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 выпадения осадков»

Студент _____ИУ7-55Б_____ _____ _____С. А. Мирзоян_____
(Группа) (Подпись, дата) (И.О.Фамилия)

Руководитель курсового проекта _____ _____А. А. Павельев_____
(Подпись, дата) (И.О.Фамилия)

2019 г.

Оглавление

Введение 3
1. Аналитическая часть 4
1.1 Формализация объектов синтезируемой сцены 4
1.2 Анализ алгоритмов удаления невидимых линий и поверхностей 5
1.2.1 Алгоритм обратной трассировки лучей 6
1.2.2 Алгоритм, использующий Z буфер 6
1.2.3 Алгоритм Робертса 7
1.2.4 Алгоритм Варнока 7
1.2.5 Вывод 7
1.3 Анализ методов закрашивания 8
1.4 Анализ алгоритмов построения теней 8
1.5 Анализ алгоритмов моделирования осадков 9
1.6 Описание трехмерных преобразований сцены 10
1.7 Выводы из аналитического раздела 11
2. Конструкторская часть 12
2.1 Требования к программе 12
2.2 Общий алгоритм визуализации трехмерной сцены 12
2.3 Алгоритм Z-буфера 12
2.4 Простой метод освещения 13
2.5 Генерация осадков 13
2.6 Выбор используемых типов и структур данных 13
3. Технологическая часть 15
3.1 Выбор и обоснование языка программирования и среды разработки 15
3.2 Структура и состав классов 15
3.3 Сведения о модулях программы 17
3.4 Интерфейс программы 17
4. Экспериментальная часть 19
4.1 Цель эксперимента 19
4.2 Апробация 19
4.3 Описание эксперимента 26
Заключение 31
Список использованной литературы 32

Введение

Трудно переоценить значимость машинной графики, ибо она используется повсеместно: для удобного отображения данных, в компьютерных играх, в кино для создания эффектов, проектирования и моделирования различных конструкций. Перед разработчиками, создающими трехмерные сцены, встает задача создания реалистичных изображений, которые будут учитывать такие физические явления как преломление, отражение и рассеивание света. Для создания еще более реалистичного изображения учитывается дифракция, интерференция, вторичные отражения света, тени.

Существует множество алгоритмов машинной графики, которые решают эту задачу. Чаще всего эти алгоритмы затратные по ресурсам: чем более качественное изображение требуется получить, тем больше времени и памяти тратится на его синтез. Это становится проблемой при создании динамической сцены, где на каждом временном интервале необходимо производить повторные расчеты.

Цель данной работы – реализовать построение трехмерной сцены и визуализацию выпадения осадков на сцене, изображающей парк.

Чтобы достигнуть поставленной цели, требуется решить следующие задачи:

1) описать структуру трехмерной сцены, включая объекты, из которых состоит сцена, и дать описание выбранных погодных явлений, которые будут визуализированы;

2) выбор алгоритмов машинной графики, которые позволят визуализировать трехмерную сцену;

3) реализация данных алгоритмов для создания трехмерной сцены;

4) разработать программное обеспечение, которое позволит отобразить трехмерную сцену и визуализировать погодные эффекты.

1. Аналитическая часть

1.1 Формализация объектов синтезируемой сцены

Сцена состоит из следующих объектов.

- Источника света – вектор направления света, предполагается, что источник расположен в бесконечности. Цвет свечения описывается через перечисление их шестнадцатеричных значений.
- Плоскость земли – ограничивающая плоскость, предполагается, что под плоскостью земли не расположено объектов.
- Объекты сцены – представлены набором следующих объектов:
    - Деревья
    - Скамейка

      Каждый из объектов записан в файле с расширением obj, где описаны вершины и поверхности объектов
- Осадки – частички с заданной траекторией движения, цветом.

  Задаются интенсивностью и направлением движения.
    - Траектория движения – прямая. Ax + By + C = 0
    - Интенсивность осадков – количество частичек, скорость выпадения.

Объекты сцены наилучшим образом описываются через поверхностные модели. Т.к. каркасные модели не дадут реалистичное изображение, которого нужно достичь, а объемная модель будет излишеством, более затратным по памяти, чем поверхностная.

Поверхностную модель можно задать несколькими способами [1].

Параметрическим представлением – для получения поверхности нужно вычислять функцию, зависящую от параметра. Так как в сцене нет никаких поверхностей вращения, использование параметрического представления будет затруднительно.

Полигональной сеткой – совокупностью вершин, ребер и граней, которые определяют форму объекта [2].

- Вершинное представление (вершины указывают на другие вершины, с которыми они соединены). Для генерации списка граней для рендеринга нужно обойти все данные, что затрудняет работу с ним.
- Список граней представляет объект как множество граней и вершин.
- Таблица углов (веер треугольников) – хранит вершины в предопределенной таблице. Не подойдет для моей задачи, так как изменение данного представления затратно по времени.

Наиболее удобным способом хранения моей сцены является список граней т.к. данные в нем можно эффективно преобразовывать, представление позволяет явный поиск вершин грани и граней, окружающих вершину.

1.2 Анализ алгоритмов удаления невидимых линий и поверхностей

При выборе алгоритма удаления невидимых линий нужно будет учесть особенности поставленной задачи. Речь идет о наличии частичек осадков, которые будут перемещаться. Ввиду этого нюанса алгоритм, который будет использоваться, должен работать быстро, иначе осадки будут выглядеть как набор кадров, а не как плавная анимация.

Чтобы избавиться от данного явления можно добавить предзагрузку кадров анимации движения осадков и показывать их только после полного рендеринга анимации. Для сокращения времени расчетов нужно будет рендерить только часть анимации, а потом повторять ее много раз. Однако при таком подходе нужно будет учесть один фактор: первый кадр созданного блока анимации должен совпадать с последним кадром созданного блока анимации. Иначе будут видны места «склейки» двух блоков.

Также ввиду особенностей сцены (деревья и скамья) имеет смысл синтезировать сначала сцену сквера, а потом добавлять осадки поверх нее.

Рассмотрим ключевые алгоритмы построения трехмерной сцены [6].

1.2.1 Алгоритм обратной трассировки лучей

В этом алгоритме за счет скорости работы достигается излишняя, для заданной сцены, универсальность. Обратная трассировка позволяет работать с несколькими источниками света, передавать множество разных оптических явлений [3].

Для создания реалистичного изображения, по правилам обратной трассировки, каждую частицу в осадках (капля дождя, снежинка) нужно будет рассматривать, как отдельный объект сцены, внутри которого могут возникнуть явления дисперсии, преломления и внутреннего отражения.

Положительной стороной данного алгоритма является возможность использования в параллельных вычислительных системах (т.к. расчет отдельной точки выполняется независимо от других точек), так же такая система синтезирует высоко реалистичное изображение.

Серьезным недостатком этого алгоритма является большое количество необходимых вычислений для синтеза изображения сцены.

Вывод: так как в заданной сцене не подчеркиваются явления преломления и отражения света, использование алгоритмов трассировки будет излишним. При заметном замедлении работы программы, качество изображения заметно не улучшится.

1.2.2 Алгоритм, использующий Z буфер

Несомненным плюсом данного алгоритма может являться его простота, которая не мешает решению задачи удаления поверхностей и визуализации их пересечения. В этом алгоритме не тратится время на сортировку элементов сцены, что дает преимущество в скорости работы. Особенно полезным это может стать при большом количестве объектов на сцене.

Так как размер синтезируемого изображения сравнительно мал, затраты по памяти, при хранении информации о каждом пикселе, в данном алгоритме незначительны для современных компьютеров.

1.2.3 Алгоритм Робертса

Серьезным недостатком является вычислительная трудоемкость алгоритма. В теории она растет как квадрат количества объектов. Поэтому при большом количестве объектов на сцене, этот алгоритм будет заметно замедлять анимацию. Можно использовать разные оптимизации для повышения эффективности, например, сортировку по оси Z.

Преимуществом данного алгоритма является точность вычислений. Она достигается за счет работы в объектном пространстве, в отличии от большинства других алгоритмов.

Некоторые из оптимизаций крайне сложны, что затрудняет реализацию этого алгоритма.

1.2.4 Алгоритм Варнока

Алгоритм Варнока основывается на рекурсивном разбиении экрана. В зависимости от расположения объектов это может стать, как положительной, так и отрицательной стороной алгоритма. Чем меньше пересечений объектов, тем быстрее алгоритм завершит свою работу.

1.2.5 Вывод

Алгоритм метода трассировки лучей не позволит в реальном времени смоделировать туман из-за рассеивания света. Можно лишь его сымитировать используя зависимость затуманенности от пройденного расстояния луча. Поэтому предпочтительнее использовать Z буфер для динамической сцены визуализации погоды, т.к. важна скорость работы алгоритма. На его основе будет просто визуализировать туман и дождь. В случае тумана можно хранить глубину пикселя, с помощью которой можно будет вычислить насколько «затуманенным» он будет. В случае осадков, можно накладывать осадки на уже посчитанный Z буфер сцены, не проводя повторных расчетов.

1.3 Анализ методов закрашивания

Простая закраска

Вся грань закрашивается одним уровнем интенсивности, который высчитывается по закону Ламберта.

Этот метод крайне прост в реализации и совершенно не требователен к ресурсам. Однако плохо подходит для тел вращения, плохо учитывает отраженный свет. Для моей задачи этот метод очень хорошо подходит, так как вся работа ведется с гранями зданий, тел вращения нет.

Закраска по Гуро

Основа закраски по Гуро – билинейная интерполяция интенсивностей, за счет которой устраняется дискретность изменения интенсивности и создается иллюзия гладкой криволинейной поверхности. Хорошо сочетается с диффузным отражением.

Закраска по Фонгу

Основа закраски по Фонгу – билинейная интерполяция векторов нормалей. Достигается лучшая локальная аппроксимация кривизны поверхности. Изображение выходит более реалистичным, зеркальные блики выглядят правдоподобнее, чем в методе закраски по Гуро.

Однако по сравнению с методом Гуро, закраска по Фонгу требует больших вычислительных затрат, так как интерполируются значения векторов нормалей, на основе которых потом вычисляется интенсивность.

Вывод: так как фигуры сцены состоят из плоскостей закраска по Фонгу и Гуро будет скорее мешать: ребра зданий будут сглажены. Тело здания будет хуже восприниматься. Поэтому лучше всего использовать простую закраску.

1.4 Анализ алгоритмов построения теней

При трассировке лучей тени получаются без дополнительных вычислений. Пиксел затенен, когда луч попадает на объект и позже не попадает ни в объект,

ни в источник света. Так как в анализе алгоритмов трассировка лучей не была выбрана в качестве алгоритма синтеза сцены, то тени нужно вычислять отдельно.

Один из способов нахождения теней – вычисление проекций тел. Можно использовать метод теневых карт, в котором предполагается, что освещены только те фрагменты, которые видны из положения источника. Находить видимость можно с помощью алгоритма Робертса, алгоритма Z буфера и других. Для создания теневых карт будет использоваться алгоритм Z буфера так как этот алгоритм позволит быстро найти видимость объектов сцены.

1.5 Анализ алгоритмов моделирования осадков

Реалистичное и эффективное представление сцен с атмосферными осадками крайне непростая задача из-за огромного количества капель дождя / снежинок. Каждая из этих частиц обладает такими свойствами, как: размер, масса, скорость, ускорение.

Система частиц

В системе частиц частицы (капли дождя, снежинки) рассматриваются как материальные точки с разными атрибутами. Сама же система частиц — это совокупность всех частиц явления [4]. Обычно все частицы в системе меняют скорость или размер по общему закону. Различная интенсивность осадков достигается за счет изменения общего количества частиц.

Для ускорения программы предполагают, что частицы не отбрасывают тени и не поглощаются свет. Без этих допущений придется проделать много вычислений.

Главным достоинством данного метода является то, что он позволяет описывать реалистичное поведение осадков, в зависимости от геометрических форм сцены или таких физических явлений как ветер.

Недостатком же является большой объем расчетов, требуемой памяти, а также то, что не существует единого стандарта по реализации системы частиц.

Использование видео

Исследования показывают, что человек не различает отдельных капель при интенсивном дожде, и поэтому можно смешивать заранее подготовленную видеопоследовательность с исходной, чтобы добиться ощущения дождя в сцене. Кроме этого, в работе предлагается применять некоторые фильтры изменения яркости изображения, чтобы плавно управлять плотностью дождя. Все это вполне может быть проделано в реальном времени на современных графических картах, самая дорогая операция – собственно, раскодирование видеопотока. К сожалению, метод не может охватить некоторые важные особенности динамических систем. Например, невозможно изменить направление движения дождя, учесть скорость движения камеры, и вообще позволять камере далеко отклоняться от перпендикулярной направлению капель. Кроме того, такой метод не учитывает геометрию сцены. В целом метод обеспечивает очень хорошее качество, но с серьезными ограничениями на свободу движения камеры и управления свойствами дождя.[5]

Анимация осадков в области экрана.

Является одним из часто используемых методов. В отличии от системы частиц, этот метод работает в двумерном пространстве экрана, ему не требуется хранить координату Z, не требуются сложные расчеты. Однако недостатком такого метода является то, что он не учитывает геометрию сцены. Однако в рамках данного проекта этот метод подходит, ввиду отсутствия сложных геометрических объектов, как навес или крыша.

1.6 Описание трехмерных преобразований сцены

Сдвиг точки:

$$\begin{cases} X = x + dx \\ Y = y + dy \\ Z = z + dz \end{cases}$$

Масштабирование относительно начала координат:

$$\begin{cases} X = x \cdot k_x \\ Y = y \cdot k_y \\ Z = z \cdot k_z \end{cases}$$

Поворот относительно осей x, y, z на угол φ:

$$\begin{cases} X = x \\ Y = y \cdot \cos\phi + z \cdot \sin\phi \\ Z = -y \cdot \sin\phi + z \cdot \cos\phi \end{cases} \quad \begin{cases} X = x \cdot \cos\phi - z \cdot \sin\phi \\ Y = y \\ Z = x\sin\phi + z\cos\phi \end{cases} \quad \begin{cases} X = x \cdot \cos\phi + y \cdot \sin\phi \\ Y = -x \cdot \sin\phi + y \cdot \cos\phi \\ Z = z \end{cases}$$

1.7 Выводы из аналитического раздела

В данном разделе были рассмотрены алгоритмы удаления невидимых линий и поверхностей, методы закрашивания поверхностей, алгоритмы построения теней и моделирования осадков. В качестве алгоритма удаления невидимых линий был выбран Z-буфер, методом закраски – простой, построение теней будет выполняться с помощью теневых карт, построенных алгоритмом Zбуфера, для моделирования осадков была выбрана система частиц.

2. Конструкторская часть

В данном разделе будут рассмотрены требования к программе и алгоритмы визуализации сцены и погодных явлений.

2.1 Требования к программе

Программа должна предоставлять следующие возможности:

- визуальное отображение сцены;
- добавление нового объекта в сцену;
- активация режима снегопада;
- активация режима дождя;
- поворот исходной сцены.

2.2 Общий алгоритм визуализации трехмерной сцены

Визуализации трехмерной сцены сквера с осадками происходит поэтапно. Рассмотрим алгоритм визуализации.

1. Задать объекты сцены
2. Задать источники света и положение наблюдателя
3. Для каждого полигона высчитать нормаль и интенсивность цвета, найти внутренние пиксели
4. Используя алгоритм Z буфера получить изображение сцены, и буфер глубины
5. Используя алгоритм Z буфера получить буфер глубины для источника света
6. Найти затененные участки при помощи наложения двух буферов
7. Если присутствуют осадки, то выполнять пункты 7.1 и 7.2 до тех пор, пока не изменится один из параметров, влияющих на изображение сцены -> переход в 1 (перемещение объектов, вращение камеры).
    7.1 Наложить осадки на полученное изображение
    7.2 Отобразить изображение
8. Обновить данные о положении частиц

Иначе отобразить изображение.

2.3 Алгоритм Z-буфера

1. Всем элементам буфера кадра присвоить фоновое значение
2. Инициализировать Z буфер минимальными значениями глубины

3. Выполнить растровую развертку каждого многоугольника сцены:
    a. Для каждого пикселя, связанного с многоугольником вычислить его глубину z(x, y)
    b. Сравнить глубину пискселя со значением, хранимым в Z буфере. Если z(x, y) > zбуф(x, y), то zбуф(x,y) =z(x,y), цвет(x, y) = цветПикселя.
4. Отобразить результат

2.4 Простой метод освещения

В простом методе освещения интенсивность рассчитывается по закону Ламберта:

$$I = I_0 * \cos(\alpha), \text{ где}$$

I – результирующая интенсивность света в точке

I_0 – интенсивность источника

α – угол между нормалью к поверхности и вектором направления света

2.5 Генерация осадков

1. Инициализация начальных данных (интенсивности осадков)
2. Пока не получена команда прекращения осадков:
    2.1 Обновление положения частиц по заданному закону
    2.2 Инициализация новых частиц
    2.3 Отображение частиц на дисплее

2.6 Выбор используемых типов и структур данных

Для разрабатываемого ПО нужно будет реализовать следующие типы и структуры данных.

- Источник света – вектор направленности света.
- Сцена – задается объектами сцены
- Объекты сцены – задаются вершинами и гранями, грани задаются треугольниками.

- Система частиц – хранит в себе частицы, направление движения
- Математические абстракции
    - Точка – хранит координаты x, y, z
    - Вектор – хранит направление по x, y, z
    - Многоугольник – хранит вершины, нормаль, цвет
- Интерфейс – используются библиотечные классы для предоставления доступа к интерфейсу.

3. Технологическая часть

3.1 Выбор и обоснование языка программирования и среды разработки

В качестве языка программирования был выбран C++ т.к.:

- данный язык программирования объектно-ориентирован, что даст в полной мере
    - использовать наследование, абстрактные классы и т.д.;
    - представлять трехмерные объекты сцены в виде объектов классов, что позволит легко организовать взаимодействие между ними, положительно влияя на читабельность кода.

В качестве среды разработки была выбрана «Visual Studio 2019» по следующим причинам:

- она имеет множество удобств, которые облегчают процесс написания и отладки кода;
- она обеспечивает работу с Windows Forms – интерфейсом, который упрощает доступ к элементам интерфейса Microsoft Windows за счет создания обертки для существующего Win32 API в управляемом коде;

3.2 Структура и состав классов

В этом разделе будут рассмотрена структура и состав классов, см. рис. 3.1.

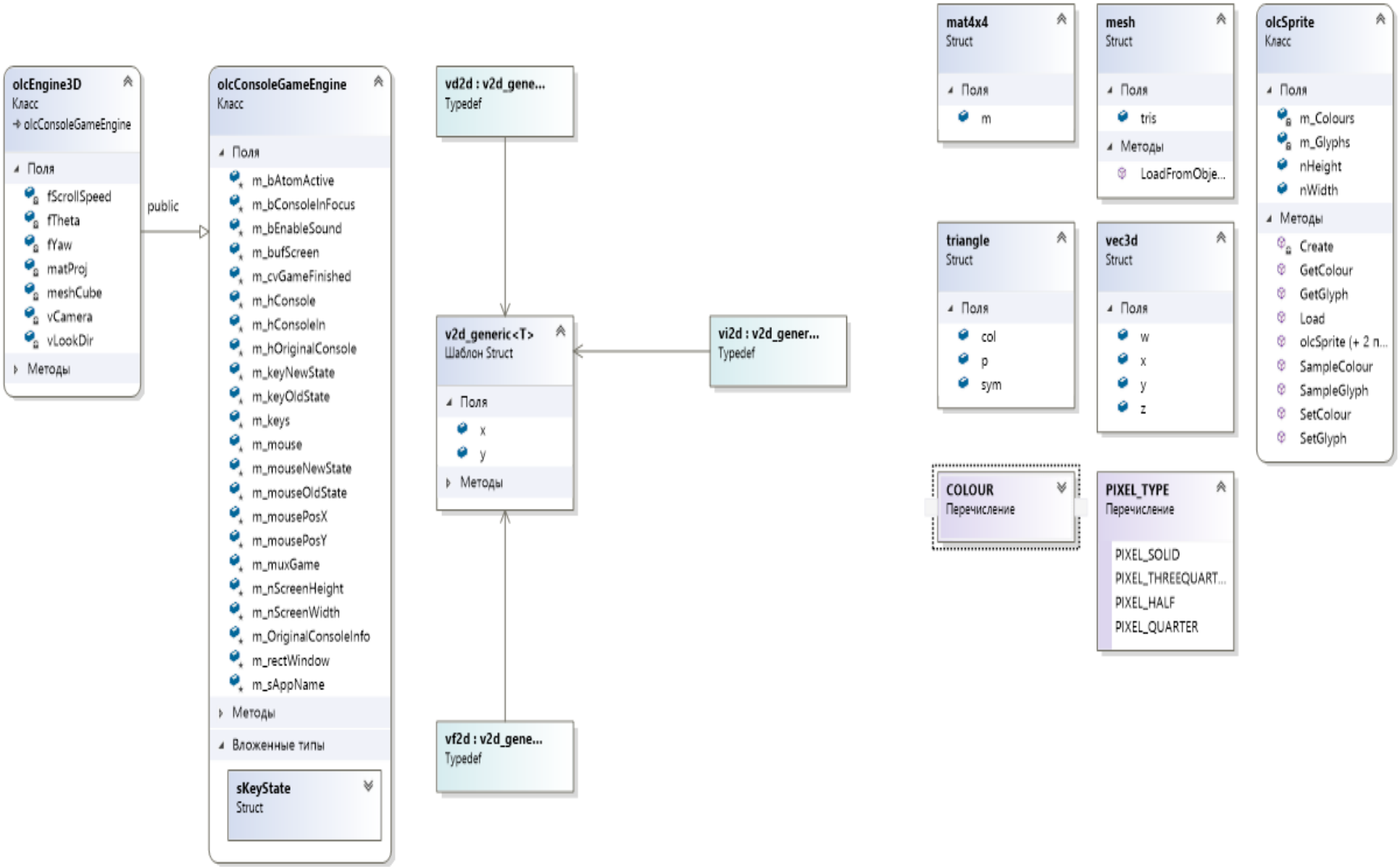

olcEngine3D
Класс
olcConsoleGameEngine
Поля
fScrollSpeed
fTheta
fYaw
matProj
meshCube
vCamera
vLookDir
Методы
public
olcConsoleGameEngine
Класс
Поля
m_bAtomActive
m_bConsoleInFocus
m_bEnableSound
m_bufScreen
m_cvGameFinished
m_hConsole
m_hConsoleIn
m_hOriginalConsole
m_keyNewState
m_keyOldState
m_keys
m_mouse
m_mouseNewState
m_mouseOldState
m_mousePosX
m_mousePosY
m_muxGame
m_nScreenHeight
m_nScreenWidth
m_OriginalConsoleInfo
m_rectWindow
m_sAppName
Методы
Вложенные типы
sKeyState
Struct
vd2d : v2d_gene...
Typedef
v2d_generic<T>
Шаблон Struct
Поля
x
y
Методы
vi2d : v2d_gener...
Typedef
vf2d : v2d_gene...
Typedef
mat4x4
Struct
Поля
m
mesh
Struct
Поля
tris
Методы
LoadFromObje...
olcSprite
Класс
Поля
m_Colours
m_Glyphs
nHeight
nWidth
Методы
Create
GetColour
GetGlyph
Load
olcSprite (+ 2 п...
SampleColour
SampleGlyph
SetColour
SetGlyph
triangle
Struct
Поля
col
p
sym
vec3d
Struct
Поля
w
x
y
z
COLOUR
Перечисление
PIXEL_TYPE
Перечисление
PIXEL_SOLID
PIXEL_THREEQUART...
PIXEL_HALF
PIXEL_QUARTER

Рис. 3.1 Диаграмма классов

Program – входная точка в программу

olcEndine3D – класс пользовательского интерфейса

Mesh – хранит информацию о сцене: размеры, модели внутри сцены, имеет методы создания сцены, ее преобразования;

Model – хранит информацию о модели: ее вершины, индексы вершин для создания многоугольников, многоугольники, имеет методы загрузки и преобразования модели;

Triangle – класс треугольника, хранит цвет, вершины и точки внутри.

Vector – хранит направление вектора и его длину, имеет методы поиска длины вектора, нахождение углов между двумя векторами, векторное и скалярное умножение векторов;

3.3 Сведения о модулях программы

ConsoleApplication2.cs – главная точка входа в приложение;

OlcEngine3D – графический движок, содержащий основные методы построения изображения

3.4 Интерфейс программы

Интерфейс делится на два блока(рис. 3.2 и 3.3).

Блок «Инструкция» - здесь описаны элементы управления:

- W, A, S, D – клавиши, отвечающие за движение камеры вперед, назад, поворот влево и вправо соответственно.
- «Стрелки вверх и вниз» - клавиши, отвечающие за соответствующие перемещения камеры.
- 1, 2, 3 – клавиши режима дождя:
  - 1 – слабая интенсивность
  - 2 – средняя интенсивность
  - 3 – сильная интенсивность

- 4, 5, 6 - клавиши режима снегопада:
  - 1 – слабая интенсивность
  - 2 – средняя интенсивность
  - 3 – сильная интенсивность
- 0 – остановить анимацию выпадения осадков

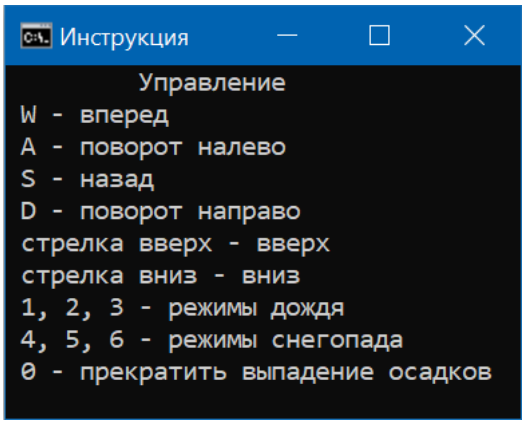


Рис. 3.2 Интерфейс программы – инструкция к управлению

Блок «Программа»

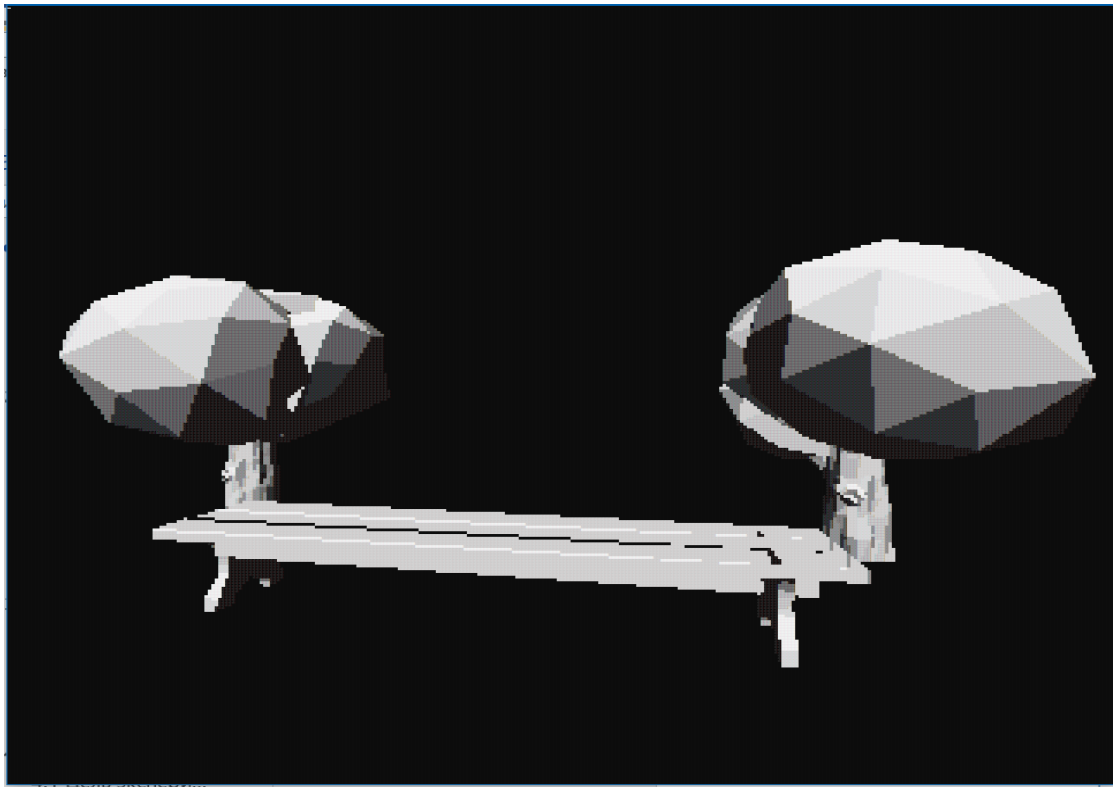


Рис. 3.3 Интерфейс программы

4. Экспериментальная часть

В данном разделе будет проведена апробация реализованной программы для проверки корректности ее работы и поставлен эксперимент по оценки эффективности работы программы.

4.1 Цель эксперимента

Целью эксперимента является проверка правильности выполнения поставленной задачи, оценка эффективности при многопоточной реализации просчета теней.

4.2 Апробация

На рисунках 4.1 и 4.2 показан один и тот же сквер с разных позиций. Можно заметить, что тени отображаются корректно.

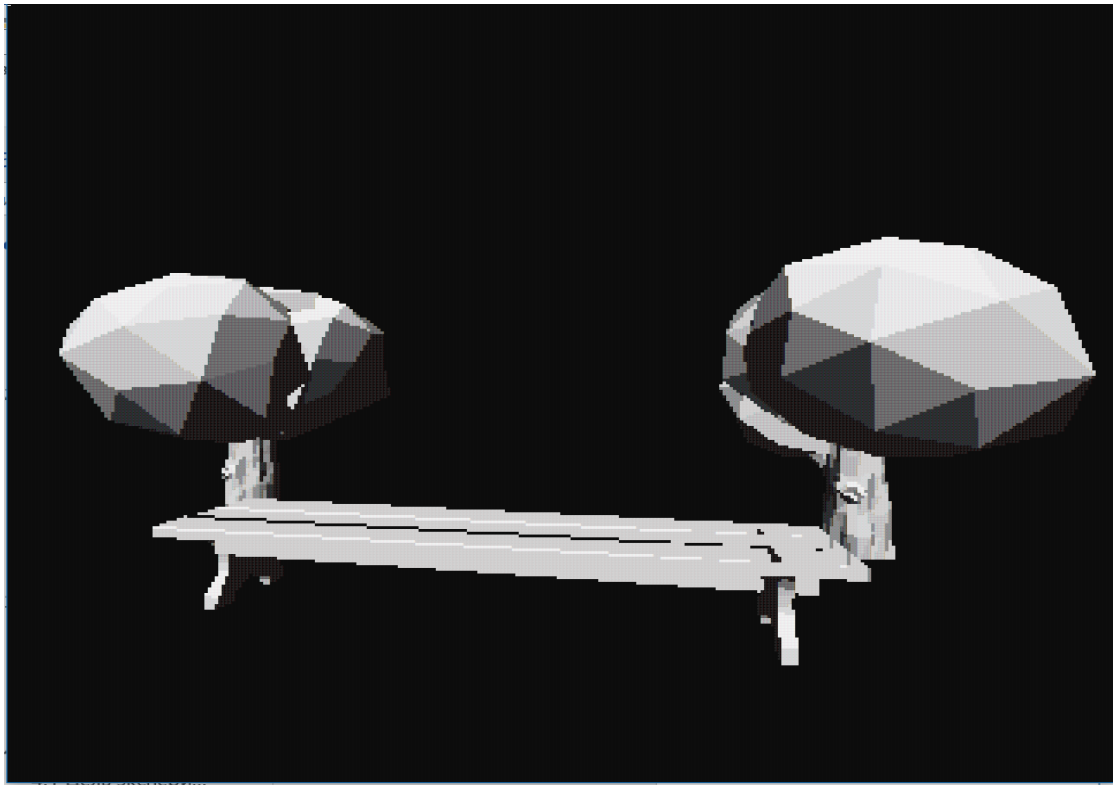


Рис. 4.1 Визуализация сцены, источник света находится за камерой

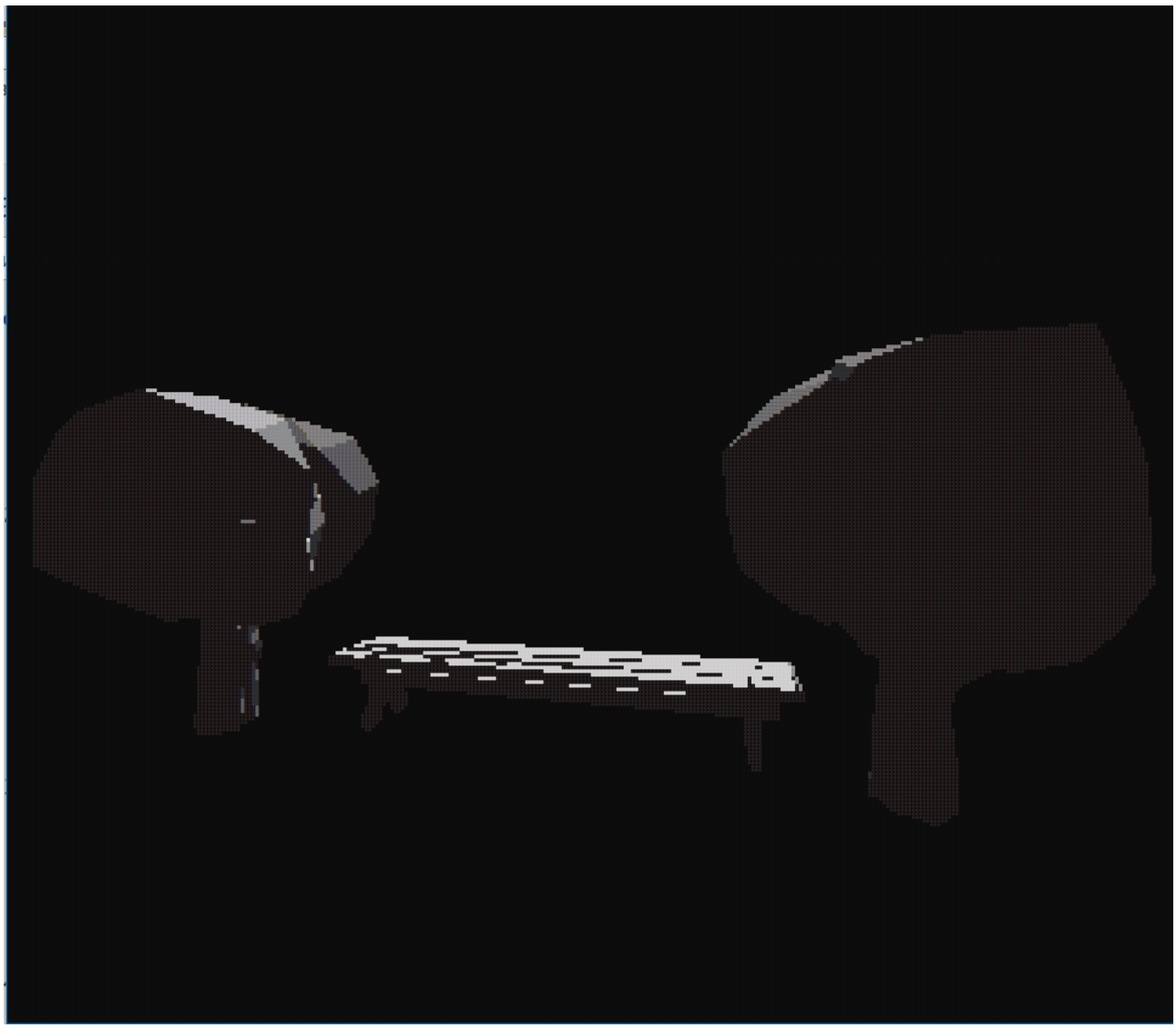


Рис. 4.2 Визуализация сцены, камера направлена в сторону источника света

На рисунках 4.3, 4.4, 4.5 показаны эффекты дождя с разными интенсивностями. Из рисунков видно, что эффект дождя работает корректно.

На рисунках 4.6, 4.7, 4.8 показаны эффекты снегопада с разными интенсивностями. Из рисунков видно, что эффект дождя работает корректно.

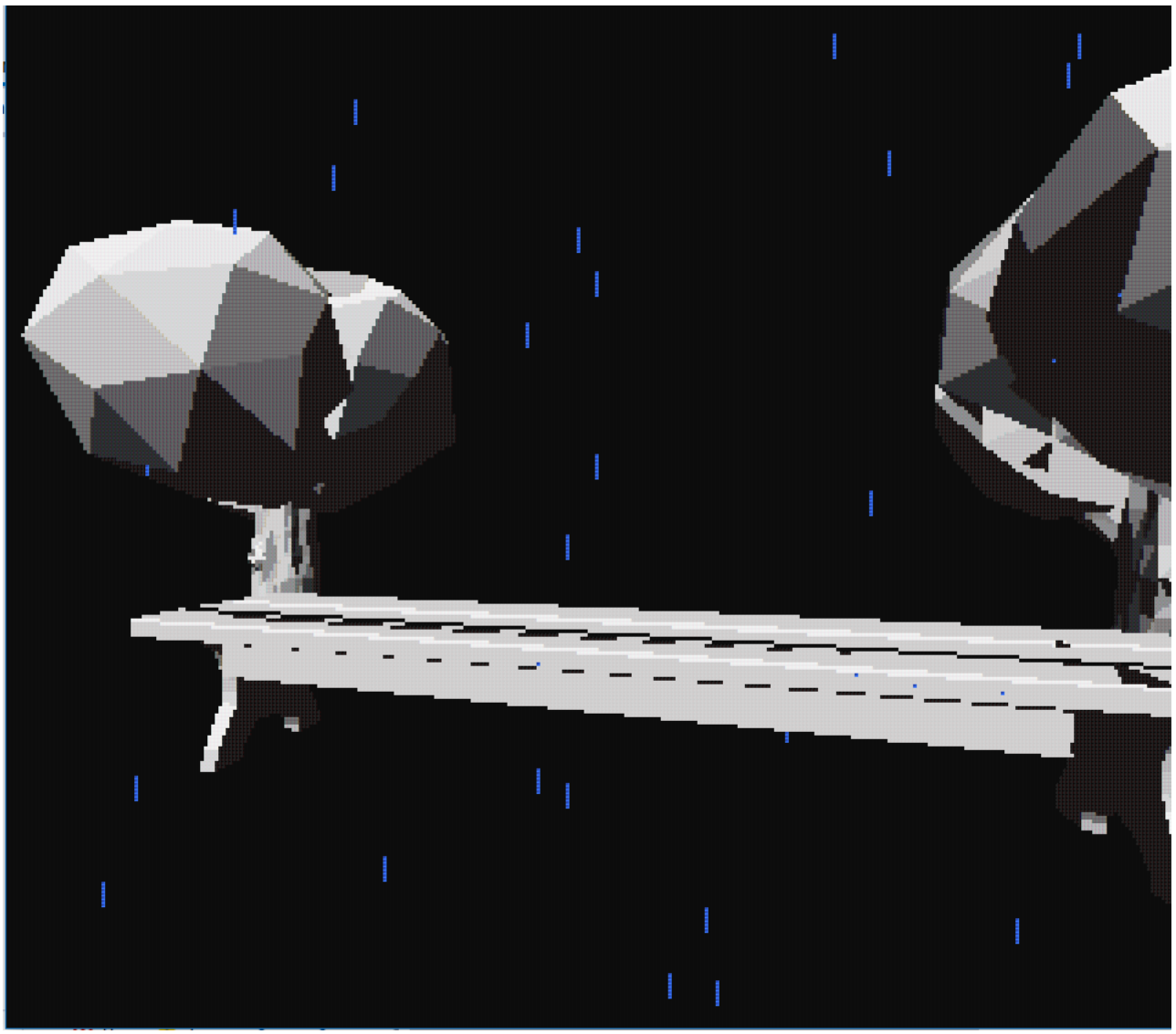


Рис. 4.3 Эффект дождя, слабая интенсивность

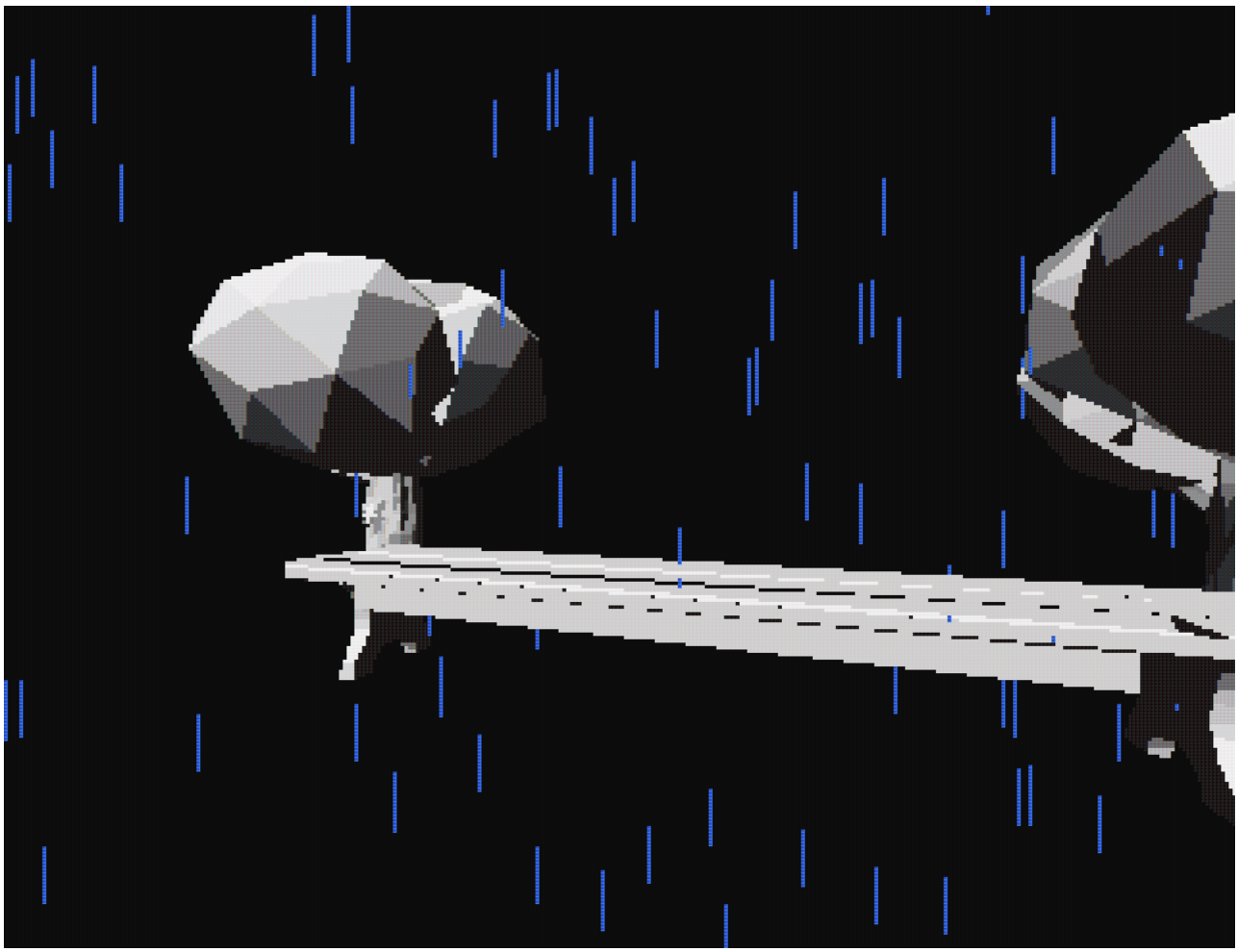


Рис 4.4. Эффект дождя, средняя интенсивность

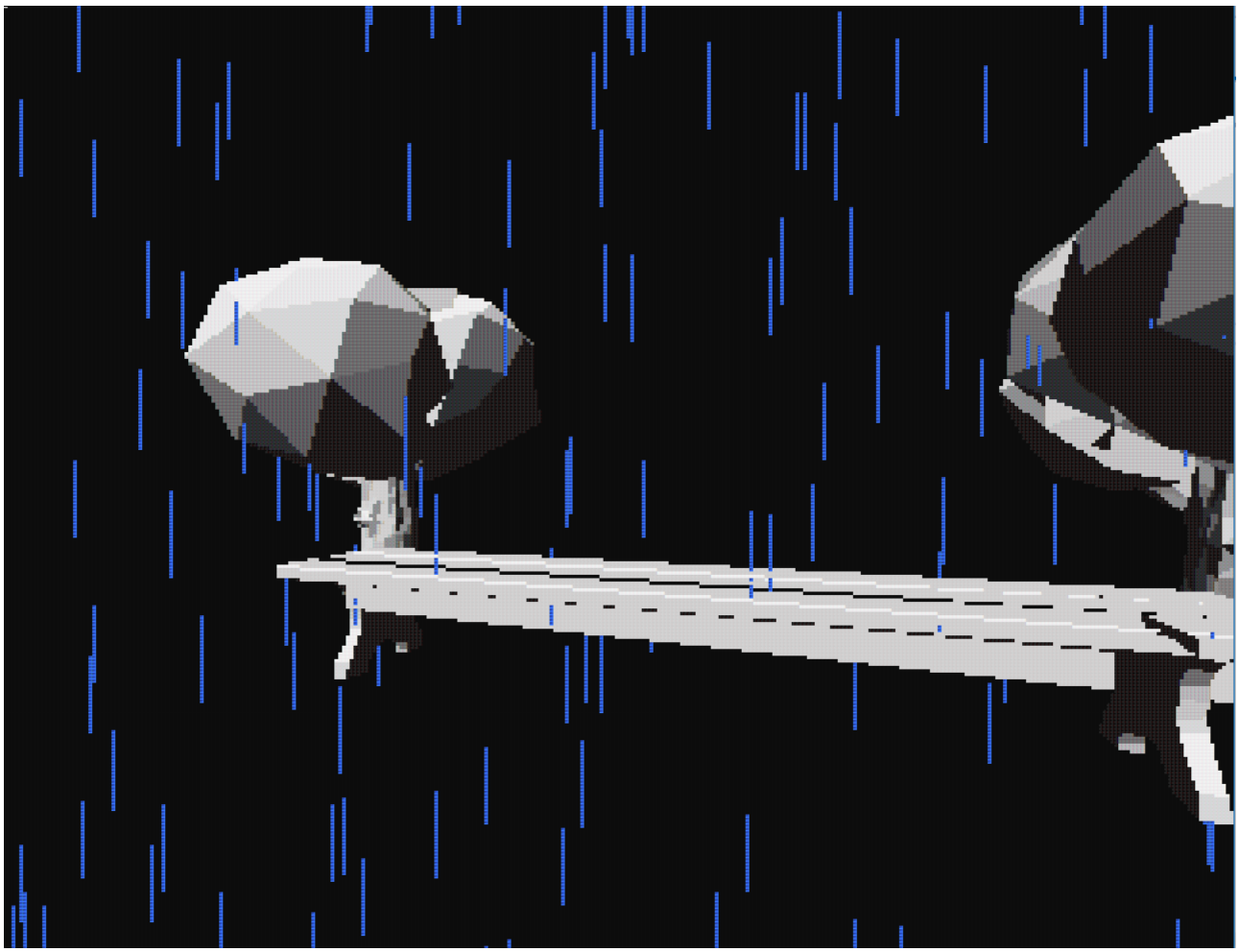


Рис 4.5. Эффект дождя, сильная интенсивность

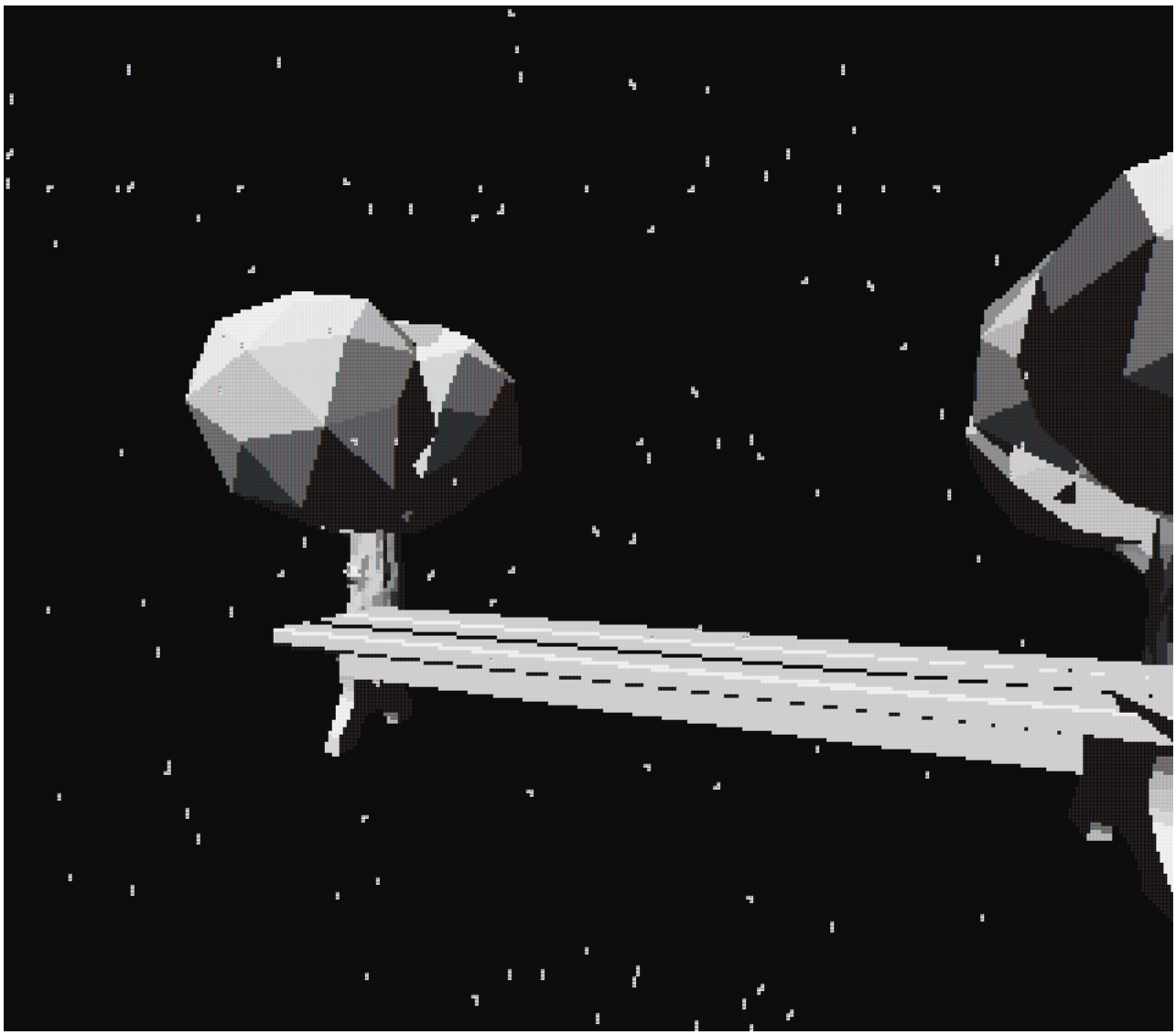


Рис. 4.6 Эффект снегопада, слабая интенсивность

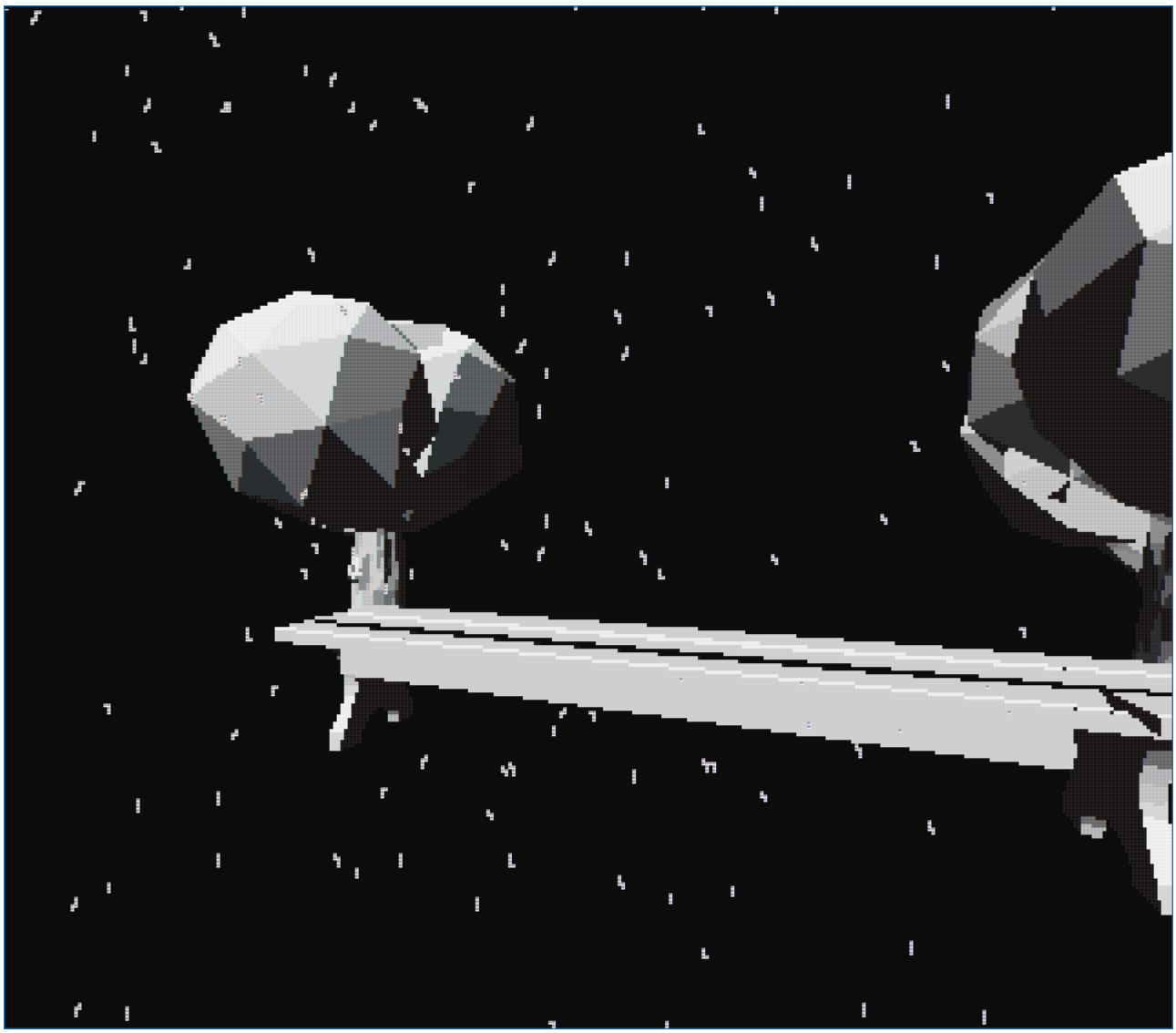


Рис 4.7. Эффект снегопада, средняя интенсивность

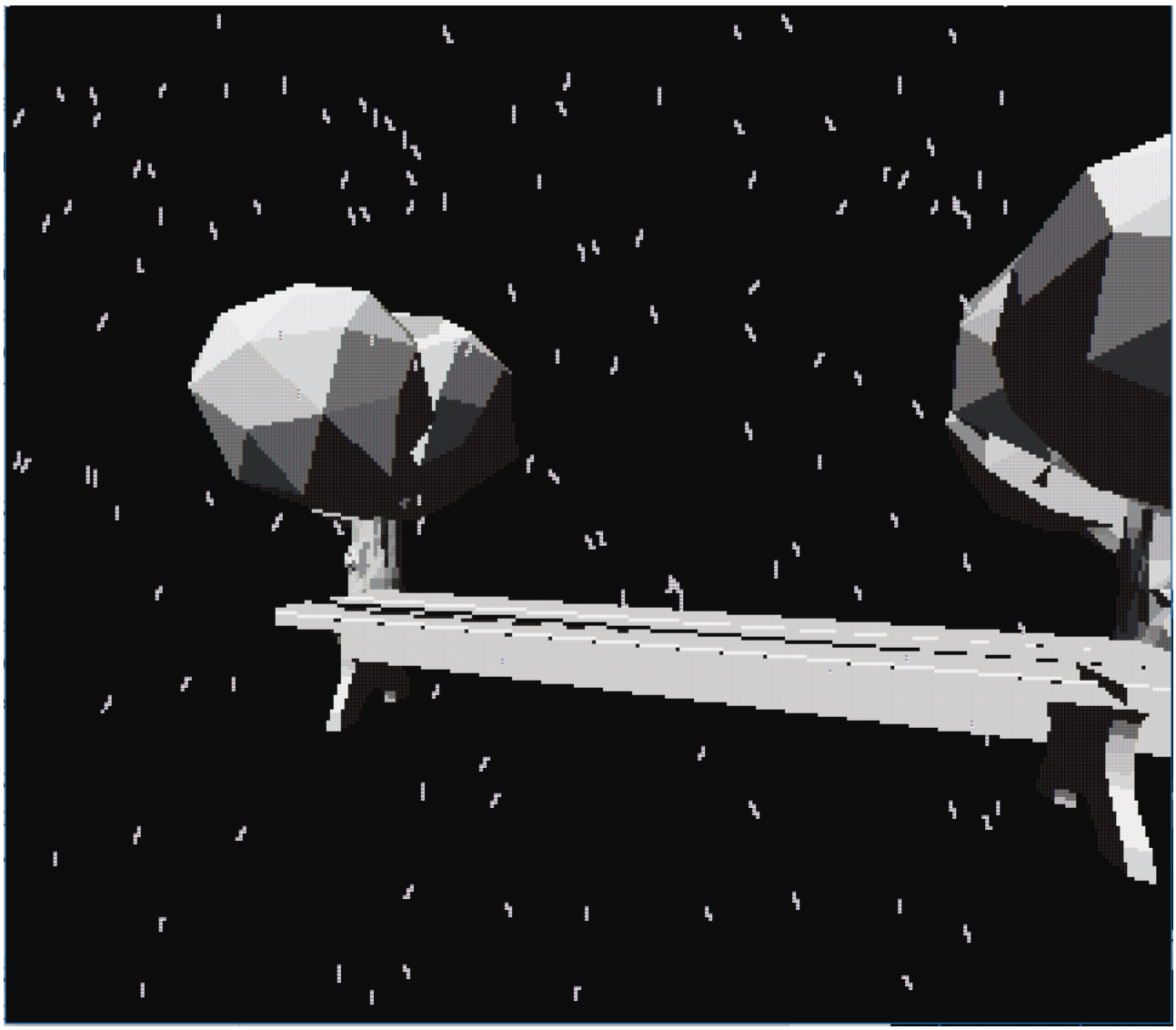


Рис 4.7. Эффект снегопада, сильная интенсивность

4.3 Описание эксперимента

Было построено несколько сцен с разным количеством поверхностей. (Рис. 4.8, 4.9, 4.10)

Рис 4.8 Сцена с большим количеством поверхностей

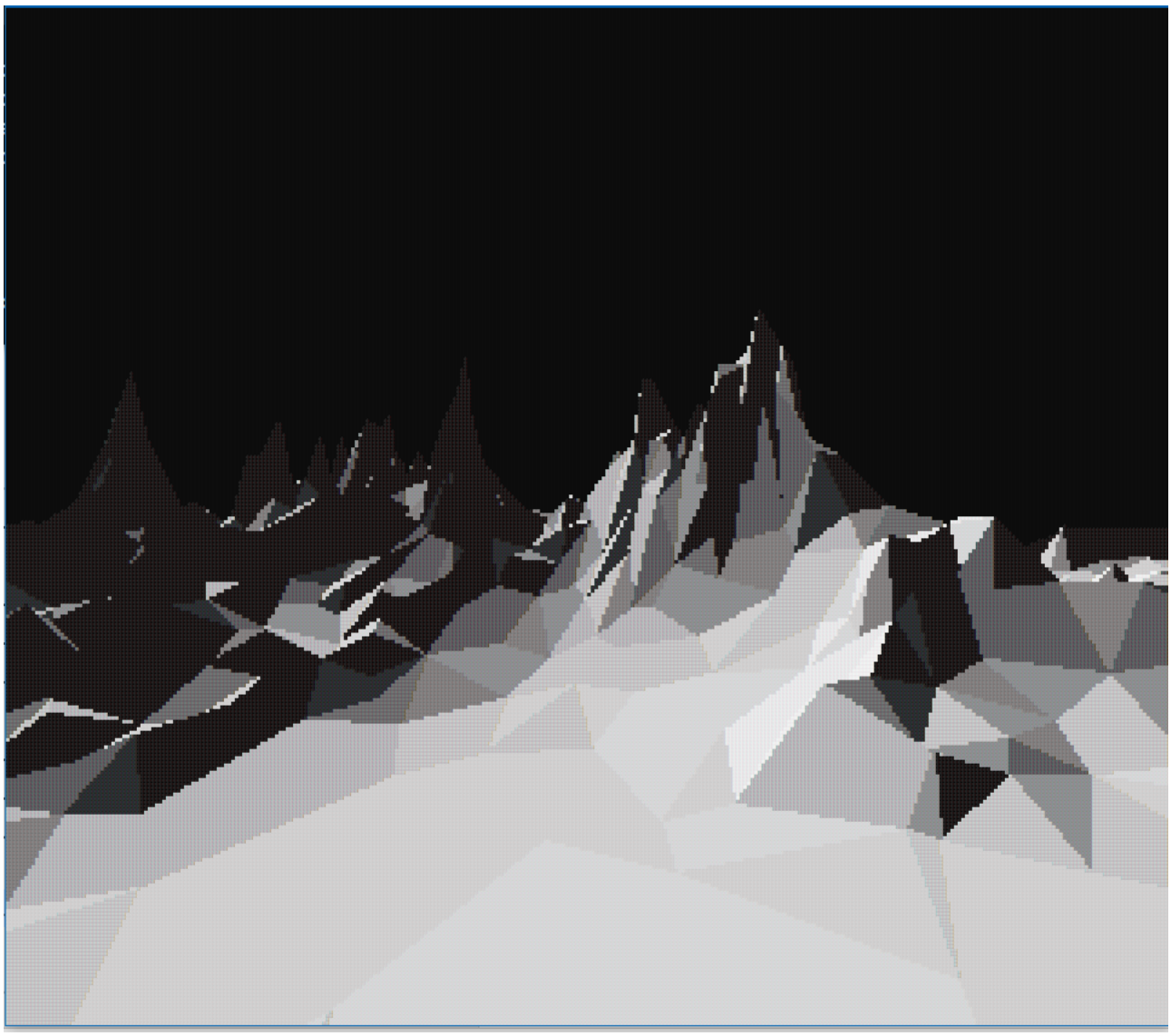


Рис. 4.9 Сцена со средним количеством поверхностей

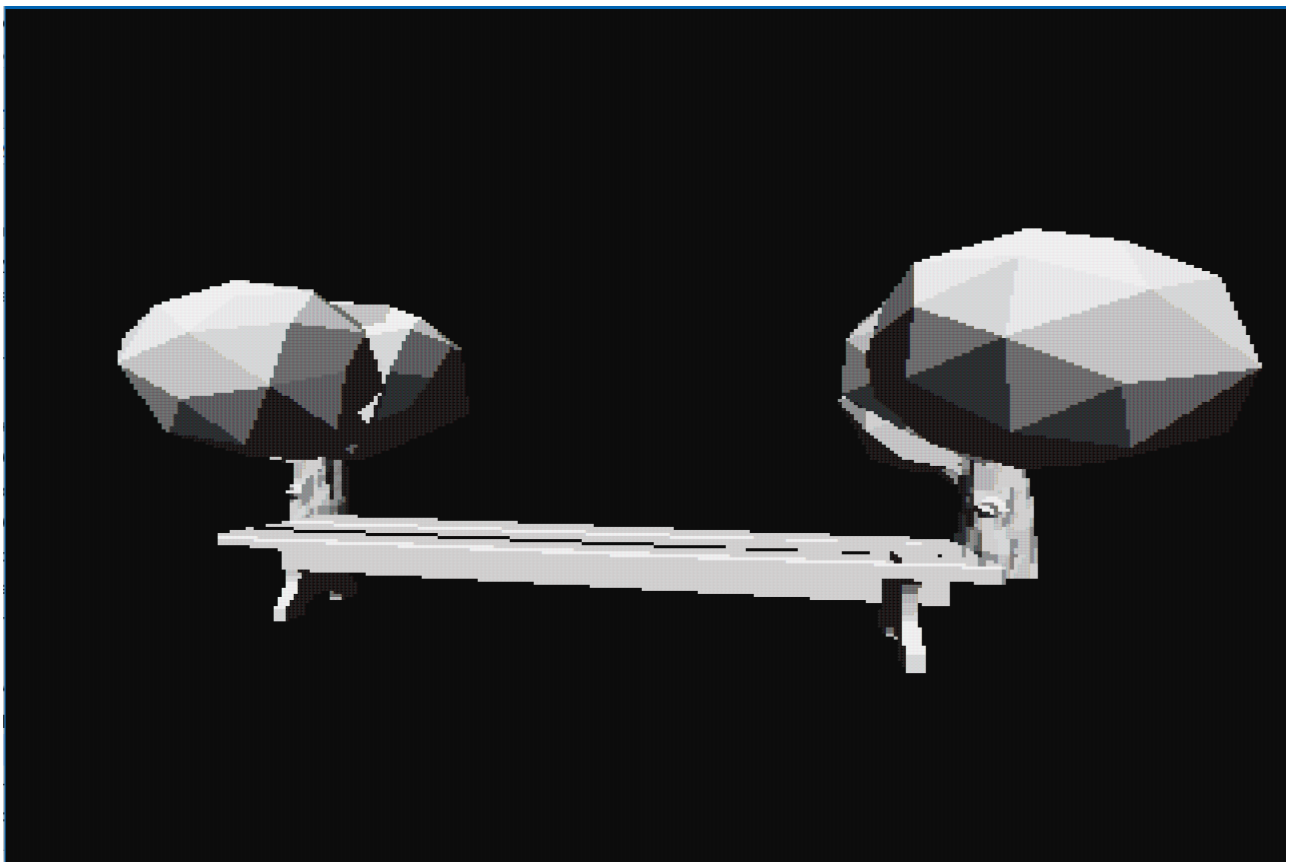
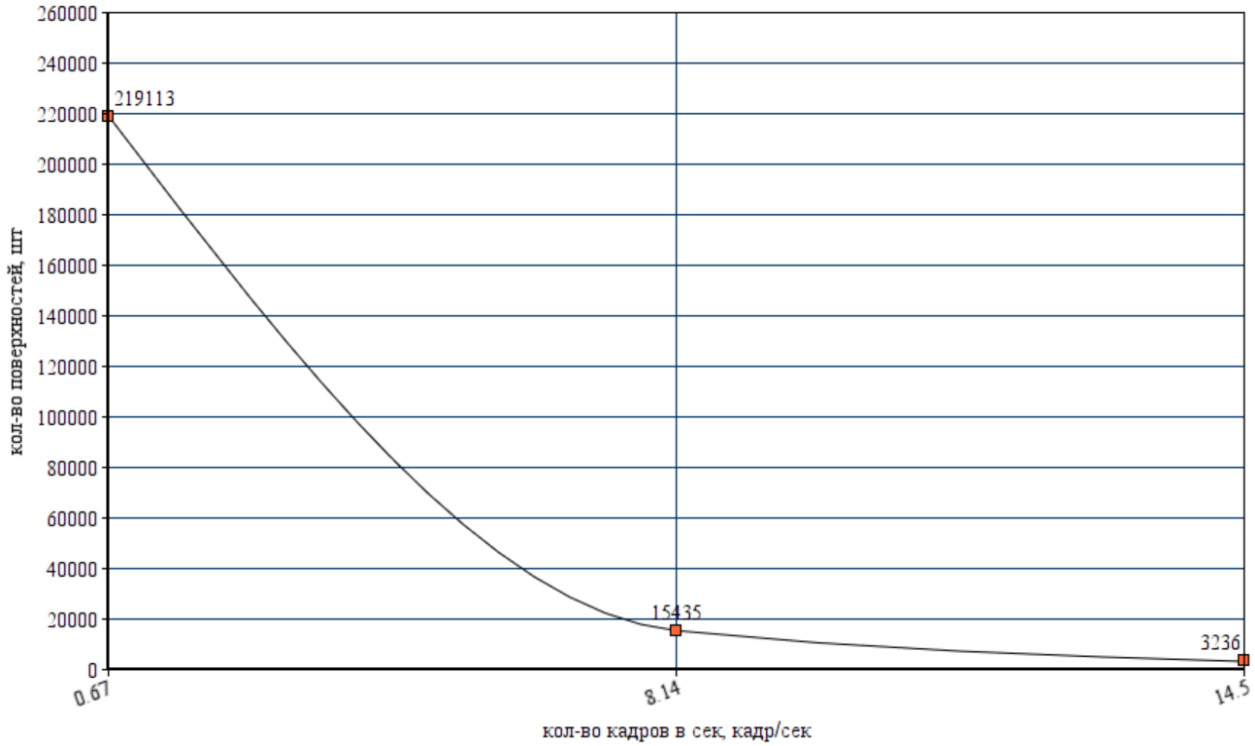


Рис 4.10 Сцена с малым количеством поверхностей

Эксперимент проводился на компьютере со следующими характеристиками:

- Intel® Core™ i5-8250U
- 4 ядра
- 8 логических процессоров
- 8ГБ оперативной памяти

В ходе эксперимента было выявлена следующая закономерность: количество отображаемых кадров в секунду обратно пропорционально количеству поверхностей, из которых состоит сцена. Данную закономерность можно наблюдать на следующем графике:


260000
240000
220000
200000
180000
160000
140000
120000
100000
80000
60000
40000
20000
0
кол-во поверхностей, шт
219113
15435
3236
0.67
8.14
14.5
кол-во кадров в сек, кадр/сек

Заключение

Во время выполнения курсового проекта была описана структура трехмерной сцены, были рассмотрены основные алгоритмы машинной графики, такие как удаление невидимых линий, методы, методы закрашивания, методы генерации осадков. Были проанализированы их достоинства и недостатки, выбраны и реализованы наиболее подходящие для решения поставленной задачи. Было разработано программное обеспечение для визуализации сцены и погодных явлений.

Программа реализована таким образом, что пользователь изменять режим выпадения осадков и их вид.

В ходе выполнения поставленной задачи были изучены возможности Windows Forms, получены знания в области машинной графики.

В ходе выполнения экспериментальной части, была установлена зависимость количества отображаемых кадров в секунду от количества поверхностей объектов, из которых состоит сцена.

Список использованной литературы

1. Методы представления дискретных данных [Электронный ресурс]. – Режим доступа: https://www.graphicon.ru/oldgr/ru/library/multires_rep/index.html (дата обращения 27.06.19)
2. Bruce Baumgart, Winged-Edge Polyhedron Representation for Computer Vision. National Computer Conference, May 1975
3. Е. А. Снижко. Компьютерная геометрия и графика [Текст], 2005. - 17 с.
4. Визуализация эффекта дождя в автомобильных тренажерах; С.М.Козлов, Н.А.Елыков, И.В.Белаго, 2006.
5. Starik S., Werman M., ”Simulation of Rain in Videos*”, 2003*
6. D. F. Rogers. Procedural Elements for Computer Graphics. 2nd ed., 1998 – p.457-517