МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ
(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)
Институт №8 «Информационные технологии и прикладная математика»
Кафедра 806 «Вычислительная математика и программирование»

Лабораторные работы
по курсу «Методы, средства и технологии мультимедиа»

Выполнила: Алексюнина Ю.В.

Группа: М8О-407

Преподаватель: А.В. Крапивенко

Москва, 2020

Апробация пакета 3D Studio MAX

1. Цели

Ознакомление с основными возможностями пакета 3Ds MAX (или аналогичного, по своему усмотрению), создание геометрических примитивов, изучение методов редактирования объектов, создание и модификация физических свойств материалов, установка камер и источников света, визуализация сцены.

2. Задание

Смоделировать текстурированный 3D объект корпусной мебели, на смоделированном объекте расположить самостоятельно сконструированное тело вращения, предусмотреть его прозрачность и тени. Поместить сцену в замкнутое пространство (например, комнату), добавить несколько произвольных предметов обстановки. Произвести рендеринг с учетом отражений. При этом точку обзора камеры необходимо композиционно выбирать так, чтобы все требования к объектам сцены были видны сразу, на одном изображении в приемлемом масштабе.

3. ПО

3ds MAX 2016

4. Теория

3D-моделирование.

Моделирование 3D-сцен(способы):

1. Полигональный(триангулярный) – аппроксимация треугольниками(3D)/полигонами(2D) – самый простой и емкий
2. Функционально-параметрический – если моделируются гладкие поверхности – применяется технология сплайнов
3. Воксельный(метод объемного пикселя) – кубики(по аналогии с треугольниками) – могут содержать объем/вес – быстро рендерится.

Типы моделирования:

1. Каркасное – описываются только контуры объекта без плоскостей, моделирование самого низкого уровня, описывается в терминах точек и линий.

2. Поверхностное – описываются только снаружи – с помощью линий/точек/поверхностей – наиболее популярно.
3. Твердотельное – описывает и внутреннее тоже – обеспечивает полное однозначное описание трехмерной геометрической формы – с помощью него м.создать дырку в кубике.

Примитивы в полигональном способе:

- Луч
- Прямая
- Точка
- Вектор
- Плоскость

С их помощью можно задать полигональную сетку.

Функционально-параметрический способ(сплайны):
Сплайн – гибкий график; интерполяционный кубический сплайн – функция S(x), обладает следующими свойствами:

1. График этой функции проходит через каждую точку заданного массива: $S(x_i)=y_i$, i=0..m
2. На каждом из отрезков $[x_i, x_{i+1}]$, i=0..m-1 функция является многочленом 3-ей степени, $S(x)= \sum a^i_j (x-x_i)^j$, j = 0..3
3. На всем отрезке задания $[x_0, x_m]$ функция S(x) имеет непрерывную вторую производную.

Но чаще используются параметрически заданные кривые.
Параметрически заданная кривая – мн-во ϒ точек M(x,y,z), координаты которых определяются соотношениями(параметрическими уравнениями кривой ϒ): x=x(t); y=y(t); z=z(t); 0≤t≤b, где x(t), y(t), z(t) – функции, непрерывные на отрезке [a;b].

Также существует кривая Безье, определяемая массивом V($V_1V_2V_m$ – дуги кривой) – кривая, определяемая след. векторным уравнением:

$$r(t) = \sum_{i=0}^{m} C_m^i \, t^i (1-t)^{m-i} \, V_i, 0 \le t \le 1, \text{где}$$

$$C_m^i = \frac{m!}{i!\,(m-i)!} - \text{коэф. в разложении бинома Ньютона.}$$

При m=3 – кубическая кривая Безье.

Для 3D-моделирование используется аналог кривой Безье – B-Spline.

B-Spline кривая(для V_0, V_1, V_2, V_3) определяется следующим векторным параметрическим уравнением:

$$r(t) = \frac{(1-t)^3}{6}V_0 + \frac{3t^3 - 6t^2 + 4}{6}V_1 + \frac{-3t^3 + 3t^2 + 3t + 1}{6}V_2 + \frac{t^3}{6}V_3,$$
$$0 \le t \le 1$$

У B-Spline есть надстройка NURBS – рациональные B-Spline, задаваемые на неравномерной сетке.

Плюсы:

- Можно управлять детализацией
- Пересчету будут подлежать только ближайшие точки (при редактировании)
- Легко оценить точность и стабильность алгоритма

Компоненты NURBS-кривой:

1. Начало кривой
2. Направление кривой
3. Интервал
4. Точки редактирования
5. Управл.вершина
6. Каркас
7. Конец кривой

Одна поверхность или кривая не могут одновременно управлять и точками, и вершинами, но внутри одного объекта NURBS могут содержаться и Point, и CV.

Типы описания NURBS-кривых и –поверхностей:

1. Point
2. CV(Control Vertex)

Разница – в способе управления: Point управляется точками на самом объекте (объект проходит через эти точки), CV управляется вершинами, расположенными вне объекта и связанными между собой линиями.

NURBS-кривая C(u) является векторной кусочно-непрерывной рациональной полиномиальной функцией и определяется как

$C(u) = \frac{\sum_{i=0}^{n} w_i P_i N_{i,k}(u)}{\sum_{i=0}^{n} w_i N_{i,k}(u)}$, где w_i –вес, P_i – управл.вектор, $N_{i,k}$ – нормир.базисные B-Spline функций с шагом k

Чем больше w_i, тем ближе к ней проходит NURBS-кривая.

Этапы подготовки сцены:

1. Геометрическое моделирование сцены. Создание объектов, их пространственная привязка.
2. Топологическая привязка объектов. Создание кинематической схемы, определение изменения характеристик объектов с течением времени для анимации.
3. Задание физических свойств объектов. Наложение текстур, задание отражения, прозрачности и других свойств.
4. Определение физических свойств среды. Настройка атмосферных и гравитационных эффектов.
5. Расстановка источников света. Выбор точек наблюдения, установка камер.
6. Выбор модели освещения. Рендеринг.

Рассмотрим подробнее каждый из этапов.

1 – осуществляется с помощью примитивов:
сплайновое/полигональное(точки, линии, плоскости)/воксельное; делаем каркас.

2 – привязка объектов к геолокации, задание отношений объектов между собой, прописывание анимации.
Инверсная кинематика – когда выставляется общая топология объекта, прописывая для каждой части степени свободы (в каких плоскостях м.перемещать), потом просто можно потянуть за одну часть, и программа автоматически пропишет движение остальных присоединенных.
Определение изменения характеристик объектов с течением времени:

1) Риггинг – задание скелета 3D-модели – задание углов и степеней свободы для каждой части.
2) Скиннинг – создание кожи – созданный риг обшивается поверхностью. Можно до скиннинга сделать мышцы.
3) Хайринг – процесс обтягивания растительностью. Траектория волос – сложная.
4) Создание одежды

3 – С помощью текстур – определяется прозрачность, отражение, шероховатость и т.д.; физ.характеристики: вес, плотность – нужно для моделирования физических процессов(например, прыгает мяч)
Текстуры:
А) Проективные – берется растровая маска(картинка) и определенным способом проецируется на поверхность объекта.
Способы проецирования:

- Стандартные:
  I. Плоское – взяли плоскую поверхность и параллельно наклеили другую.
  II. Цилиндрическое – в цилиндрических координатах.
  III. Сферическое – для круглого, переход к сферическим координатам.
- Дополнительные:
  IV. UV Mapping – сложный способ; привязывается поточечно: берутся текстуры и привязываются к контрольным точкам, каким образом должно быть изменено правило проецирование текстуры на объекте.
  V. UC Mapping

Б) Процедурные – задается некая функция, которая занимается отрисовкой в зависимости от координат пикселя.
Создаем функцию, которая определяется каким-либо законом, в который входит некоторая периодическая функция. Чтобы результат не получился слишком правильным, моделируем с помощью шумовой функции, на которую накладываются следующие требования:

1) Непрерывность
2) Принимает значения из [0;1]
3) Ведет себя аналогично равномерно распределенной случайной величине.

С помощью этого способа моделируется то, где нужен быстрый рендеринг и минимум подкачки.

4 – Определение физических свойств среды - аналогично с 3, но для среды (туман, вода), необходимо задать коэффициенты преломления.
Настройка атмосферных и гравитационных эффектов – Particle Effects – техника, которая управляет поведением частиц и позволяет моделировать

различные сложные эффекты(взрыв, снег, дождь), разлетание по определенным законам. Чтобы это работало, задается поверхность, из которой вылетают частицы, и правила для нее. Может использоваться в фильмах для массовки.

5 – Источники света:
А) Spot(лампа) – конус света из определенной точки
Б) Free – направленный поток параллельный лучей
В) Omni – точечный источник, излучающий равномерно во все стороны

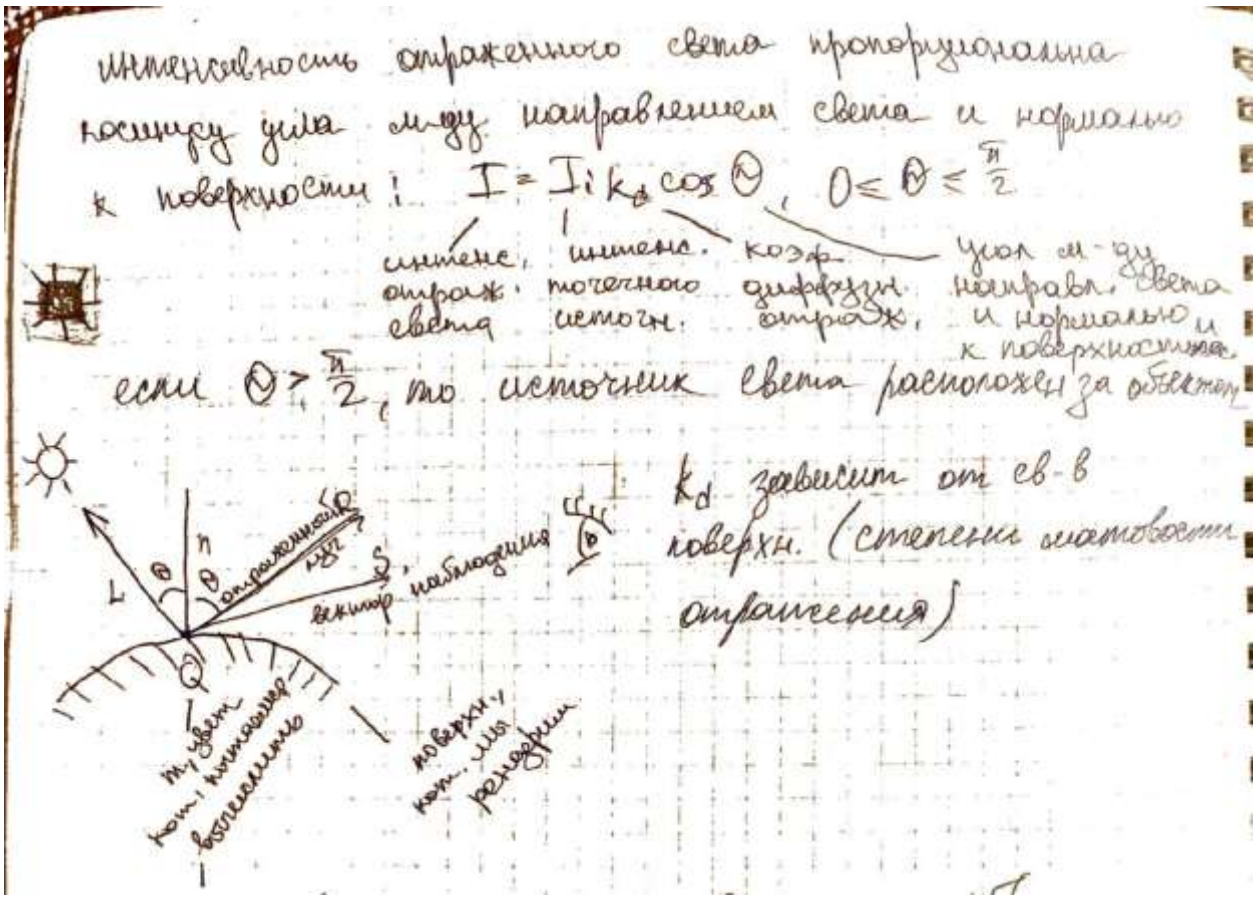
Характеристики источников света:

- Цвет(текстура)
- Яркость
- Объекты, которые им (не) освещаются
- Тени и т.д.

Расстановка камер определяется фокусное расстояние, угол обзора, эффект боке – размытие(чтобы не вся сцена была резкой, а выглядела более естественно).

6 –
Чтобы получить отраженный луч, нужно уметь его моделировать, в том числе, там, где отсутствует производная (на углах) – провести вектор нормали с помощью векторно-скалярного произведения/сложения с нормалями соседних граней.

Простая модель освещения:
Отраженный свет:
А) Зеркальный – отражается сразу от внешней поверхности объекта; положение наблюдателя важно
Б) Диффузный(матовый) – рассеивается равномерно по всем направлениям – всет точечного источника отражается от идеального рассеивателя по з.косинусов Ламберта:



Для вычисления: из т.Q строим обратным ходом вектор к источнику определяем расстояние L между источником и точкой Q(это важно для вычисления затухания света), строим нормаль n, отражающий луч R, учитываем вектор наблюдения S(для зеркальной составляющей).

Если использовать только з.Ламберта, то все будет блеклым(матовым), нужно добавить внешнюю освещенность(рассеянный свет), тогда

$I = I_a k_a + I_i k_d cos\theta$, где $I_a -$ интенсивность рассенного света, k_a $-$ коэффициент диффузного отражения рассеянного света

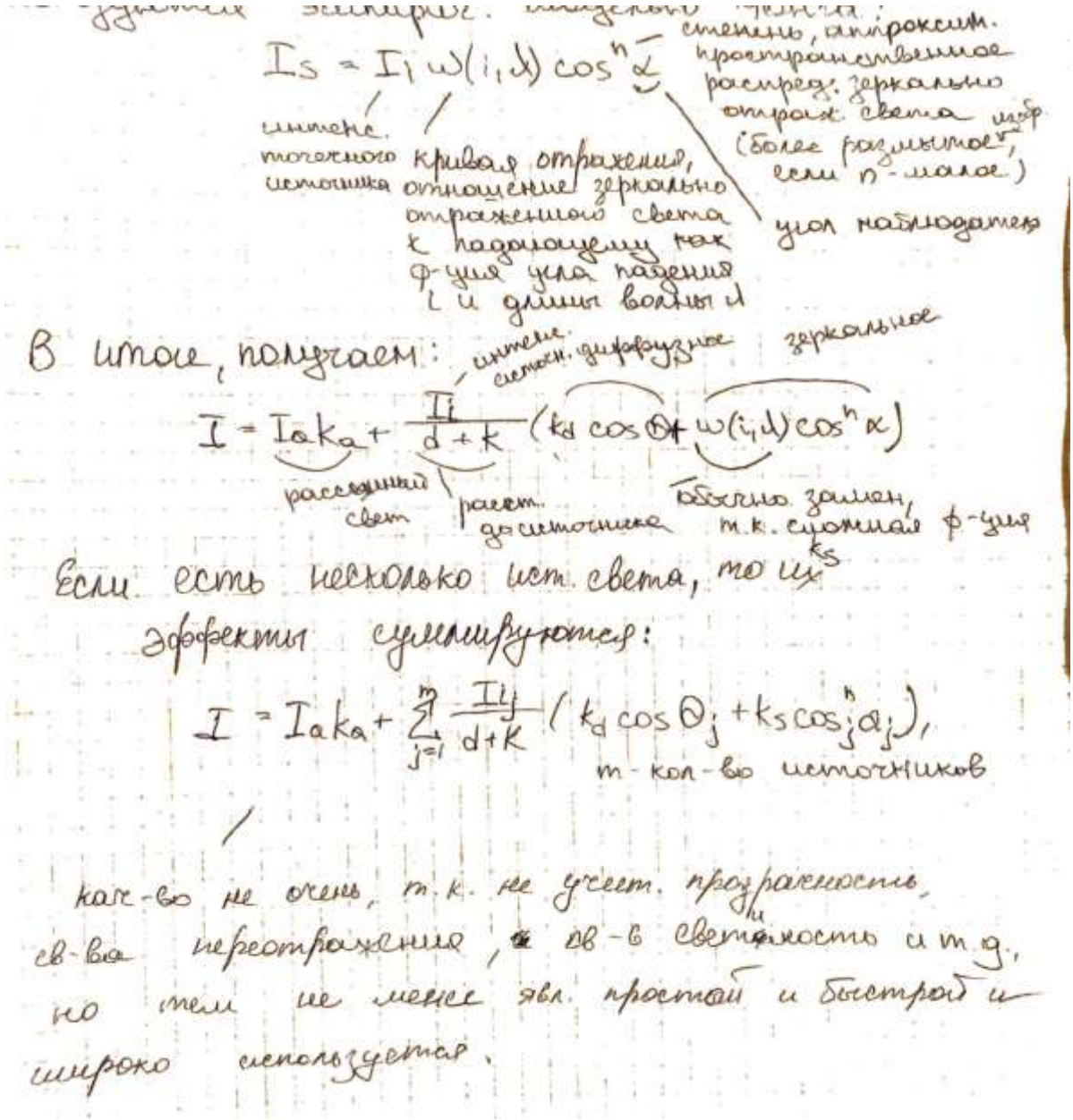
Также нужно учесть расстояние до источника света:

$I = I_a k_a + \frac{(I_i k_d cos\theta)}{d + k}$, где $k -$ константа,

$d -$ расстояние от центра проекции до объекта

Добавляем зеркальную составляющую, она в общем случае зависит нелинейно от угла падения/длины волны/свойств вещества.

В простой модели освещения пользуются эмпирической моделью Фонга:



Методы рендеринга:

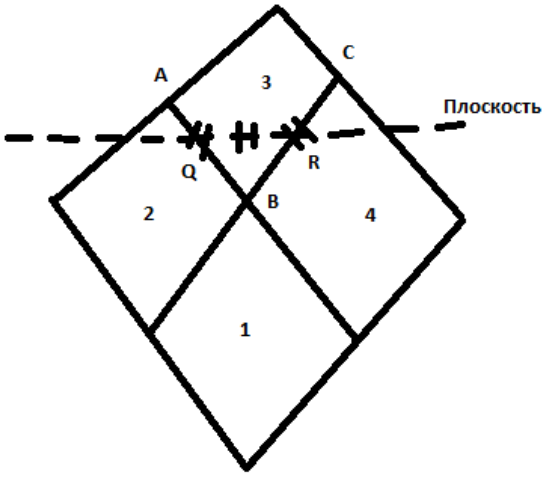
Закраска (наиболее быстрые):

I. Сканирующая плоскость, метод Гуро
II. Интерполяция вектора нормали, метод Фонга
III. Глобальная модель освещенности Уиттеда

Более реалистичные, но сложные:

IV. Прямая/обратная трассировка лучей
V. Метод излучательности

I, II – используется принцип секущей плоскости: берем экран и по нему пускается секущая плоскость сверху вниз и слева направо. Плоскость перпендикулярна поверхности экрана, и она пронизывает все объекты перпендикулярно, в глубину. Простыми алгоритмами(Z-буфер) определяется видимость объектов относительно наблюдателя и определяет, какие конкретно фрагменты объекта нужно визуализировать. Для ускорения процесса не будем прогонять через формулу простой освещенности каждый пиксель, применим интерполяцию.



Чтобы посчитать цвет точки P, сначала посчитаем цвет A и B, B и C; потом, относительно плоскости мы интерполируем цвета точек A и B в точку Q и цвета B и C в точку R. Такой же интерполяцией относительно Q и R получаем цвет в точке P.

Для I:

$$I_Q = u * I_A + (1-u)I_B, 0 \le u \le 1, u = \frac{AQ\ (\text{расстояние})}{AB}$$

$$I_R = w * I_B + (1-w)I_C, 0 \le w \le 1, w = \frac{BR\ (\text{расстояние})}{BC} =>$$

$$I_P = t * I_Q + (1-t)I_R, 0 \le t \le 1, u = \frac{QP\ (\text{расстояние})}{QR}$$

Вычисляются нормали только для A, B, C, используя формулу простой освещенности.

Для II: подход такой же, но интерполируется не значение интенсивности, а вычисленный вектор нормали.

I хорош для диффузных отражений, II – для зеркальных.

В II зеркальное отражение лучше, так как нормали имеют своё истинное направление, на стыках нет разрывов.

III – учитывает отражения других объектов и учитывает прозрачность – возможность преломления поверхности.

IV – основана на идее отслеживания каждого луча, который попадает к наблюдателю.
Способы отслеживания:
А) Прямая трассировка – отслеживаем каждый луч, выпущенный из источника света, и ведем его к наблюдателю до тех пор, пока он попадет (или не попадет). Данный способ на практике практически не используется для рендеринга, так как почти все лучи растворяются в пространстве, не попадая.
Б) Обратная трассировка – смотрим со стороны наблюдателя и, сквозь поверхность экрана, пускаем обратные лучи, перпендикулярные экрану, сквозь каждый пиксель рендера, след-но, кол-во лучей равно кол-ву пикселей изображения; каждый луч отслеживается до пересечения с каким-либо объектом сцены; если он не пересекается, то отслеживание прекращается; если столкнулся с объектом, то из точки столкновения испускается минимум 3 луча: отраженный, преломленный(если объект полупрозрачный), лучи к источнику света(смотря, сколько источников света). Затем аналогично повторяем процедуру заново – рекурсивно отслеживаем все лучи до определенного момента(выход из рекурсии). Выход из рекурсии осуществляется в двух случаях: по глубине рекурсии(кол-во итераций) и по затуханию луча(интенсивность ниже заданного порога).
Этот способ дает реалистичное отображение всех объектов на сцене 1 в 1.

В трассировке применяются методы оптимизации:

- Отсечение по глубине
- Отсечение по весу
- Метод Монте-Карло – для борьбы с лестничными эффектами
- Копирование объектов
- Метод описания ограничивающих фигур – вокруг сложных объектов описываем простую выпуклую фигуру и сначала проверяем пересечение лучей с ним – если не пересекается, то дальше не идем; м.описать дерево огранич.фигур.

- Метод равномерного разбиения пространства – само пространство сцены «нарезается» на кубики и запоминается, внутри каких кубиков есть что-то, а в каких нет; потом проверяем прохождение луча через эти кубики – если луч проходит только через пустые, дальше его не рассматриваем.

Недостаток IV: Если камера чуть сместилась в следующем кадре, все нужно считать заново.

В 3Ds стоит авторский метод рендеринга, который использует идеи метода трассировки лучей.

V – обеспечивает и высокую точность при работе с диффузными объектами, и отдельное вычисление глобальной освещенности независимо от положения наблюдателя.
В основе – лежит закон сохранения энергии в замкнутой системе. Сцена – замкнутая система, в которой должны сохраняться уравнения баланса энергии:

$$B_i = E_i + p_i \sum_{j=1}^{n} F_{ij} B_j, \qquad i = 1..n$$

$B_i -$ энергия, отбрасываемая $i -$ тым фрагментом сцены
$n -$ количество фрагментов, на которые разбита сцена
$E_i -$ собственная излучательность фрагмента
$F_{ij} -$ доля энергии $j -$ того фрагмента, попад. на $i -$ тый фрагмент
$p_i -$ коэффициент отражения

В результате получаем СЛАУ, решив которую, посчитаем точно энергию(светимость каждого фрагмента).
Наиболее сложно вычислить F_{ij}: Выберем фрагменты A_i, A_j и элементарные участки dA_i, dA_j с нормалями n_i, n_j, тогда доля энергии элемента dA_j, попадающего на элемент A_i:

$$F(dA_i, dA_j) = \frac{\cos\varphi_i cos\varphi_j}{\pi r^2} \text{ или } F(A_i, A_j) = \frac{1}{A_i} \iint_{AiA_j} \frac{\cos\varphi_i cos\varphi_j}{\pi r^2}\, dA_i dA_j$$

5. Ход выполнения

Сначала я создала Box-ы и Cone-ы, и с помощью них сконструировала шкаф с будущим зеркалом. Потом создала тело вращения (форму для шарлотки) с помощью Spline-а и модификатора Lathe, который придает форму объекту. Также создала комод с еще одним зеркалом, добавила пару объектов из примитивов и создала из них юлу и заготовку для плюшевого авокадо.
После этого попробовала выставить освещение, с помощью таргетированного света и точечного источника света(фонового), поставила тени.
Создала пол, к которому позже добавились потолок и стены, чтобы созданные объекты не были просто в пустоте.
Затем училась работать с материалами в Material Editor-е: искала и загружала необходимые текстуры, которые придумывала по ходу выполнения задания, корректировала уровень отражения.
У некоторых объектов пришлось изменить положение материала, чтобы они надлежащим образом выглядели на объектах; для этого я использовала Gizmo в модификаторе UVWMap. Самым сложным для изменения оказался материал для авокадо, поскольку я не рисовала текстуру сама, а взяла готовую из интернета. В итоге, у меня получился двуликий авокадо.
После этого нужно было заняться зеркалами и создать материал для них. В перечне стандартных материалов выбрала Raytrace и в графе отражения поставила белый цвет, который дает полное отражение.
В моей комнате в реальности на стенах висит много картинок, фотографий и рисунков, поэтому я решила добавить немного забавных картинок на одну из стен с помощью тех же Box-ов и наложения материалов.

6. Результат

И еще немного скриншотов…


ТЕПЛОТА

7. Вывод

3Ds Max имеет широкие возможности для создания различных типов объектов, от самых простых до самых сложных. Данная лабораторная работа помогла мне познакомиться с основами 3D-моделирования и получить некоторый опыт. Она интересна тем, что, помимо именно обучения 3D-моделированию, позволяет проявить творческие способности, поскольку нет конкретного ТЗ, есть лишь аспекты, которые обязательно должны быть, а остальное – на выбор исполнителя.

Моделирование виртуальных миров в программах

«Internet Space Builder» и «Unity 3D»

1. Цели

Создание виртуальной трёхмерной обстановки для работы с ней через сеть Internet. Знакомство и экспорт в формат VRML 97. Знакомство с Unity.

2. Задание

Импортировать результаты работы ЛР №1 в среды ISB и Unity. Проверить соответствие цветов и текстур. Добавить к телу вращения два любых вида произвольной интерактивности и проверить результаты работы с помощью plug-in'ов к веб-браузерам.
В выводах описать сравнительные преимущества каждой из исследуемых программных сред.

3. ПО

ISB 3.0, Cortona VRML Client, Unity 5.6.4f1

4. Теория

Система виртуальной реальности – комплекс программно-аппаратных средств, создающих эффект погружения пользователя в искусственно созданную трехмерную среду. Максимальный эффект достигается при возможности интерактивного взаимодействия пользователя с окружающей виртуальной средой.
Области использования VR-технологий: визуализация, моделирование, навигация, развлечения, игры.
Одной из наиболее популярных областей применения, помимо игровых виртуальных сред, являются интерактивные WEB-приложения:

- Виртуальные руководства пользователя с 3D-разделами
- Конструкторские и инженерные приложения со сложным моделированием
- Курсы дистанционного обучения
- Сценарное 3D-моделирование событий
- Виртуальные экскурсии по городам, музеям, замкам
- Дизайн внутренних интерьеров и тд

Для создания виртуальных средств в среде Internet используется широкий спектр программных средств:

1. Панорамные видеотехнологии – создают виртуальную среду, в которой угол обзора составляет 360 градусов за счет склейки отдельных изображений в круговую панораму. Для получения качественных изображений требуется специальный круговой штатив с автоматизацией шага съемки или вращающаяся панорамная камера.

Чтобы можно было рассматривать конкретные объекты с любой степенью детализации и с любого угла, они фотографируются или рендерятся отдельно и компонуются с общей панорамой.

2. VRML – язык моделирования виртуальной реальности, описания трехмерных сцен, в котором определены иерархические преобразования, источники света, возможность произвольной смены точки наблюдения, геометрические тела, анимация, различные свойства материалов и атмосферных сред.

Он обеспечивает технологию для интеграции двумерных и трехмерных объектов текста и мультимедийных данных в единой согласованной модели виртуального мира.

Среди объектов VRML-графики есть аватар – анимированный объект, часто с человеческими чертами, с которым можно взаимодействовать и который может управлять как пользователем, так и компьютером.

Средства работы с VRML-сценами:

1. Трехмерные среды для создания VRML-сцен (ISB)
2. Средства визуализации VRML-сцен(Cortona, CosmoPlayer): работает в соответствии с установками конкретного web-сервера. Клиентский браузер посылает запросы на web-сервер, который должен возвратить запрошенный документ в формате стандарта MIME. Браузер анализирует эту информацию и с помощью плагинов воспроизводит VRML-сцены.
3. Прямое текстовое кодирование на языке VRML:
   Файл VRML состоит из следующей послед-ти функциональных блоков: заголовка, описания прототипов, описания трехмерной сцены, маршрутов событий. Каждая строка, начинающаяся со знака «#», считается комментарием, кроме заголовка файла. После заголовка идут описания трехмерных объектов, их параметров и свойств, а также события и их обработчики для создания динамики в виртуальном мире.

Основные понятия языка:

1. Имена – регистро-зависимые, не кириллица, не управляющие символы от 0 до 32, без синтаксических символов, без цифр в начале.
2. Узлы – все трехмерные тела и любые мультимедийные сообщения(весь трехмерный мир представляет собой дерево объектов)

   Типы:

   А) группировочные – объединение вложенных узлов в ветви «дерева сцены» - родители

   Б) конечные - определяют конкретные геометрические примитивы, звуки, видео, камеры; не могут иметь вложенных узлов-детей, кроме вспомогательных

   В) вспомогательные – специфические узлы, располагающиеся только в определенных узлах: сенсоры, узлы-интерполяторы, которые меняют объекты сцены в соответствии с заданным набором опорных точек, и некоторые другие.

   3. Поля и события.

   Поля – уникальные параметры, отличающие данный объект от других аналогичных, каждое поле имеет свой тип и значение по умолчанию.

   События – средства обмена инф-цией между узлами; входящие и исходящие события.

В VRML определены следующие примитивы: куб, сфера, цилиндр, конус. Цвет фигуры определяется с помощью объекта Material, либо <Texture2>(текстура).

По умолчанию любой описанный объект будет располагаться точно по центру окна браузера, поэтому, если описать два одинаковых цилиндра, то они сольются друг с другом. Поэтому нужно поменять координаты второго цилиндра с помощью узла **Translation**(определяет координаты объекта)
Узлы, модифицирующие св-ва фигур, действуют на все фигуры. Чтобы ограничить область их действия, фигуры необходимо сгруппировать с помощью узла **Separator**
Rotation – вращение фигур вокруг осей координат
Scale – масштаб по x,y,z
Anchor – возможность перемещения между виртуальными мирами или переход по гиперссылке

Сенсоры окружающей среды – контролируют течение процессов в окружающей среде(течение времени и расположение пользователя):

- Сенсор времени – не имеет конкретной позиции в виртуальном мире и ассоциированных с ним геометрических объектов; может использоваться для порождения событий или управления узлами-интерполяторами
- Сенсор видимости – представляет собой некоторый параллелепипед, генерирующий события, когда он попадет в/выходит из поля зрения пользователя, по которым можно точно определить время и характер произошедшего
- Сенсор приближения – похож на сенсор видимости, но также отслеживает перемещения внутри параллелепипеда и генерирует сообщения при изменении позиции/ориентации пользователя
- Сенсор коллизий – включение/выключение распознавания коллизий(столкновений) между потомками этого узла и пользователем

Узлы-манипуляторы:

- Датчик касания – реагирует на взаимодействие курсора мыши с геометрическими объектами – потомками своего родительского узла.

Датчики буксировки:

- Сферический сенсор – вращение объекта вокруг центральной точки самого сенсора
- Цилиндрический сенсор – вращение только вокруг одной оси
- Плоский сенсор – перемещение вдоль осей X и Y в локальной системе координат

Unity3d - это очень гибкий мультиплатформенный движок, предоставляющий большую свободу действий пользователю и работающий под различными операционными системами. Возможность импортирования моделей напрямую из 3d редактора, является одной из самых главных. В Unity3d чаще всего создаются компьютерные игры.

5. Ход выполнения

Работа с Unity. Экспортируем из 3Ds max комнату(при экспорте в *.fbx) указываем AxisConversion: Z-up – нужно, так как Unity использует систему координат с осью Y, направленной вверх.

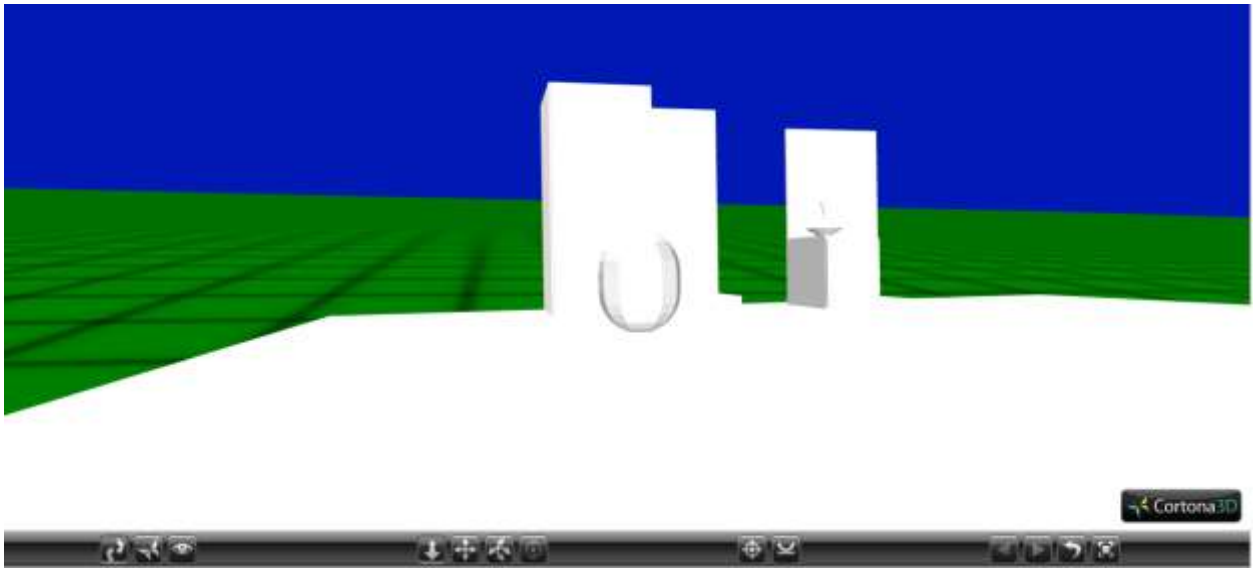
Затем импортируем ее в Unity. Подгружаем заново все материалы (важно, чтобы они не были с одинаковыми именами). Возникла проблема с материалом зеркала, поскольку его невозможно было экспортировать из 3Ds max корректно, и я попыталась написать скрипт для отражения. К сожалению, он работает не полностью.
Необходимо взаимодействие с предметами сцены. Для этого воспользуемся FPSController(управление персонажем от первого лица) из StandartAssets(они скачиваются и импортируются из AssetStore).
Нужно добавить несколько видов произвольной интерактивности, и для этого были выбраны следующие предметы: дверца шкафа (открывается/закрывается со звуком), миска (падает со звуком), юла (начинает покачиваться со звуком). Все взаимодействия осуществляются по клику.
Для каждого из предметов, а также для FPSController был написан C#-скрипт с последовательностями действий в каждый момент времени.
Основная сложность заключалась в прописывании определенных параметров при написании скриптов (подбор угла/расстояния).
После того, как в режиме Game все взаимодействия начали работать корректно, установила Unity WebGL для работы проекта в браузере. К сожалению, современные версии Google Chrome/Mozilla Firefox не поддерживают данный плагин из соображений безопасности, если его запускать с компьютера, отдельно собрав и сохранив перед этим. Но если нажать кнопку «Build and Run», то запустится небольшой локальный веб-сервер, через который страница загрузится.

Далее, я перешла к работе с программами Internet Space Builder 3.0 и Cortona VRML Client. Но сначала вернулась к проекту комнаты в 3Ds max и ре-экспортировала его в другой формат файла, так как оказалось, что ISB воспринимает на импорт только файлы формата *.3ds. При импорте сцены я обнаружила, что она слишком большая по некоторым критериям для урезанной версии программы. А поскольку полную версию программы было найти практически невозможно, пришлось перетащить все примитивы отдельно на сцену в ISB(как самостоятельные объекты), а затем удалять по одному объекту до тех пор, пока кнопка сохранения не стала «Enabled». Также, даже при исходном импорте перенеслись не все объекты, небольшая часть из них пропала. Когда, наконец, у меня появилась возможность сохранения сцены, нужно было сделать 2 вида произвольной

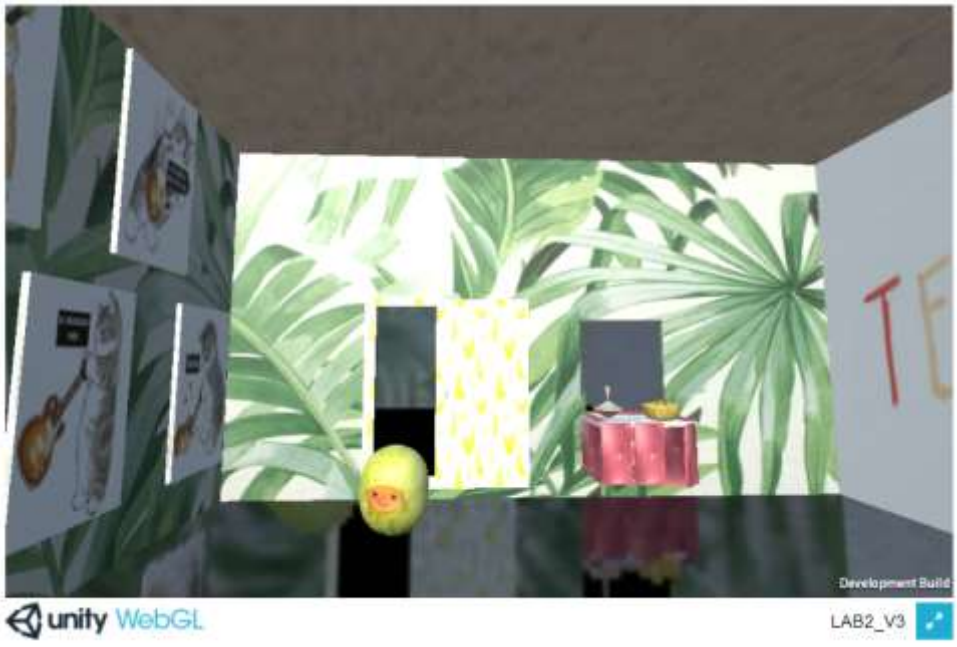
интерактивности. В данном проекте – при нажатии на предполагаемое авокадо издается звук, а при клике на юлу – переход по ссылке. После завершения работы с ISB была установлена Cortona VRML Client в качестве плагина для браузера. Примечательно, что данная программа «заставила» мой ноутбук «вспомнить» о существовании Internet Explorer, поскольку только в нем она может работать корректно, не применяя лишних усилий к ее установке.

6. Результат

Для ISB 3.0 & Cortona VRML Client:



Для Unity3D & Unity WebGL:



7. Вывод

Если сравнивать Unity3D и ISB, то, на самом деле, их довольно сложно сравнивать, поскольку ISB 3.0 – ПО, выпущенное относительно давно и имеющее куда меньший функционал, чем Unity. Конечно же, Unity абсолютно превосходит ISB при выполнении данной работы, кроме, разве что, одного пункта: для добавления интерактивности в ISB потребовалось куда меньше времени и усилий, чем в Unity. В остальном же Unity – явный лидер в этой «гонке».

Множества Жюлиа и Мандельброта

1. Цели

Изучить процесс построения алгебраических фракталов и результаты их визуализации.

2. Задание

1. В среде программы FractInt рассмотреть классическую формулу z(n+1)=z(n)2+c (mandel). Увеличить масштаб, с помощью правой клавиши мыши изучить вид соответствующих множеств Жюлиа. В отчете привести пример связного множества Жюлиа, Канторовой пыли.

В качестве параметров формулы mandel задать:

· для группы 08-406: Real Perturbation of Z(0) = 0.05*n

· для группы 08-407: Imaginary Perturbation of Z(0) = 0.05*n

· для группы 08-408: Real Perturbation of Z(0) и Imaginary Perturbation of Z(0) = 0.05*n

где n – порядковый номер по списку.

Индивидуальное задание: Imaginary Perturbation of Z(0) = 0.05

2. Подобрать для формулы удобный вид с помощью клавиш позиционирования <PgUp> и <PgDown>, клавиш палитры <+> и <->; привести изображение в отчете.

3. Рассчитать неподвижную траекторию, привести пример точки, для которой последовательность будет ограничена.

3. ПО

FractInt, запускаемый из эмулятора DOS “DOSBox”.

4. Теория

Идея, использованная Мандельбротом, состояла в том, чтобы вместо действительных чисел рассмотреть комплексные и наблюдать процесс $z_0 \rightarrow z_1 \rightarrow z_2 \rightarrow \ldots$ не на прямой, а в плоскости. Процесс Мандельброта основан на простой формуле

$$z_{n+1} = f(z_n) = z_n^2 + c$$

Множество Мандельброта M для полинома $f_c(z) = z^2 + c$ определяется как множество всех c є C, для которых орбита точки 0 ограничена:

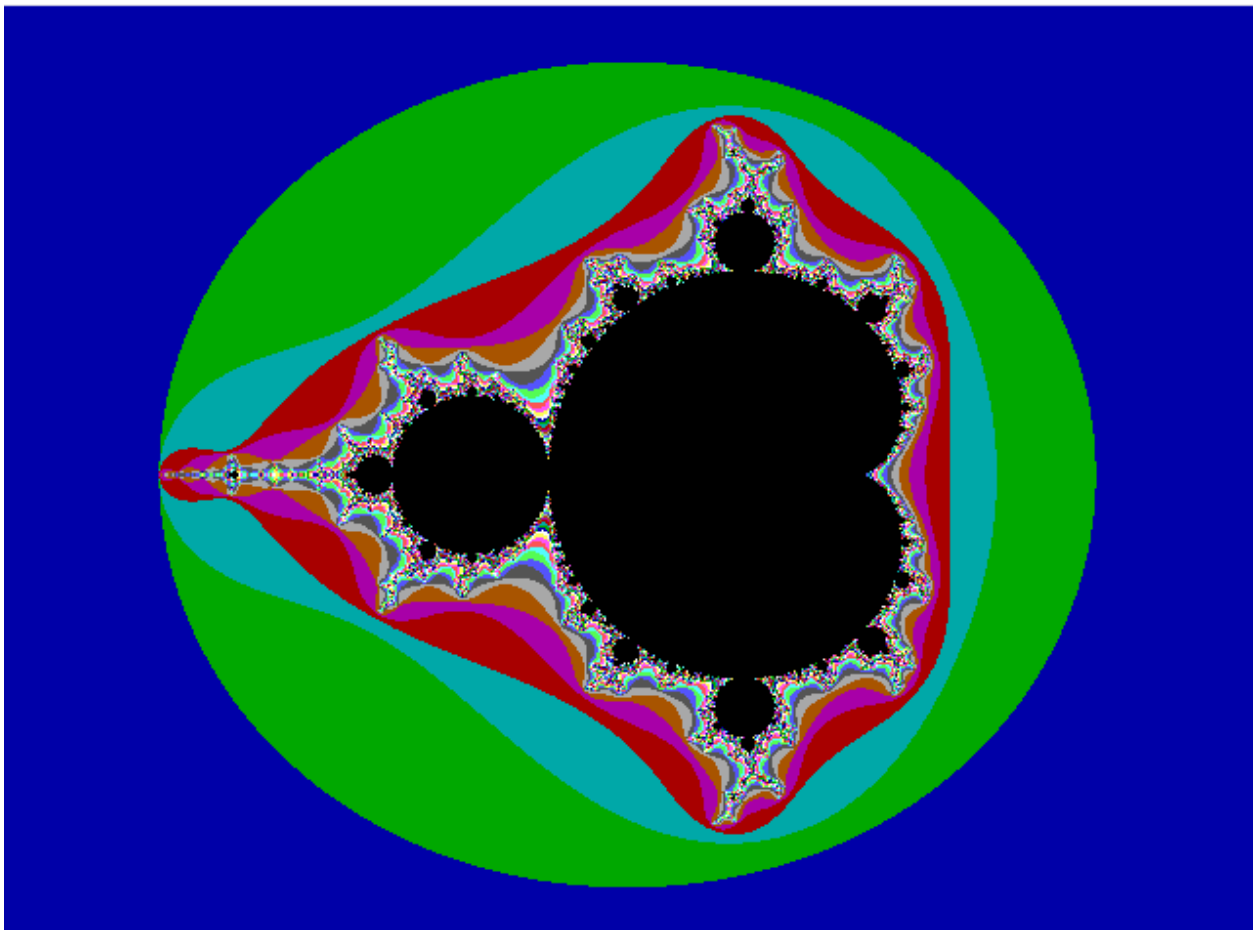
$$M = \{c \epsilon C : \{f_c^{(n)}(0)\}_{n=0}^{\infty} \text{ ограничена}\}$$

Множество Жюлиа функции f, обозначаемое как J(f), определяется как

$$J(f) = \partial\{z : f^{(n)} \to \infty \text{ при } n \to \infty\}$$

Таким образом, множество Жюлиа функции f есть граница множества точек f(z), стремящихся к бесконечности при итерировании f(z).

Строго математически, изображения множеств Мандельброта и Жюлиа должны быть чёрно-белыми — точка либо принадлежит множеству, либо нет. Но были предложены варианты сделать изображения цветными. Самым распространённым способом является окрашивание точек около внешней границы множества в зависимости от количества итераций, за которое становится очевидным, что точка не принадлежит множеству (за которое начинает выполняться критерий $|z_n| > 2$).



На картинке виден классический фрактал множества Мандельброта: черный цвет в середине показывает, что в этих точках функция стремится к нулю –

это и есть множество Мандельброта. За пределами этого множества функция стремится к бесконечности. Границы множества являются фрактальными, функция ведет себя непредсказуемо – хаотично.

Точки, принадлежащие множеству Мандельброта, соответствуют связным множествам Жюлиа, а точки не принадлежащие — несвязным.

Когда произойдет выход за границу множества Мандельброта, сопуствующее ему множество Жюлиа как бы взорвется, превратившись в Канторову пыль. Эта пыль становится все мельче с удалением точки от множества Мандельброта.

5. Ход выполнения

Зададим параметр формулы mandel на ImaginaryPerturbation of Z(0) = 0.05 (по варианту).

Подбираем для формулы удобный вид с помощью клавиш позиционирования <PgUp> и <PgDown>, клавиш палитры <+> и <->; изображения приведены в пункте 6.

Неподвижная траектория — точка, при которой выполняется равенство f(z) = z.

$$f(z_{n+1}) = z_{n+1}^2 + \Re(c) + i\Im(c) = \left(\Re(z_n) + i\Im(z_n)\right)^2 + c = z_{n+1} = \Re(z_n) + i\Im(z_n)$$

$$\begin{cases} \Re(z_n) = \Re(z_n)^2 - \Im(z_n)^2 + \Re(c) \\ \Im(z_n) = 2\Re(z_n)\Im(z_n) + \Im(c) \end{cases}$$

Подставим $z_n = 0.05i$:

$$\begin{cases} -0.05^2 + \Re(c) = 0 \\ \Im(c) = 0.05 \end{cases}$$

c = 0.0025 + 0.05i – неподвижная точка.

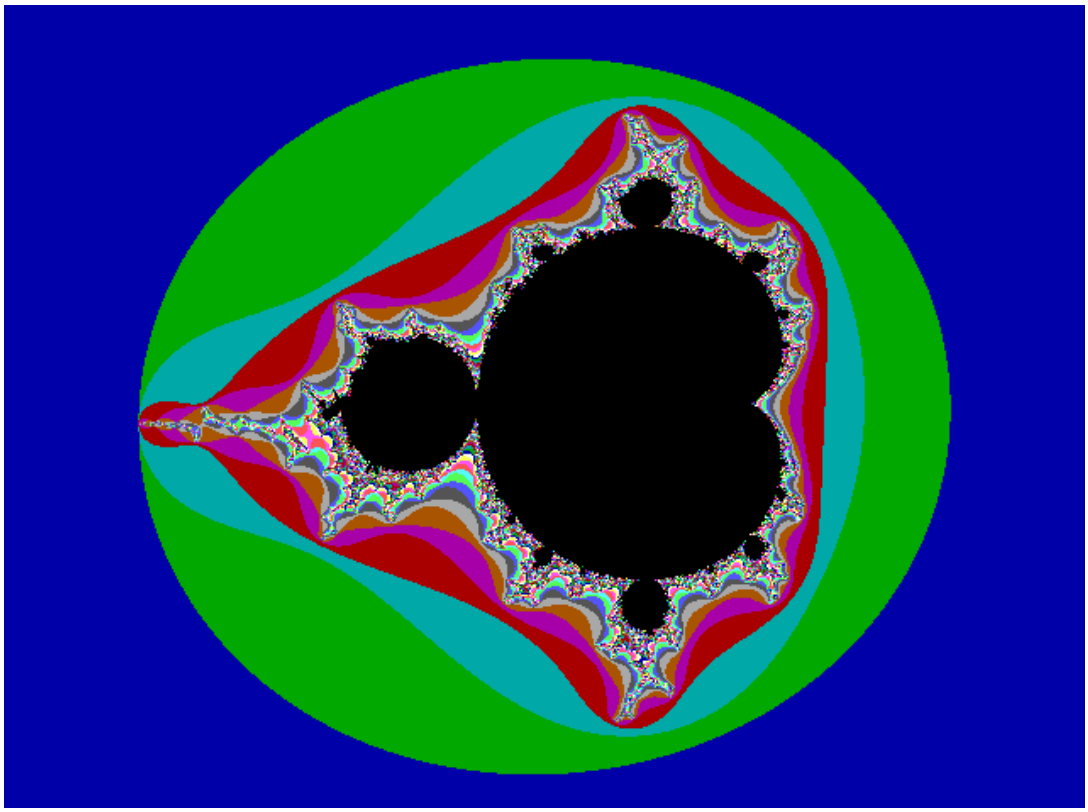
Рассмотрим точку, для которой последовательность ограничена — любая точка из множества Мандельброта. Например, точка c = 0.2-0.3i. Рассчитаем

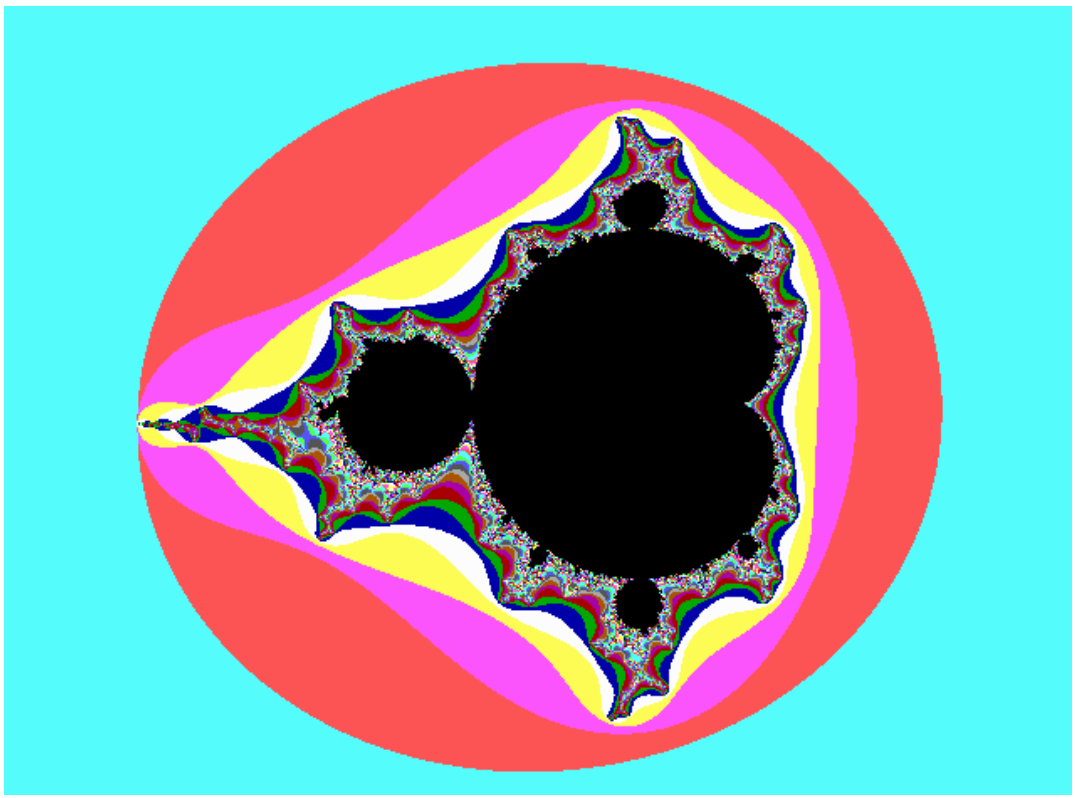
последовательность f для неё – запустим итерационный процесс до тех пор, пока полученные значения не будут лежать достаточно близко друг к другу:

```
it: 1 z: (0.1975-0.3j)
it: 2 z: (0.14900625-0.4185j)
it: 3 z: (0.04706061253906252-0.42471823124999997j)
it: 4 z: (0.021829125296423335-0.3399750002382644j)
it: 5 z: (0.0848935099241991-0.3148427137557052j)
...
it: 22 z: (0.07903139857594357-0.35655871106749615j)
it: 23 z: (0.07911184752275546-0.3563586672202j)
it: 24 z: (0.07926718471550649-0.3563843850890736j)
it: 25 z: (0.07927345663740512-0.3564991737651556j)
it: 26 z: (0.07919262003200396-0.3565218435854856j)
it: 27 z: (0.0791636461139399-0.3564677977843498j)
it: 28 z: (0.07919759200882903-0.3564385811896315j)
it: 29 z: (0.07922379641951943-0.3564581546585246j)
```

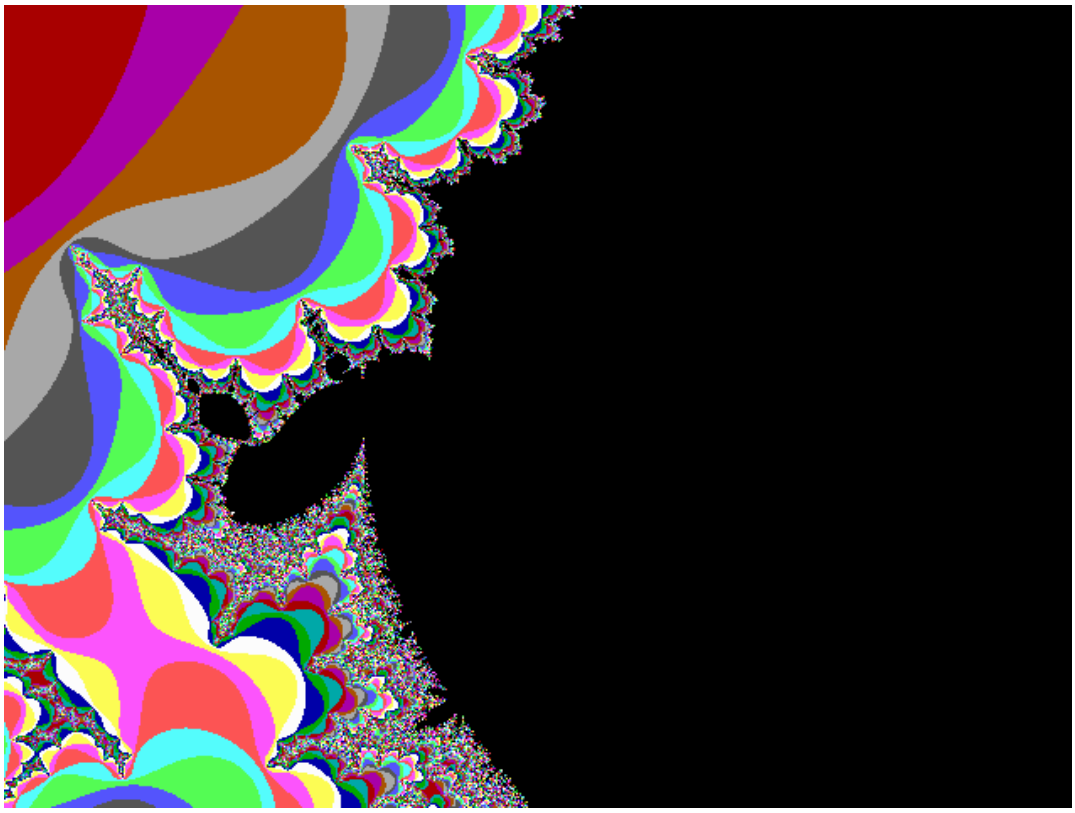
6. **Результат**

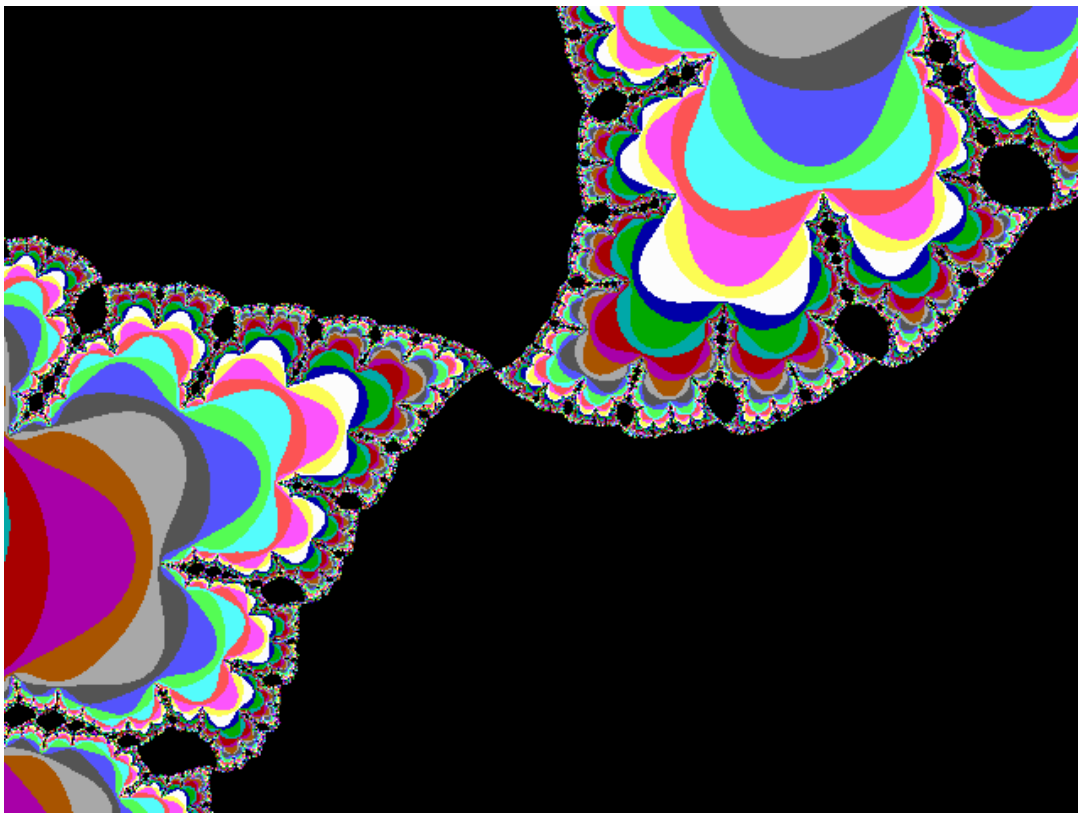
Множество Мандельброта — черные точки:





Множество Жюлиа — границы множества Мандельброта.





7. Вывод

В данной лабораторной работе был рассмотрен простейший фрактал — граница множества Мандельброта.

Итеративные системы функций. Фрактальная компрессия изображений

1. **Цели**

Ознакомиться с основными принципами работы фрактальной компрессии и декомпрессии.

2. **Задание**

В программе WinFact изучить формулу fern (папоротник) и аффинные преобразования для получения изображения папоротника. Вычислить текущий угол поворота левой ветви. Аффинным преобразованием повернуть левую ветвь папоротника на угол 10 градусов + номер по списку(по варианту: 11 градусов). В отчете привести изображение модифицированного папоротника и математические выкладки по расчету коэффициентов матрицы итеративной системы функций.

3. **ПО**

Fractint, запускаемый из эмулятора DOS “DOSBox”.

4. **Теория**

Папоротник Барнсли - фрактал, названный в честь Майкла Барнсли, британского математика, который первым описал его в своей книге "Фракталы Повсюду". Является одним из основных примеров "самоподобных" множеств, т.е. представляет собой математически генерируемый "шаблон", воспроизводимый при любом увеличении или уменьшении количества итераций.

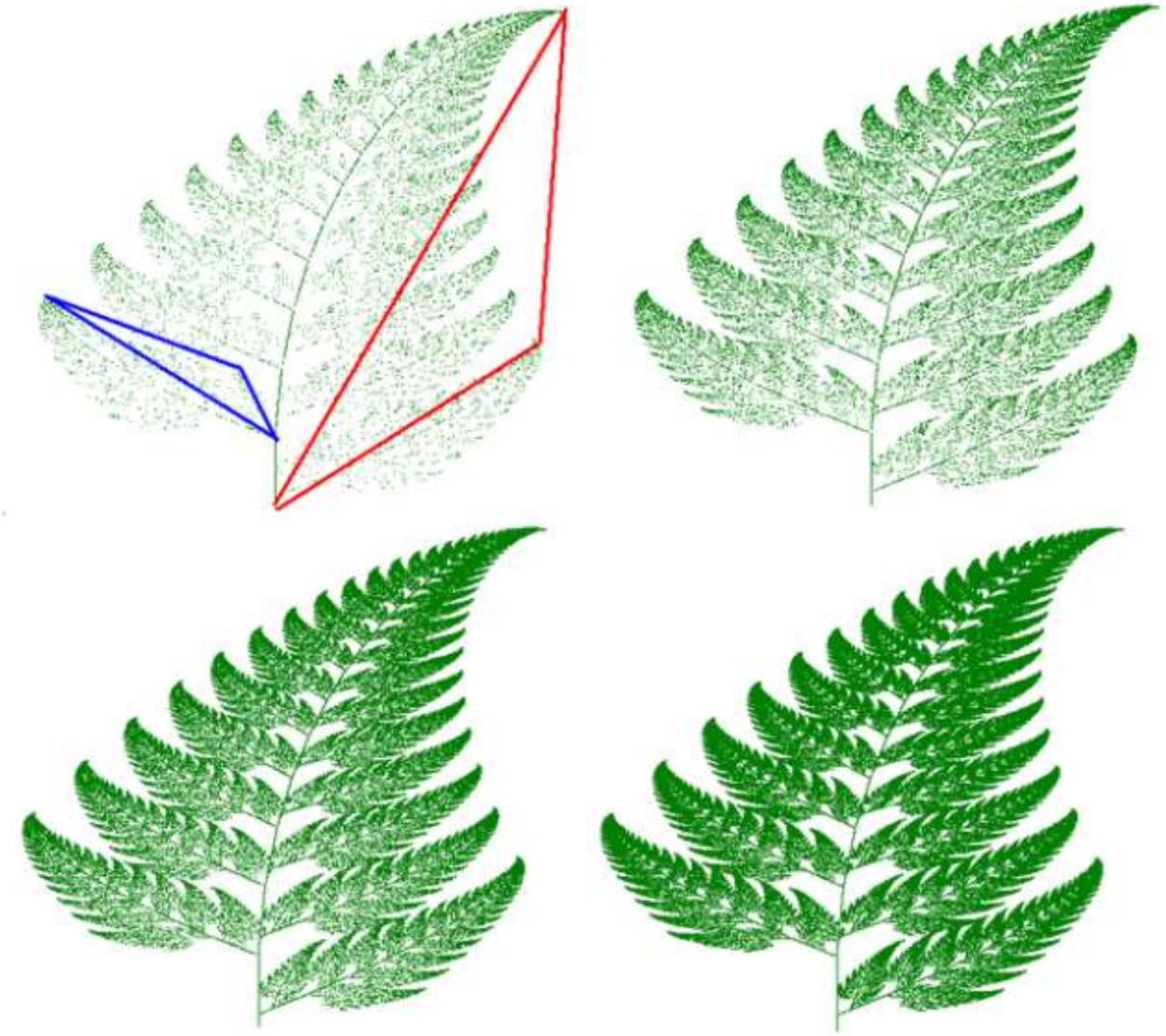
Папоротник Барнсли строится при помощи 4-х аффинных преобразований вида:

$$f(x,y) = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} e \\ f \end{bmatrix}$$

w	a	b	c	d	e	f	p

$f1$	0	0	0	0.16	0	0	0.01
$f2$	0.85	0.04	-0.04	0.85	0	1.6	0.85
$f3$	0.2	-0.26	0.23	0.22	0	1.6	0.07
$f4$	-0.15	0.28	0.26	0.24	0	0.44	0.07

где столбцы a-f - коэффициенты уравнения, а p - коэффициент вероятности.



Первая точка находится в начале координат (x0 = 0, y0 = 0), а затем новые точки итеративно вычисляются путем случайного применения одного из следующих четырех преобразований координат:

- (1) $\mathbf{x_{n+1} = 0; y_n + 1 = 0.16\ y_n}$.

Данное преобразование выбирается в 1% случаев и указывает на точку у основания "стебля". Эта часть рисунка в результате итерационных преобразований завершается первой.

- (2) $x_{n+1} = 0.85\, x_n + 0.04\, y_n;\ y_{n+1} = -0.04\, x_n + 0.85\, y_n + 1.6.$

Преобразование (2) используется в 85% случаев и указывает на любую точку листовки попадающую в красный треугольник

- (3) $x_{n+1} = 0.2\, x_n - 0.26\, y_n;\ y_{n+1} = 0.23\, x_n + 0.22\, y_n + 1.6.$

Выбирается в 7% случаев - попадания точки в синий треугольник и симметричного ему относительно главного стебля треугольника.

- (4) $x_{n+1} = -0.15\, x_n + 0.28\, y_n;\ y_{n+1} = 0.26\, x_n + 0.24\, y_n + 0.44.$

В оставшихся 7% случаев используется преобразование (4) - для симметричных преобразованию (3) относительно стеблей 2-го порядка позиций.

5. Ход выполнения

Исходное изображение фрактала fern описывается следующим набором коэффициентов:

```
fern {
    0   0     0   .16 0       0 .01
   .85 .04  -.04  .85 0 1.6  .85
   .2   -.26   .23  .22 0 1.6  .07
  -.15 .28   .26  .24 0 .44  .07
  }
```

Нас интересует только левая ветвь папоротника, преобразования которой описаны в 3 строке. Аффинные преобразования поворота и масштабирования описаны в первых 4 коэффициентах *a, b, c, d*:

$$\Lambda = \begin{pmatrix} a & b \\ c & d \end{pmatrix} = \begin{pmatrix} 0{,}2 & -0{,}26 \\ 0{,}23 & 0{,}22 \end{pmatrix}$$

Эта матрица является произведением матрицы поворота на матрицу масштабирования, т.е.

$$\Lambda = R \cdot S = \begin{pmatrix} \cos(\alpha) & -\sin(\alpha) \\ \sin(\alpha) & \cos(\alpha) \end{pmatrix} \cdot \begin{pmatrix} s_x & 0 \\ 0 & s_y \end{pmatrix} = \begin{pmatrix} \cos(\alpha)\, s_x & -\sin(\alpha) s_y \\ \sin(\alpha)\, s_x & \cos(\alpha) s_y \end{pmatrix}$$

Получаем следующую систему:

$$\begin{cases} \cos(\alpha)\, s_x = 0{,}2 \\ -\sin(\alpha) s_y = -0{,}26 \\ \sin(\alpha)\, s_x = 0{,}23 \\ \cos(\alpha) s_y = 0{,}22 \end{cases}$$

$$\begin{cases} \dfrac{\sin(\alpha)}{\cos(\alpha)} = tg(\alpha) = \dfrac{0{,}23}{0{,}2} \Longrightarrow \alpha \approx 48{,}99^\circ \\ \dfrac{\sin(\alpha)}{\cos(\alpha)} = tg(\alpha) = \dfrac{0{,}26}{0{,}22} \Longrightarrow \alpha \approx 49{,}76^\circ \end{cases}$$

Значения слегка различны, поэтому возьмем среднее: $\alpha = 49{,}375^\circ$
Вычислим значения матрицы масштабирования:

$$s_x = \frac{0{,}2}{\cos(\alpha)} = 0{,}307$$

$$s_y = \frac{0{,}26}{\sin(\alpha)} = 0{,}343$$

Для изменения папоротника необходимо пересчитать матрицу аффинных преобразований с углом $\tilde{\alpha} = 11^\circ$ (10° + номер по списку):

$$\Lambda = \begin{pmatrix} \cos(\tilde{\alpha})\, s_x & -\sin(\tilde{\alpha}) s_y \\ \sin(\tilde{\alpha})\, s_x & \cos(\tilde{\alpha}) s_y \end{pmatrix} = \begin{pmatrix} 0{,}981 \cdot 0{,}307 & -0{,}19 \cdot 0{,}343 \\ 0{,}19 \cdot 0{,}307 & 0{,}981 \cdot 0{,}343 \end{pmatrix}$$

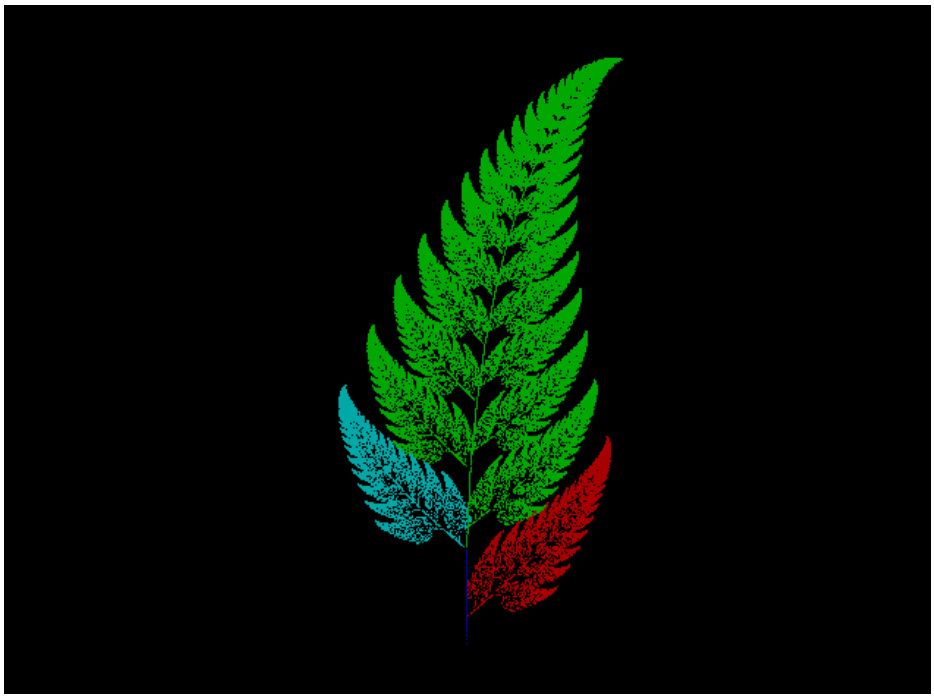
$$= \begin{pmatrix} 0{,}301 & -0{,}065 \\ 0{,}058 & 0{,}336 \end{pmatrix}$$

Теперь папоротник будет описываться следующим набором коэффициентов:

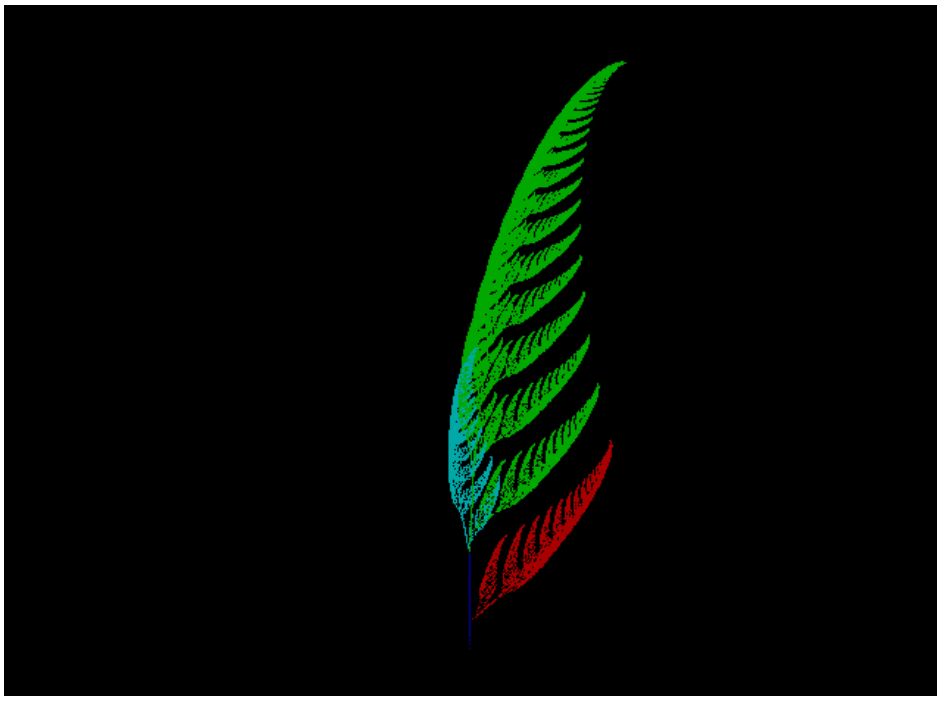
```
fern {
    0   0    0   .16 0      0 .01
   .85 .04 -.04  .85 0 1.6  .85
   .301  -.065  .058 .336 0 1.6 .07
  -.15 .28  .26  .24 0 .44  .07
  }
```

6. **Результат**

Исходный лист папоротника:



Левая ветвь под углом 11 градусов:



7. **Вывод**

С помощью фрактального сжатия можно добиться очень малого коэффициента сжатия (например, для изображений в статье Барнсли он достигал 1:10000), однако время на архивацию требуется примерно в 1000

раз больше, чем на алгоритм JPEG, но при этом разархивирование выполняется в 5-10 раз быстрее. Поэтому, если изображение будет сжато только один раз, а передано по сети и распаковано множество раз, то выгодней использовать фрактальный алгоритм. Вытеснение JPEG фрактальным алгоритмом в повсеместном использовании произойдет еще не скоро, однако в области приложений мультимедиа, в компьютерных играх его использование вполне оправдано.