось достаточно пара для формирования облака.

**act** - если равен 1, то это означает, что в данном вокселе началась фаза преобразования из пара в воду (то есть, в облако).

**cld** - если равен 1, то это означает, что в данном вокселе существует облако.

Работая только с этими параметрами, можно изменять структуры облачной поверхности. Можно реализовывать различные эффекты, такие как зарождение облаков, угасание облаков, перемещение облаков, изменение структуры облаков и др. [2,5] На рисунке 1.2.1 представлено графическое представление данного метода.

|  |
| --- |
| Рис 1.2.1 Графическое представление метода визуализации облаков на основе вокселов и трех битовых полей |

### Перспективная проекция

Важной составляющей сцены является точка обзора или камера. Камера позволяет получить реалистичное изображение объектов, основываясь на принципах проецирования. Важными параметрами для камеры являются точка обзора, угол обзора, размеры окна просмотра и расстояние до окна просмотра. Основываясь на этих параметрах можно получить изображение в перспективе, близкое к реальному. Так, объекты, находящиеся вдали визуально выглядят меньше, а объекты, расположенные ближе к наблюдателю, кажутся больше. На рисунке 1.2.2 можно наблюдать пример того, как точка проецируется на экран камеры.

|  |
| --- |
| Рис 1.2.2 Пример проецирования точки в окно просмотра. |

Этот метод реализации перспективы подходит и для облаков. Так, можно показать приближение облаков и даже реализовать пролет через них. Точки, которые находятся вне видимости камеры (находятся за ней или их проекции не попадают в окно просмотра) отбрасываются и появляется возможность наблюдать «внутренности» облачного слоя. Таким образом можно реализовать, например, пролет через облака. Пример работы алгоритма для облаков показан на рисунках 1.2.4 и 1.2.5.

|  |
| --- |
| Рис 1.2.4 Пример применения перспективы при работе с облаками. |
| Рис 1.2.5 Пример работы камеры с облаками. По кругу расположены разные окна просмотра и можно наблюдать разный результат работы алгоритма. Облака сгенерированы в 3D. |

## 1.4. Выбор, обоснование метода моделирования и алгоритма

Для реализации этого проекта был выбран метод визуализации облаков, основанный на трехмерной решетке вокселов. Этот метод позволяет получить изображения высокого качества и высокой степени реалистичности. Облака, полученные этим методом, можно будет рассмотреть со всех сторон, реализовать эффект пролета сквозь облако, а так же другие эффекты. Этот метод наиболее универсален, поэтому в дальнейшем может использоваться в различного рода приложениях.

Методы, основанные на 2D текстурах не дадут достаточно реалистичного трехмерного изображения, поэтому их невозможно использовать.

Для создания таблицы вокселов будет использован трехмерный шум Перлина, так как задание значений вокселов вручную требует слишком большого количества усилий. Таким образом облака будут генерироваться процедурно. Использование шума Перлина позволит использовать его преимущества и настраивать плотность облаков.

Для визуализации будет использован метод билбордов, то есть каждая частица облака будет представлена, как небольшая текстура, всегда обращенная к наблюдателю. Также будет реализована камера и использована перспектива для создания наиболее реалистичной сцены и возможности пролета сквозь облака.

# 2. КОНСТРУКТОРСКИЙ РАЗДЕЛ

Разработка алгоритма решения задачи

Выбор используемых типов и структур данных

Разработка структуры программы

В этом разделе будет представлен алгоритм решения поставленной задачи, а также описаны используемые типы и структуры данных, разработана структура программы.

## 2.1. Алгоритм работы приложения

Исходя из вышесказанного, можно составить следующий алгоритм работы приложения:

1. ввод пользовательских данных;
2. создание сцены;
3. создание трехмерной сетки на основе шума Перлина;
4. применение пользовательских настроек плотности и выборка значимых точек;
5. преобразования сцены: масштабирование, поворот и перемещение;
6. преобразование к координатам камеры;
7. рендеринг полученной сцены.

Далее опишем применяемые алгоритмы и их математическую модель.

## 2.2. Разработка алгоритмов

### Шум Перлина

Функция шума Перлина является результатом сложения нескольких шумовых функций с разными значениями амплитуды (amplitude) и частоты (frequency). Для построения этих функций введены следующие величины:

* seed — случайная величина, используемая в шумовой функции;
* persistence — величина, обеспечивающая зависимость амплитуда от частоты;
* octaves — количество октав в шуме.

Октавой (octave) называется каждая сгенерированная и добавленная в суммарный поток функция. Количество октав, также, является основной величиной в данном процессе и влияет на качество изображения.

Эти величины соотносятся между собой по следующим формулам:

(2.2.1)

(2.2.2) , где i — номер генерируемой функции.

Ядром функции служит шумовая функция, которая зависит от параметра seed. На вход функции подаются целые числа x, y, z, а на выходе получается число между -1 и 1.

Далее используется интерполяция, например, косинусная. На этапе интерполяции происходит сглаживание функции.

Эти операции выполняются для каждой октавы, а затем полученный результат складывается в итоговую функцию шума Перлина.

Схема полученного алгоритма представлена на рисунке 2.2.1.

|  |
| --- |
| Рис 2.2.1. Схема алгоритма получения шума Перлина |

### Создание трехмерной сетки на основе шума Перлина

С использованием функции Перлина можно реализовать трехмерную сетку — VoxelGrid. Для каждого узла сетки необходимо получить значение функции шума Перлина. Это значение будет отвечать за прозрачность данного вокселя.

Используем следующие структуры данных:

1. Структура одного вокселя:

Voxel {

QColor qcolor;

double density;

};

1. Структура сетки:

VoxelGrid {

int xcount, ycount, zcount;

double voxelsize;

double defaultDensity;

vector<Voxel> grid;

}

Для заполнения сетки достаточно пройтись по всем элементам и присвоить каждому вокселю новое значение прозрачности, полученное с помощью функции Перлина. Схема алгоритма представлена на рисунке 2.2.2

При такой реализации многие воксели получают значения близкие или равные нулю. При рендеринге они будут только замедлять работу системы. Чтобы избежать этого, реализуем кеш, в котором будем хранить все значимые точки, а так же их цвета. Стандартный цвет каждого вокселя — белый. Также во время выполнения записи в кеш можно наложить пользовательские настройки и отсечь некоторые точки, которые не удовлетворяют установленному значению плотности.

|  |
| --- |
| Рис 2.2.2 Схема алгоритма создания трехмерной сетки на основе шума Перлина |

### Преобразования камеры

Для того, чтобы была возможность рассмотреть сцену со всех сторон, необходимо реализовать преобразования камеры: масштабирование, поворот и перенос.

Для реализации матричных преобразований перейдем к однородным координатам [x,y,z,1] . В таком виде можно преобразовывать точки, умножая на матрицу преобразований размером 4х4.

Трехмерный перенос реализуется матрицей

Однако, гораздо проще и быстрее просто прибавить необходимое смещение к текущей координате.

Трехмерное частичное изменение масштаба реализуется с помощью матрицы .

Однако умножение на эту матрицу дает масштабирование только относительно начала координат. Более полное масштабирование реализуется с помощью формул:

,

,

,

где C(Cx, Cy,Cz) — центр масштабирования.

Поворот вокруг оси Z описывается матрицей

.

Поворот вокруг оси Х описывается матрицей

.

Поворот вокруг оси Y описывается матрицей

.

Применяя эти формулы, можно двигать, поворачивать и масштабировать изображение.

### Перспектива

Рассмотрим любую камеру как точку - центр проецирования и экран - плоский прямоугольник в 3D пространстве, на плоскость которого идет проецирование. Наша стандартная камера, например, задается точкой (0,0,-dist) и экраном с вершинами (-xSize/2,ySize/2), ..., (xSize/2,-ySize/2). Можно задать эту систему тремя векторами, задающими с точки зрения камеры направления вперед, вправо и вверх; вектор "вперед" соединяет центр проецирования и центр экрана, вектор "вправо" соединяет центр экрана и правую его границу, вектор "вверх", соответственно, центр экрана и верхнюю его границу. Обозначим эти вектора как p, q и r соответственно, а центр проецирования за s. Вот пример для стандартной камеры.

|  |
| --- |
| Рис 2.2.3 Стандартная камера |

Здесь (для стандартной камеры; обозначим ее вектора как Sp, Sq, Sr, Ss)

Sp = p = (0,0,dist)  
Sq = q = (xSize/2,0,0)  
Sr = r = (0,ySize/2,0)  
Ss = s = (0,0,-dist)

Любые три взаимно перпендикулярных вектора и точка - центр координат задают в 3D пространстве систему координат. Так что объект мы можем рассматривать в системе обычных координат (x,y,z), в системе координат стандартной камеры (Sp,Sq,Sr) или в системе (p,q,r), соответствующей какой-то произвольной камере. В любом случае, если (a,b,c) - координаты точки в системе координат камеры (точнее, в системе координат с центром в точке s и базисом (p,q,r)), то координаты проекции точки на экране равны

screenX = xSize/2 + xSize/2 \* a/c  
screenY = ySize/2 - ySize/2 \* b/c

В случае стандартной камеры переход от обычной системы координат к системе координат камеры очевиден:

a = x / (xSize/2)  
b = y / (ySize/2)  
c = (z + dist) / dist

Подставив это в формулы для screenX, screenY, получим как раз те самые формулы для проекции на стандартную камеру.

## 2.3. Разработка и обоснование используемых типов и структур данных

## 2.4. Разработка структуры программного комплекса

# 3. ТЕХНОЛОГИЧЕСКИЙ РАЗДЕЛ

## 3.1 Выбор и обоснование языка программирования

* 1. Для разработки был выбран язык С++ и среда разработки QtCreator. Выбор сделан в пользу данного языка программирования, так как он быстрее аналогов, предоставляет все необходимые возможности, а также знаком лично мне. Среда разработки позволяет создать удобный интерфейс для приложения, а так же упрощает работу с классами и позволяет использовать встроенные типы. Также среда разработки позволяет работать с пиксельными картами, что является необходимым для реализации данного курсового проекта.

## 3.2. Интерфейс пользователя

## 3.3. Хранение и обмен данными в системе

## 3.4. Разработка программы и тестовых примеров

## 3.5. Требования к аппаратуре

## 3.6. Требования к программному обеспечению

## 3.7. Порядок работы

## 3.8. Обращение к программе

## 3.9. Входные и выходные данные

* 1. На вход программа получает

## 3.10.Сообщения системы

# 4. ЭКСПЕРИМЕНТАЛЬНО-ИССЛЕДОВАТЕЛЬСКИЙ РАЗДЕЛ

# 4.1 Исследование характеристик программы

# 4.2. Примеры использования программы

# СПИСОК ЛИТЕРАТУРЫ

1. Практическая модель динамических атмосферных явлений при визуализации открытых пространств в системах визуализации реального времени Н.А.Елыков, И.В.Белаго, С.М.Козлов С.А.Кузиковский, М.М.Лаврентьев, Лаборатория программных систем машинной графики ИАиЭ СО РАН. Новосибирск, Россия.

2. Электронный ресурс: Визуализация неба и облаков. [Александр Мальковский](mailto:malkovsky@gin.keldysh.ru). URL: <https://www.graphicon.ru/oldgr/courses/cg02b/assigns/hw-5/hw5_cld.htm>

3. ШУМ ПЕРЛИНА КАК СПОСОБ ПОЛУЧЕНИЯ КОМПЬЮТЕРНЫХ СПЕЦЭФФЕКТОВ ПРИРОДНЫХ ЯВЛЕНИЙ Слеповичев Сергей Олегович, магистрант. Пензенский государственный университет (Россия, г. Пенза)

4. ВИЗУАЛИЗАЦИЯ ПОЛУПРОЗРАЧНЫХ ОБЪЕКТОВ НА БАЗЕ ФУНКЦИЙ ВОЗМУЩЕНИЯ И ПРОЗРАЧНОСТИ С.И. Вяткин, Б.С. Долговесов (Новосибирск) Российская академия наук, Сибирское отделение, Автометрия. 2005, том 41, №3

5. A Simple, Efficient Method for Realistic Animation of Clouds. Yoshinori Dobashi, Kazufumi Kaneda, Hideo Yamashita, Tsuyoshi Okita, Tomoyuki Nishita.

6. Realistic and Fast Cloud Rendering. Niniane Wang, Microsoft Corporation (now at Google Inc.) [niniane@ofb.net](mailto:niniane@ofb.net) November 11, 2003

# ПРИЛОЖЕНИЯ

## П.А. Листинг программы

## П.Б. Копии экрана

## П.В. Распечатки результатов

# ГРАФИЧЕСКАЯ ЧАСТЬ

## 1. Постановка задачи

## 2. Математические методы решения задачи

## 3. Функциональная схема системы

## 4. Схема алгоритма

## 5. Сравнительные характеристики аналогов

## 6. Листинг программы (фрагмент)

## 7. Интерфейс пользователя

## 8. Иллюстрация работы с примером задания исходных данных