*Федеральное государственное бюджетное образовательное учреждение*

*высшего профессионального образования*

|  |  |
| --- | --- |
| **Gerb-BMSTU_01** | ***«Московский государственный технический университет  имени Н.Э. Баумана»***  ***(МГТУ им. Н.Э. Баумана)*** |

ФАКУЛЬТЕТ Информатика и системы управления

КАФЕДРА Программное обеспечение ЭВМ и информационные технологии

**РАСЧЁТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**к курсовому проекту на тему:**

Моделирование реалистичной модели облаков.

Студент **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_**Буевич Е.В.\_

(Подпись) (И.О.Фамилия)

Руководитель курсового проекта **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_**Новик Н.В.**\_\_**

(Подпись, дата) (И.О.Фамилия)

Москва, 2017

**Содержание**

[Введение 3](#_Toc438583883)

[1.Аналитический раздел 4](#_Toc438583884)

[1.1 Анализ существующих алгоритмов 4](#_Toc438583885)

[1.1.1 Алгоритм итерируемых функций 4](#_Toc438583886)

[1.1.2 Алгоритм клеточного автомата 7](#_Toc438583887)

[1.1.3 Алгоритм шумовых функций 4](#_Toc438583888)

[1.1.4 Выбор алгоритма 8](#_Toc438583889)

[2. Конструкторский раздел. 4](#_Toc438583894)

[Общий алгоритм программы 9](#_Toc438583895)

[2.1 Реализация текстур облака методом шума Перлина 10](#_Toc438583896)

[2.2 Алгоритм наложения текстур 11](#_Toc438583897)

[3.Технологический раздел 13](#_Toc438583899)

[3.1 Выбор языка программирования и среды разработки 13](#_Toc438583900)

[3.2 Схемы классов 14](#_Toc438583901)

[3.3 Интерфейс программы 19](#_Toc438583902)

[4.Исследовательский раздел 22](#_Toc438583904)

[Заключение 24](#_Toc438583905)

[Список использованной литературы 25](#_Toc438583906)

# Введение

В настоящее время, компьютерная графика широко используется в современном мире. Интерес к ней проявляют представители самых разных профессий: геймдизайнеры, программисты, конструкторы, мультипликаторы, кинорежиссеры, физики, биологи и многие другие. Неудивительно, ведь с помощью методов компьютерной графики можно создать изображения, в которых нуждаются, к примеру, компьютерные игры и фантастические фильмы. Или же моделировать различные эксперименты, к примеру, для изучения физики этих процессов. Из всего выше следует, что применение машинной графики в наши дни очень многогранно.

Целью курсового проекта является разработка приложения для моделирования облаков, и отображение смоделированного облака на экран монитора. Основной задачей является реалистичная визуализация модели облака.

В рамках реализации проекта должны быть решены следующие задачи:

* Проведение анализа алгоритмов компьютерной графики, использующихся для создания реалистичной модели облака, и выбран наиболее подходящий для решения поставленной задачи.
* Подробное изучение выбранного алгоритма для поставленной задачи и обоснование выбора структур данных для описания модели облака.
* Проектирование архитектуры программы и ее интерфейс.
* Реализация выбранных алгоритмов и структур данных.
* Проведение исследования с помощью программы.

# 1.Аналитический раздел

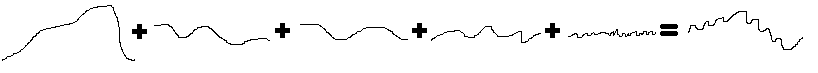
## 1.1 Выбор алгоритма построения модели облака

Для выбора алгоритма построения модели облаков, рассмотрим наиболее распространенные алгоритмы и осуществим выбор наиболее подходящего для решения задачи исходя из требований программы.

### **1.1.1 Использование шумовых функций**

Облака имеют сложную физическую структуру. В одних можно увидеть скопление их «массы», в других – разреженность, и все эти области плавно переходят друг в друга. Один из способов описать такую структуру при моделировании облачности является использование шумовой функции, например, шума Перлина (PerlinNoise)[4]. Идея этого метода состоит в построении функции, комбинирующей в себе две функции одновременно, шумовую и интерполяционную.

Чтобы ее реализовать, нужно построить множество функций, в которых основными параметрами являются частота и амплитуда. Амплитуда – значение локальных экстремумов, частота – разница между двумя нулями. Сначала строится функция с минимальной выбранной частотой и максимальной амплитудой. Далее, для построения последующих функций нужно увеличивать частоту и уменьшать амплитуду. Каждая из этих функций называется октава. Их количество непосредственно влияет на качество получившегося облака. И сложив их все, мы получим результирующую – в какой-то мере гладкую и в достаточной степени хаотичную.



*Рис.1 Получение результирующей функции.*

Полученная текстура используется как карта высот для облачной поверхности, заданной с помощью воксельной модели. Важно, чтобы ширина и длинна карты высот и трехмерной сетки воксельной модели совпадал. Каждый воксель определяется одним битовым параметром – наличием существования облака. Далее, берутся столбцы вокселей с координатами i, j в сетке и для каждого столбца проставляем присутствие облака в клетках со следующими координатами:

### Здесь N- высота трехмерной сетки, - цвет текстуры, полученной с помощью шумовых функций (в 256 градациях серого), а rnd – число, необходимое для того, чтобы модель не была симметрична.

### **1.1.2 Метод «Систем Итерируемых функций»**

Роль фракталов в машинной графике достаточно велика. Они используются чаще всего для того, чтобы с помощью нескольких коэффициентов задать линии и поверхности очень сложной формы. Так же они используются для генерации объектов, схожих с природными: деревья, береговые линии и т.д.

Для построения реалистичных облаков используют стохастические фракталы, метод «Систем Итерируемых Функций» (IteratedFunctionSystem – IFC)[1]. Его идея состоит в представлении нужного изображения с помощью системы функций из некоторого фиксированного класса функций, отображающих одно многомерное множество на другое. Наиболее часто используемая система - система из аффинных преобразований плоскости. Самая примитивная система состоит из двух уравнений:

Аффинное преобразование - композиция линейного преобразования и параллельного переноса. В двумерном пространстве для полного представления аффинного преобразования достаточно задать 6 коэффициентов.

Для построения IFS применяют кроме аффинных и другие классы простых геометрических преобразований, которые задаются небольшим числом параметров.

Например, проективные:

или квадратичные преобразования на плоскости:

Для получения итогового изображения требуется найти коэффициенты заданной системы уравнений. Сложность заключается в том, что процесс поиска достаточно трудоемкий. Для расчета коэффициентов требуется порядка ста тысяч итераций. Сама же модель генерируется с помощью итерационного процесса.



*Рис.2 Этапы генерации модели облака.*

В результате получаем изображение, деталировка которого может быть ограничена только разрешением устройства отображения.

### 

### **1.1.3 Использование клеточного автомата**

Клеточный автомат[2] – дискретная модель, которая используется во многих научных дисциплинах, в том числе и в программировании. Включает в себя решетку ячеек, каждая из которых находится в одном из конечного множества состояний, которые обычно задаются как 1 и 0.

Основная идея алгоритма[3] состоит в следующем. Модель облака будет представлять из себя трехмерную решетку, в узлах которой будут расположены воксели.

При визуализации вокселя на него будет нанесена текстура «небольших туманностей», то есть частей облака, которое будет направлено на наблюдателя. Так же вводится три битовых параметра, отвечающих за его состояние, позволяющее отражать его физическую природу как части облака. Первый параметр отражает существование облака в данном вокселе. Второй достаточное количество условий для образования облака, и в третьем – начала создания облака. Работая с этими параметрами, могут производиться различные эффекты: зарождение облака, его угасание, перемещение и др.

Поскольку облачная поверхность не стационарна, то вводятся следующие формулы, для изменений трех параметров, описанных выше:

Здесь i, j, k – координаты вокселя в клеточном автомате, – момент времени. Но этих формул может быть недостаточно, поскольку они формируют стационарную модель и никак не смогут повлиять на зарождение и угасание. Поэтому вводятся дополнительные формулы:

Здесь - пороговое значение, влияющее на угасание облаков, - пороговое значение, влияющее на изменение влажности, - пороговое значение, влияющее на изменение активности.

Используя данную воксельную модель, можно моделировать достаточно сложные облачные поверхности, однако возникает большая проблема: малейшее изменение параметров может привести к тому, что вся модель распадется на маленькие кусочки, которые будут вести себя по непонятным законам. В данном случае очень важна подгонка параметров, что само по себе очень кропотливая и сложная задача.

### **1.1.4 Выбор алгоритма**

Выбирая между данными алгоритмами, исходя из поставленных задач, нужно отметить, что модель, построенная с помощью метода IFC, очень хорошо детализирована, но помимо критерия реалистичности стоит вопрос о быстродействии. Поскольку выполняется порядка ста тысяч итераций, то этот алгоритм слишком медленный.

Что касается оставшихся двух, то можно сказать, что алгоритм с использованием шумовых функций выигрывает в быстродействии, поскольку воксель определяется с помощью одного параметра, а также не происходит вычисления параметра через формулы с предикатами. Кроме того, важно отметить, что здесь, в отличии от алгоритма с использованием клеточного автомата, не потребуется сохранять в памяти трехмерную решетку, а достаточно только сохранить карту высот. Следовательно, мы выбираем алгоритм с использованием шумовых функций.

# 2. Конструкторский раздел.

В данном разделе рассматривается реализация алгоритмов, которые будут использоваться в проекте для построения реалистичного изображения.

## Общий алгоритм программы

1. Установить параметры времени: текущее, восход солнца и его закат.
2. Сформировать виртуальный мир: создать каркасную модель солнца с текущими координатами, зависящими от времени, воксельную модель облаков и плоскость, являющуюся землей.
3. Произвести необходимое преобразование координат (повороты моделей, масштабирование и перенос) относительно камеры.
4. Создание текстур для моделей виртуального мира:
   1. Для травы выполнить генерацию текстуры по принципу случайного изменения интенсивности зеленого цвета.
   2. Для солнца/луны выполнить чтение текстуры из ранее заготовленного файла.
5. Выполнить наложение текстур солнца и травы в декартовой системе координат.
6. В цикле для каждого слоя модели облака.
   1. Выполнить генерацию текстур с помощью шума Перлина.
   2. Выполнить наложение текстур на модель облака.
7. Преобразовать координаты из декартовой системы координат в экранные, и отобразить данные модели.
8. Конец цикла.

## 

## 2.1 Реализация текстур облака методом шума Перлина

**Структура входных и выходных данных.**

С помощью алгоритма, используя размер итоговой текстуры, с помощью расчетов получаем массив цветов модели облака.

**Алгоритм текстурирования облака методом шум Перлина.**

Поочередно для каждого вокселя вычисляется цвет с помощью суммирования нескольких октав сложенных интерполяционных и сглаженных шумов в этом вокселе. Сами же суммирования двух шумов происходят следующим образом [7]:

1. Вычисляются значения сглаженной функции для соседних 4 вокселей, с помощью псевдо-случайного генератора случайных чисел, который подсчитывает значение с помощью передаваемых параметров (координат вокселя). Генератор выглядит следующим образом:

Где n вычисляется в зависимости от координат:

Полученный результат всегда варьируется от -1.0 до 1.0.

* 1. Подсчитывается сумма значений сглаженных функций «угловых» вокселей, которые расположены диагонально от заданного вокселя и делится на , где *d*–величина, зависящая от используемого пространства. К примеру, *d*=2, если используется двумерное пространство, *d*=3, если в трехмерное и т.д.
  2. Подсчитывается сумма значений сглаженных функций «боковых» вокселей, которые расположены по бокам от заданного вокселя и делится на.
  3. Подсчитывается значение сглаженной функции заданного вокселя, и делится на .

1. Берутся попарно получившиеся значения и интерполируются с помощью интерполяции Джеймса Лонга, которая используется в статье [8]:

Где a, b – пределы интерполирования, а x – число, от которого зависит результат интерполируемой функции. Обычно принимает значения от 0 до 1.

1. Далее производится интерполяция между получившимися значениями до тех пор, пока не получится одна функция.
2. Приводится получившееся значение к виду 0-255.

## 2.2 Алгоритм наложения текстур

**Структура входных и выходных данных.**

Алгоритм наложения текстур рассчитан на попиксельное наложение текстуры, поэтому на вход функции с реализацией алгоритма подаются пиксели. Далее подается размер текстуры, в которой нужно найти тексель для этого пикселя. На выходе соответственно получается тексель.

**Принцип алгоритма наложения текстур.**

После создания текстуры следующим шагом для создания реалистичной трехмерной модели объекта является текстурирование. По сути, процесс наложения текстуры заключается в последовательном высвечивании пикселей полигонов с цветом соответствующих текстелей. Тексель [9] (сокращение от английского слова Textureelement) минимальная единица текстуры трехмерного объекта, т.е. пиксель текстуры.

Очень часто при наложении текстур на сферы[10], [11] используют равнопромежуточную проекцию (UV – текстуризация). Этот метод заключается в соответствии между координатами на поверхности трехмерного объекта (x, y, z) и координатами на текстуре (u, v). Где u и v имеют значения от 0 до 1.

Для вычисления координат пикселей текстуры используют следующие формулы:

Где u и v принадлежат [0, 1], а {dx, dy, dz} – нормализованные координаты вектора, который есть разность между текущей точкой поверхности и центром сферы. Для того, чтобы получить непосредственно координаты текстуры необходимо u и v домножить на ширину и высоту текстуры соответсвенно.

# 3.Технологический раздел

В текущем разделе выбирается язык программирования, рассматриваются требования для написания данной программы и интерфейс самого приложения.

## 3.1 Выбор языка программирования и среды разработки

Для написания приложения была выбрана платформа .NET

Основные достоинства данной платформы:

* Автоматический сборщик мусора
* Реализация уже готовых форм Windows
* Расширенная возможность выделения синтаксических конструкций

Автоматический сборщик мусора и готовые формы Windows позволяют

экономить время и тратить его на реализацию самого приложения. Расширенная возможность выделения синтаксических конструкций упрощает написание кода и дальнейшую его корректировку.

Платформа .NET подразумевает написание программы на языке программирования С#, C++ и VisualBasic.

Для реализации приложения был выбран язык C# из следующих критериев:

* Автоматический сборщик мусора
* Относительная простота использования
* C# - объектно-ориентированный язык программирования предназначенный для программирования на платформе .NET

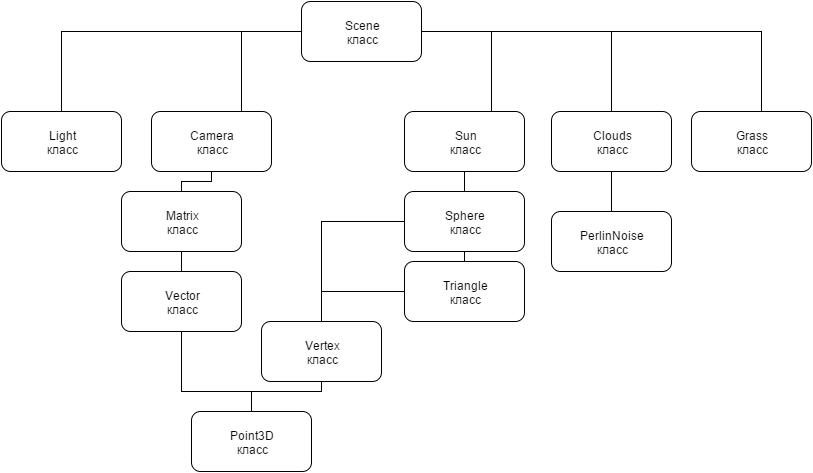
## 

## 3.2 Схемы классов

В данном подразделе представлены основные классы приложения.

**Общая схема классов.**

Построение общей схемы классов было выполнено с помощью утилиты «Draw.io».

****

*Рис. 3 Общая схема классов.*

**Управляющий класс. Класс сцены.**

****

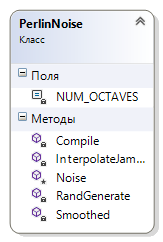
*Рис. 4 Схема класса сцены.*

**Класс облаков.**

****

*Рис.5 Класс облака*

**Класс для текстурирования облака.**



*Рис.6 Схема класса для текстурирования облака.*

**Класс камеры.**

****

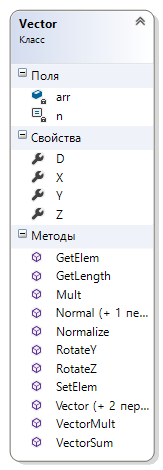
*Рис.7 Схема класса камеры*

**Класс матриц.**



*Рис.8 Класс матриц*

**Классы, необходимые для предстваления трехмерного пространства.**

****

*Рис.9 Класс Векторов Рис.10 Класс Вершин Рис.11 Класс точек*

**Класс света.**

****

*Рис. 12 Класс света*

**Класс солнца.**

****

*Рис. 13 Класс солнца*

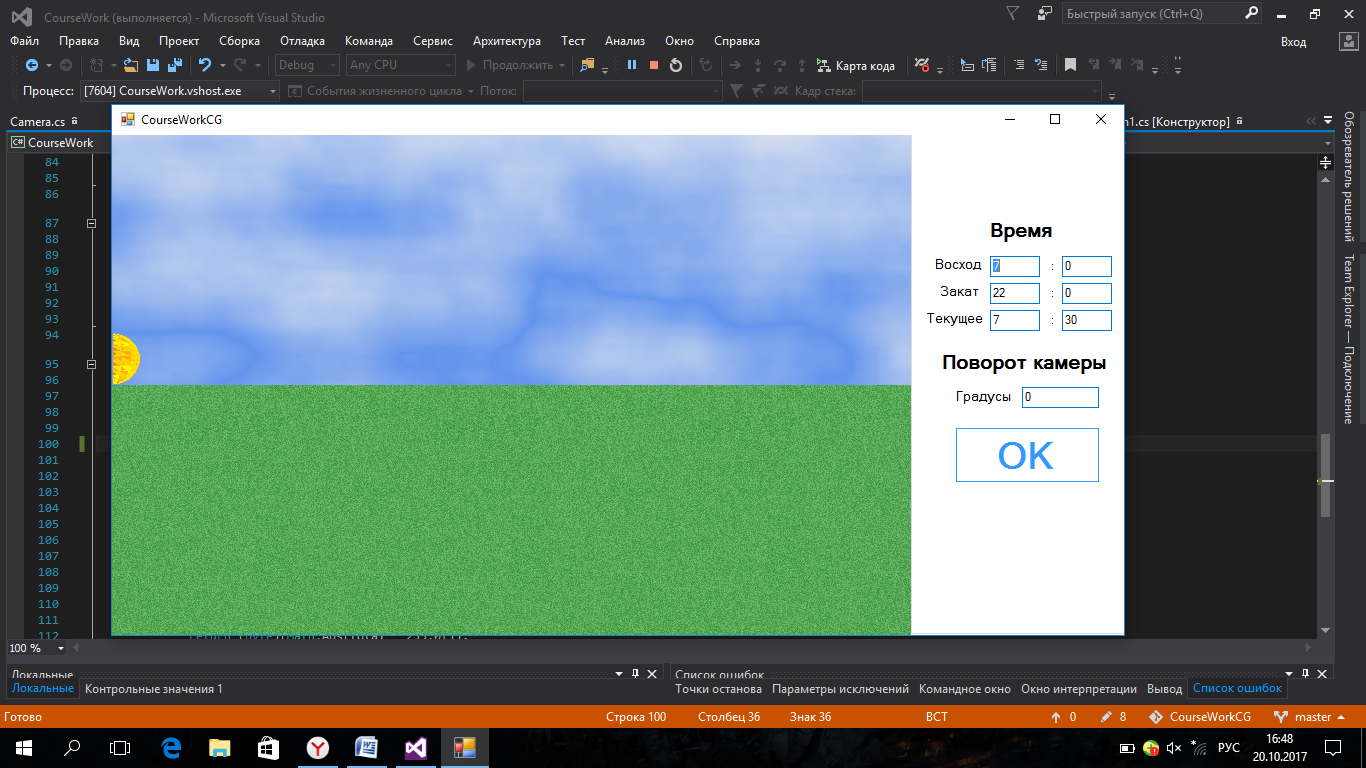
**Класс травы.**

****

*Рис. 14 Класс травы*

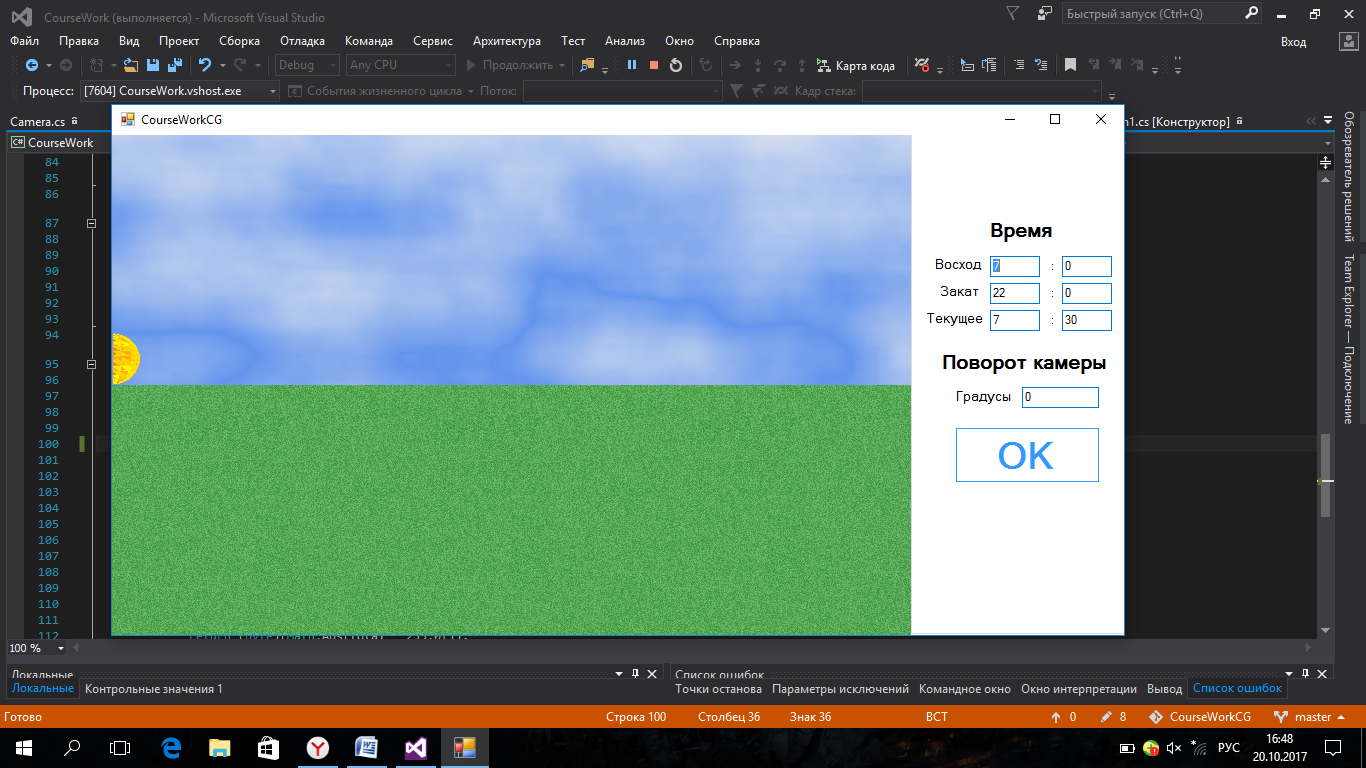
## 3.3 Интерфейс программы

Для запуска программы необходимо выполнить двойной щелчок по файлу приложения. После этого откроется окно программы, представленное на рисунке 15.



*Рис.15 Пример работы программы.*

Меню «Управление» программой располагается на рисунке 15 справа. На рисунке 16 оно представлено в увеличенном масштабе.



*Рис. 16 Меню программы*

Интерфейс программы состоит всего из одного раздела, который соответственно подразделяется на два подраздела: Время и Поворот камеры.

**Подраздел «Время»**

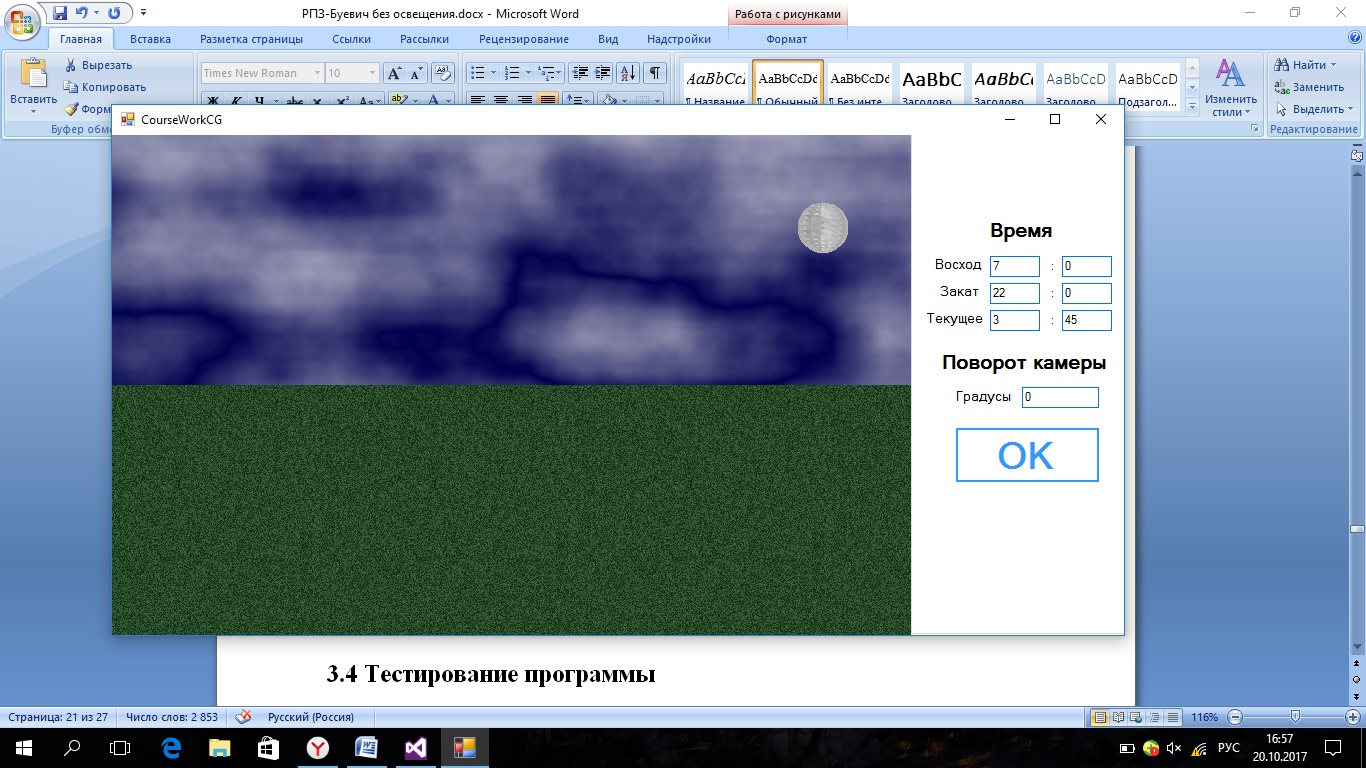
|  |  |
| --- | --- |
| Параметр | Функция |
| **Восход** | Позволяет изменять время восхода солнца. |
| **Закат** | Позволяет изменять время заката солнца. |
| **Текущее время** | Задает текущее время. |

**Подраздел «Поворот камеры»**

|  |  |
| --- | --- |
| Параметр | Функционал |
| **Градусы** | Задает угол поворота камеры. Отрицательное значение – поворот влево. Положительное – вправо. |

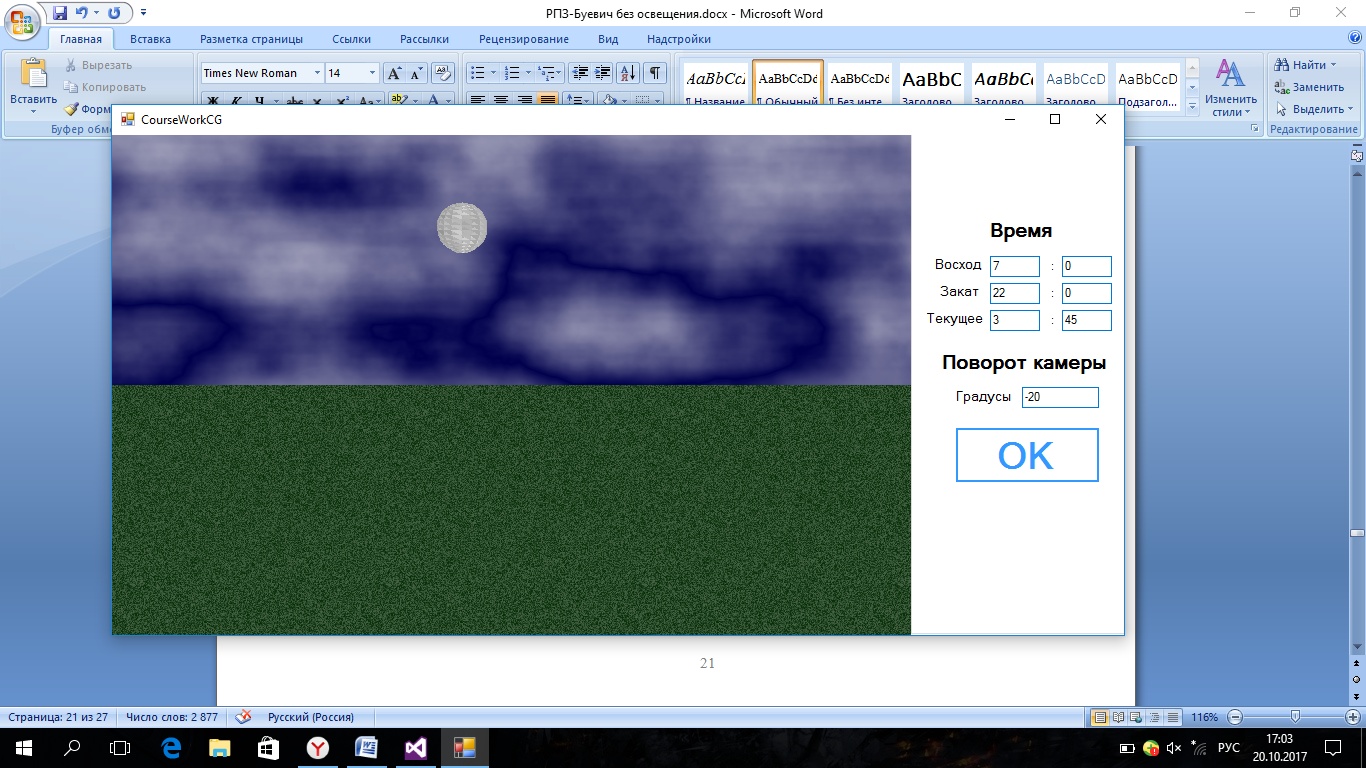
|  |  |
| --- | --- |
| Кнопка | Функционал |
| **ОК** | Применение изменений из подразделов: **«**Поворот камеры**»** и **«**Время**»** |

На рис. 17 представлен пример работы программы, с измененным текущим временем.

**

*Рис. 17 Пример работы программы. (Ночь)*

На рис. 18 представлен пример работы программы, с углом просмотра, измененным на 20 градусов влево.



*Рис. 18 Пример работы программы.*

# 4.Исследовательский раздел

Была измерена скорость вычислений в представленной программе в зависимости от размера текстуры и количества октав. Так как качество модели облака зависит от функции интерполирования и количества октав, было рассмотрено, при каких параметрах наблюдается более реалистичная модель, которая не сильно влияет на время работы, при ее реализации. Поскольку, чтобы сделать динамические облака, приходится постоянно регенерировать модель в определенный период времени.

Для проведения замеров времени использовались следующие параметры:

* Размер текстуры: 256х256, 128х128, 32х32
* Количество октав: 6, 10, 20, 50
* Количество слоев: 4

Результаты измерений времени представлены в секундах:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Октавы  Размер текстуры | 6 | 10 | 20 | 50 |
| 256х256 | 2.962682 | 7.6006152 | 14.4842088 | 20.8954943 |
| 128х128 | 0.70596496 | 1.0252610 | 2.5331036 | 5.4589794 |
| 32х32 | 0.486980 | 0.8297427 | 1.567428 | 2.812144 |

Представлен график, полученный с помощью этих данных:

*Рис. 19 График сравнения быстродействия функции генерации текстур различных размеров.*

Исходя из полученных данных, следует вывод, что использование огромной текстуры для моделирования нескольких облаков не выгодно. Достаточно использовать несколько текстур и скорость генерации увеличиться примерно в два раза.

Технические характеристики ЭВМ, на которой были проведены тесты:

* Процессор: Intel® Core™ i5-3230M CPU @ 2.60GHz 2.60GHz
* ОЗУ: 8,00 Гб DDR3
* Интегрированный графический процессор: Intel® HDGraphics 4000

# Заключение

При написании курсового проекта были рассмотрены и проанализированы основные алгоритмы построения реалистичного трехмерного изображения. Проанализированы их достоинства и недостатки, и возможность их использования для решения поставленной задачи. Для решения поставленной задачи был выбран алгоритм (шума Перлина). Выбранные алгоритмы были разработаны. Найдены оптимальные исходные данные для наилучшего соотношения быстродействия и детализации.

Разработанная программа позволяет получать на экране дисплея реалистичную модель облака в разное время суток. Текущее время, так же как и время восхода и заката можно изменить через пользовательский интерфейс

Приложение имеет несколько направлений дальнейшего развития. Например, на данном этапе не была реализована возможность движения модели облаков.

# Список использованной литературы

1. Две реализации системы итерируемых функций [Электронный ресурс]. URL:<http://100byte.ru/mxscrptxmpls/ifs/ifs.html>
2. Т. Тоффоли, Н. Марглоуса, Машины клеточных автоматов.
3. Визуализация природных явлений: [Электронный ресурс]. URL: <http://www.graphicon.ru/oldgr/courses/cg02b/assigns/hw-5/hw5_cld.htm>
4. Реализация процедурных текстур методом Шума Перлина [Электронный ресурс]. URL:<http://www.gamedev.ru/code/articles/?id=4212>
5. Improved Noise reference implementation [Электронныйресурс]. URL: <http://mrl.nyu.edu/~perlin/noise/>
6. Статья JamesLong [Электронный ресурс]. URL: <http://www.animeimaging.com/asp/PerlinNoise.aspx>
7. Тексель [Электронный ресурс]. URL:<http://www.gamedev.ru/code/terms/Texel>
8. Spherical texture mapping [Электронныйресурс]. URL: <https://www.mvps.org/directx/articles/spheremap.htm>
9. UV mapping a sphere triangle mesh [Электронныйресурс]. URL: <http://mft-dev.dk/uv-mapping-sphere/>
10. Наложение текстуры на объекты. [Электронный ресурс]. URL: <http://www.gamedev.ru/code/forum/?id=125108>
11. Сафонова А.В., «Моделирование реалистичных изображений облаков» //Молодежный научно-технический вестник
12. Основы 3D [Электронный ресурс] URL:<http://www.pmg.org.ru/basic3d/index.html>