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»

ОТЧЕТ ПО ПРОИЗВОДСТВЕННОЙ ПРАКТИКЕ

Студент Гурова Наталия Алексеевна
фамилия, имя, отчество

Группа ИУ7-44Б

Тип практики технологическая

Название предприятия МГТУ им. Н. Э. Баумана, каф. ИУ7

Студент _____ Гурова Н. А.
подпись, дата *фамилия, и.о.*

Руководитель практики _____ Куров А. В.
подпись, дата *фамилия, и.о.*

Оценка _____

Москва 2022 г.

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)

Кафедра «Программное обеспечение ЭВМ и информационные технологии» (ИУ7)

З А Д А Н И Е
на прохождение производственной практики

на предприятии ___МГТУ им. Н.Э. Баумана (кафедра ИУ7)_____

Студент _____Гурова Наталия Алексеевна ИУ7-44Б_____

(фамилия, имя, отчество; инициалы; индекс группы)

Во время прохождения производственной практики студент должен:

1) начать разработку программы для моделирования реалистичного изображения растений;
2) решить вопрос о способе представления объектов, проанализировать и выбрать алгоритмы для их обработки;
3) закрепить знания и навыки, полученные в ходе аудиторных занятий по пройденным курсам.

Дата выдачи задания « 01 » июля 2022 г.

Руководитель практики от кафедры _____ / __**Куров А. В.**____
(подпись, дата) (Фамилия И.О.)

Студент _____ / __**Гурова Н. А.**____
(подпись, дата) (Фамилия И.О.)

Оглавление

Введение

В современном мире компьютерная графика все больше проникает в обычную человеческую жизнь и используется в различных сферах. Типичные области ее применения – кинематография, компьютерные игры, наглядное отображение различных данных, а также моделирование экспериментов.

Одним из быстро развивающихся направлений компьютерной графики является моделирование и визуализация реалистичного трехмерного изображения. Для удовлетворения растущих потребностей в скорости синтеза и реалистичности полученного изображения разрабатываются новые алгоритмы и совершенствуются уже существующие.

Целью данной работы является обоснование выбора алгоритмов, которые можно использовать для получения реалистичного изображения трехмерной модели растения, их практическая реализация и адаптация (при необходимости) к условиям решаемой задачи.

1. Аналитическая часть

1.1 Формализация объектов моделируемой сцены

Сцена состоит из следующих объектов:

- Точечный источник света, представляющий собой материальную точку, из которой исходят лучи равномерно во всех направлениях.
- Модель растения, которая будет задаваться такими характеристиками как:
    1. Набор правил, описывающих в математическом виде (с помощью L-систем) процесс построения растения
    2. Размеры отдельных частей растения, таких как стебель и листья, соцветия
    3. Цвета отдельных частей растения

Функциональная модель решаемой задачи выглядит следующим образом

Рис. 1 Функциональная модель IDEF0 нулевого уровня

1.2 Анализ и выбор формы задания трехмерных моделей

Модель является отображением формы и размеров объекта.

В основном используются три вида моделей:

1. Каркасная (проволочная) модель

Такой тип модели использует информацию о вершинах и ребрах объектов. Это простейшая форма задания модели, так как мы храним минимум информации. Недостаток такого подхода состоит в том, что модель не всегда точно передает представление о форме объекта.

2. Поверхностная модель

Как следует из названия, в этой модели объекты задаются в виде набора плоскостей. Эта модель позволяет с достаточной точностью передать физические характеристики объекта, однако, если объект становится слишком сложным, могут возникнуть трудности с определением расположения материала внутри модели.

3. Объемная (твердотельная) модель

Схожа с поверхностной моделью, однако добавляется список внутренних нормалей, благодаря которому мы можем однозначно понять где находится материал модели.

Такая модель хранит исчерпывающую информацию об объекте, что хорошо при работе со сложными сценами, однако зачастую излишне.

Вывод

Для решения данной задачи были выбраны поверхностные модели, так как у каркасных моделей есть серьезный недостаток – неправильное восприятие формы, а объемные модели слишком информативны и ресурсоемки.

1.3 Анализ способа задания поверхностных моделей

Существует два распространённых способа задания поверхностных моделей:

Аналитическим способом. Этот способ задания модели характеризуется описанием модели объекта, которое доступно в неявной форме, то есть для получения визуальных характеристик необходимо дополнительно вычислять некоторую функцию, которая зависит от параметра;

Полигональной сеткой. Данный способ характеризуется совокупностью вершин, граней и ребер, которые определяют форму многогранного объекта в трехмерной компьютерной графике.

Возможные способы хранения информации о сетке:

- Список граней. Объект – это множество граней и множество вершин. В каждую грань входят как минимум 3 вершины;
- «Крылатое» представление. Каждая точка ребра указывает на две вершины, две грани и четыре ребра, которые её касаются;
- Полурёберные сетки. То же «крылатое» представление, но информация обхода хранится для половины грани;
- Таблица углов. Таблица, хранящая вершины. Обход заданной таблицы неявно задаёт полигоны. Такое представление более компактно и более производительно для нахождения полигонов, но, в связи с тем, что вершины присутствуют в описании нескольких углов, операции по их изменению медленны.
- Вершинное представление. Хранятся лишь вершины, которые указывают на другие вершины. Простота представления даёт возможность проводить над сеткой множество операций.

Решающим фактором выделяю скорость применения преобразований над объектами сцены.

Вывод

Для задания поверхностных моделей был выбран способ, использующий полигональную сетку. Подобное представление позволит избежать сложностей

при описании модели, а также даст возможность эффективно совершать преобразования над моделью за счет списка вершин.

1.4 Анализ и выбор алгоритма удаления невидимых ребер и поверхностей

Выбирая алгоритм удаления невидимых ребер и поверхностей, в первую очередь нужно выделить свойства, которыми должен обладать итоговый алгоритм, чтобы обеспечить оптимальную работу программы и реалистичное изображение. При анализе были выделены следующие свойства:

- алгоритм должен быть достаточно быстрым
- алгоритм должен использовать мало памяти
- алгоритм должен иметь высокую реалистичность изображения

Алгоритм, использующий Z-буфер

Идея: для построения сцены используются два буфера: буфер кадра, в котором хранятся атрибуты каждого пикселя, и Z-буфер, в котором хранится информация о координате Z для каждого пикселя.

Изначально в Z-буфере находятся минимальные значения Z, а в буфере кадра - информация о пикселях, которые описывают фон. Глубина каждого нового пикселя при подсчете сравнивается со значением, которое уже есть в Z-буфере. В случае, если новый пиксель расположен ближе к наблюдателю, чем предыдущий, то информация о нем заносится в буфер кадра и происходит редактирование Z-буфера.

Преимущества:

- простота реализации
- высокая производительность

Недостатки:

- большой объем требуемой памяти
- сложная реализация прозрачности

Алгоритм обратной трассировки лучей

Идея: наблюдатель видит объект с помощью испускаемого света, который согласно законам оптики, доходит до наблюдателя некоторым путем. Отслеживать пути лучей от источника к наблюдателю неэффективно с точки зрения вычислений, поэтому наилучшим способом будет отслеживание лучей в обратном направлении, то есть от наблюдателя к объекту.

Преимущества:

- высокая реалистичность синтезируемого изображения
- вычислительная сложность слабо зависит от сложности сцены

Недостатки:

- низкая производительность

Алгоритм Робертса

Данный алгоритм работает в объектном пространстве, решая задачу только с выпуклыми телами.

Алгоритм выполняется в три этапа:

1. Этап подготовки исходных данных (информации о телах).
2. Этап удаления ребер, экранируемых самим телом.
3. Этап удаления невидимых ребер, экранируемых другими телами сцены.

Преимущества:

- работа в объектном пространстве
- высокая точность вычислений

Недостатки:

- все тела должны быть выпуклыми
- рост сложности алгоритма – квадрат числа объектов
- сложность реализации.

Алгоритм художника

Идея: сначала рисуем дальние объекты, затем более близкие (подобно тому, как художник рисует картину). Наиболее распространенная реализация - сортировка по глубине (произвольное множество граней сортируется по ближнему расстоянию от наблюдателя), затем сортированные грани выводятся на экран в порядке от самой дальней до самой ближней.

Преимущества:

- требует меньшей памяти, чем, например, алгоритм Z-буффера

Недостатки:

- реалистичность изображения недостаточно высока
- сложность реализации при пересечении граней на сцене

Вывод

Для удаления невидимых линий был выбран алгоритм Z-буфера. Данный алгоритм работает достаточно быстро из-за отсутствия сортировок, позволяет добиться хорошей реалистичности, а также достаточно прост в реализации.

1.5 Анализ и выбор модели освещения

Существует два вида моделей материалов: физические модели и эмпирические.

Физические модели материалов стараются аппроксимировать свойства некоторого реального материала. Такие модели учитывают или особенности поверхности материала или поведение частиц материала.

Эмпирические модели материалов устроены иначе. Подобные модели подразумевают некий набор параметров, не имеющих физической интерпретации, но позволяющих с помощью подбора получить нужный вид модели.

Рассмотрим эмпирические модели, а конкретно модель Ламберта и модель Фонга.

Модель Ламберта

Модель Ламберта моделирует идеальное диффузное освещение (свет при попадании на поверхность рассеивается равномерно во все стороны). При такой модели освещения учитывается только ориентация поверхности (N) и направление источника света (L) (рисунок 2).

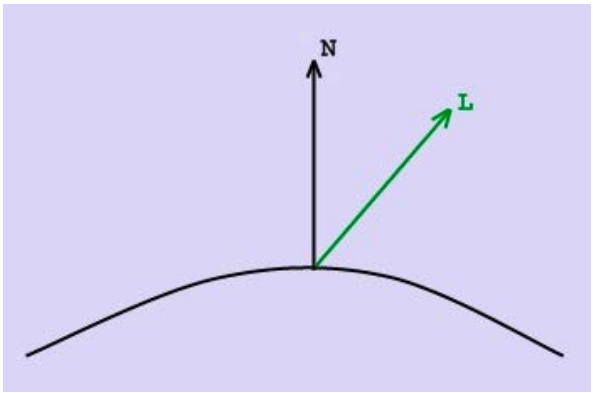


Рис. 2 Модель Ламберта

Эта модель является одной из самых простых моделей. Она удобна для анализа свойств других моделей, так как ее легко выделить из любой модели и анализировать оставшиеся составляющие.

Модель Фонга

Модель Фонга представляет собой комбинацию диффузной и зеркальной составляющих. Падающий и отраженный лучи лежат в одной плоскости с нормалью к отражающей поверхности в точке падения (рисунок 2). Нормаль делит угол между лучами на две равные части. L – направление источника света, R – направление отраженного луча, V – направление на наблюдателя.

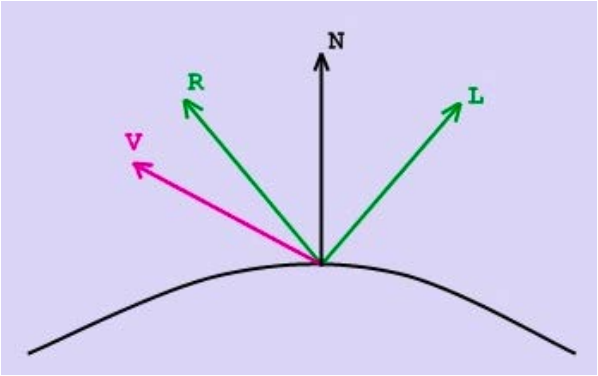


Рис. 3 Модель Фонга

Работает модель таким образом, что кроме равномерного освещения на материале могут появляться блики. Местонахождение блика на объекте определяется из закона равенства углов падения и отражения. Чем ближе наблюдатель к углам отражения, тем выше яркость соответствующей точки.

Вывод

Для освещения была выбрана модель Ламберта, так как по сравнению с моделью Фонга она требует меньше вычислительных затрат, и, следовательно, работает быстрее.

2. Конструкторская часть

2.1 Общий алгоритм решения поставленной задачи

1. Задать набор правил для построения растения
2. Рассчитать параметры поверхностей отдельных частей растения (стебель, листья и т.д.) согласно установленным правилам
3. Соединить полученные поверхности в единую модель
4. Изобразить растение на сцене

2.2 Алгоритм удаления невидимых поверхностей с Z-буфером.

Формальное описание алгоритма:

1. Заполнить буфер кадра фоновым значением интенсивности
2. Заполнить z-буфер минимальным значением z
3. Для каждого пиксела (x, y) грани вычислить его глубину z(x, y) и сравнить вычисленное значение со значением z-буфер(x, y), хранящимся в z-буфере в этой же позиции
4. Если z(x, y) > z-буфер(x, y), то записать атрибут этого многоугольника (интенсивность, цвет и т.п.) в буфер кадра и заменить z-буфер(x, y) на z(x, y).

Блок-схема алгоритма представлена на рисунке 4.

2.3 Модель освещения Ламберта

Данная модель вычисляет цвет поверхности в зависимости от того как на нее светит источник света. Согласно данной модели, освещенность точки равна произведению силы источника света и косинуса угла, под которым он светит на точку.

Блок-схема алгоритма простой закраски, основанного на принципах модели освещения Ламберта, представлена на рисунке 5.

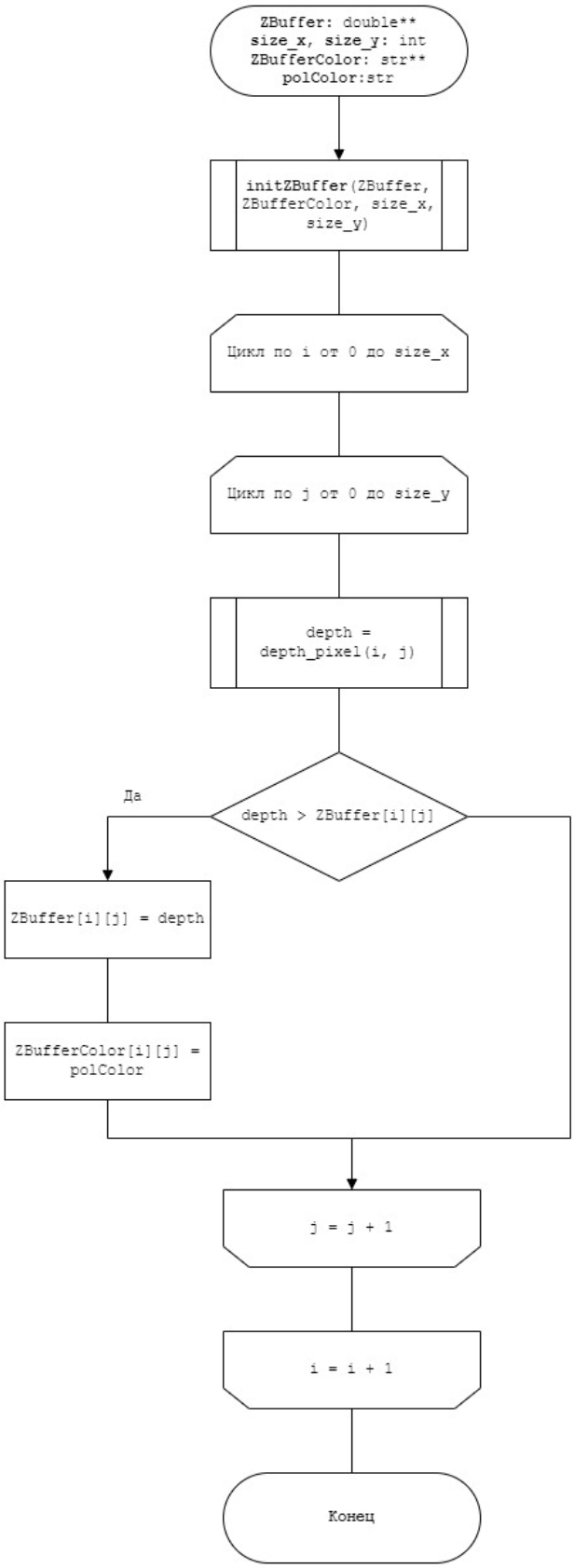


Рис.4 Схема алгоритма удаления невидимых поверхностей с Z-буфером

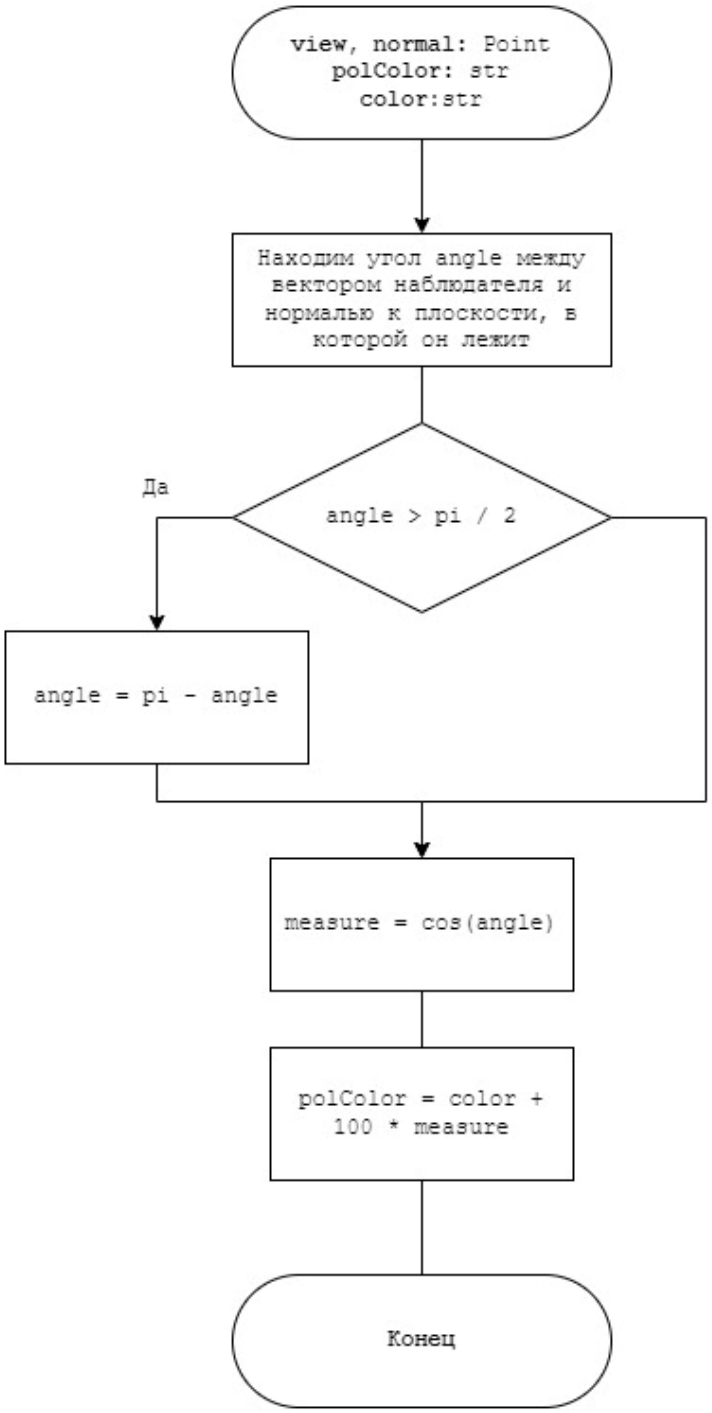


Рис. 5 Схема алгоритма простой закраски

2.4 Генерация растения

Для генерации различных растений будем использовать L-системы.

Lindenmayer System или просто L-system – это математическая модель, предложенная в 1968 году венгерским биологом и ботаником Аристидом Линденмайером, для изучения развития простых многоклеточных организмов. Позже эта модель была расширена и стала использоваться для моделирования сложных ветвящихся структур – разнообразных деревьев и цветов.

Основная идея L-систем – постоянная перезапись элементов строки (rewriting). Другими словами, это способ получения сложных объектов путем замены частей простого начального объекта по некоторым правилам.

Подобный подход удобен для описания растений, так как они обладают свойствами рекурсивности и самоподобия при росте. Например, маленькие листочки, являющиеся частью большого взрослого составного листа, имеют ту же форму, что весь лист имел на раннем этапе формирования.

Язык L-систем очень прост, он состоит из символов (алфавита) и продукционных правил. Первое состояние предложения называется аксиомой. К этой аксиоме можно применить продукционные правила для эволюции или роста системы.

Например, если у нас есть система с аксиомой A и единственным правилом A -> ABB, то после первой итерации предложение сменится на ABB, потом на ABBBB и т.д.

L-системы имеют свою классификацию от простых до сложных и мощных:

- Детерминированные контекстно-свободные L – самый простой вид L-систем. Пример такой системы представлен на рисунке 6.
- Параметрические L-системы – каждому символу добавляем переменную (возможно не одну), которая позволяет указывать некоторые параметры для этого символа (возможно величину угла поворота, длину шага, толщину линии и т.п.)

Пример такой системы представлен на рисунке 7.

- Контекстно-зависимые L-системы – принимаем во внимание окружение заменяемого символа.
- Стохастические L-системы – добавляем простым системам возможность задания вероятности выполнения того или иного правила. Подобный подход вносит некоторый элемент случайности в получающиеся структуры.

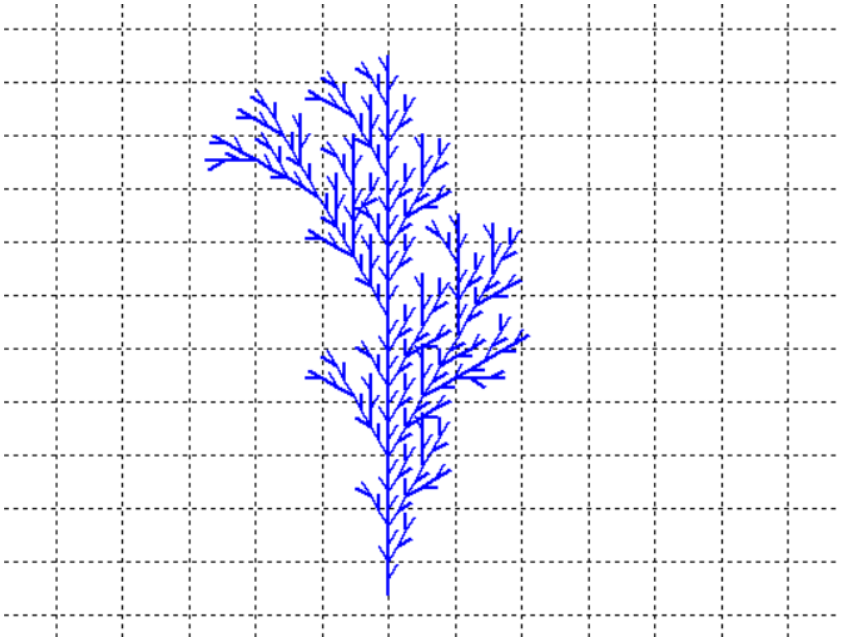


Рис. 6 axiom = ‘F’, rule = ‘F: F[-F]F[+F][F]’, n = 3, delta = 30

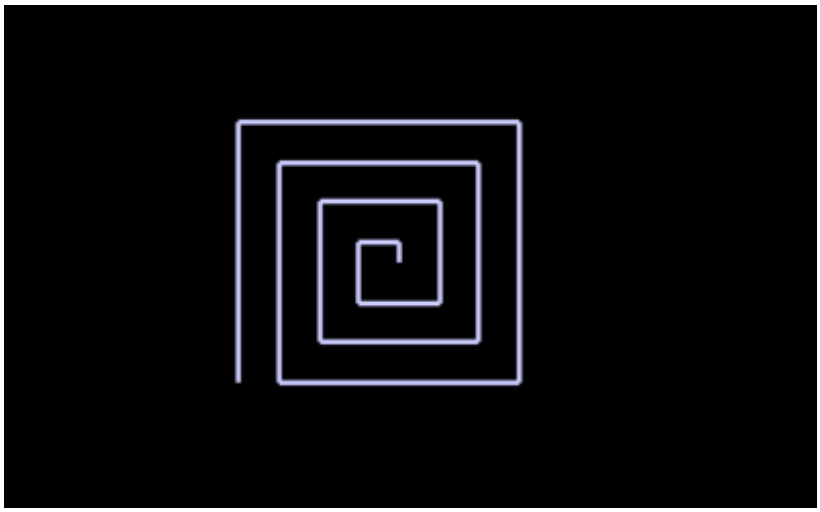


Рис. 7 axiom = ‘BA(0, 0)’, rules = [‘A(x, y) x < y: BA(x + 1, y)’, ‘A(x, y) x >= y: +BA(0, y + 1)’], n = 118, delta = 90

Для визуализации предложения используется система рендеринга, называемая “черепашьей графикой”.

Черепашья графика характеризуется размещением в декартовой системе “черепашки” и передачей ей инструкций. Черепашка движется в соответствии с полученными ею инструкциями и оставляет за собой след. В нашем случае черепашке отправляется каждый символ из предложения L-системы. Ключ к черепашьему языку представлен в таблице 1.

[A..Z]	Любая (не являющаяся константой буква алфавита перемещает черепашку вперед на фиксированное расстояние, перемещаясь черепашка рисует линию)
[a..z]	Переместить черепашку на фиксированное число шагов, не рисуя линию
+	Поворот черепашки вправо на фиксированный угол
-	Поворот черепашки влево на фиксированный угол
/	Крен черепашки вправо на фиксированный угол
\	Крен черепашки влево на фиксированный угол
^	Тангаж черепашки вверх на фиксированный угол
_	Тангаж черепашки вниз на фиксированный угол
[	Запись текущего состояния черепашки в стек
]	Извлечение состояния черепашки из стека и присвоение этого состояния
~	Нарисовать поверхность, идентифицированную модулем сразу после ~ в текущем местоположении и ориентации черепахи.
{	Создать пустой полигон и положить его в стек полигонов (для рисования поверхностей)

G	Переместить черепашку вперед на фиксированное расстояние и провести линию, не добавляя вершину к текущему многоугольнику.
g	Переместить черепашку вперед, не проводя линию и не добавляя вершину к текущему многоугольнику.
.	Добавить положение черепахи к текущему многоугольнику
}	Извлечь текущий полигон из стека полигонов и нарисовать его, используя указанные вершины.

Таблица 1. Алфавит для отрисовки трехмерной системы Линденмайера.

Пример описания модели растения с помощью различных L-систем можно увидеть на рисунках 8 и 9.

Итоговый алгоритм генерации растения выглядит следующим образом:

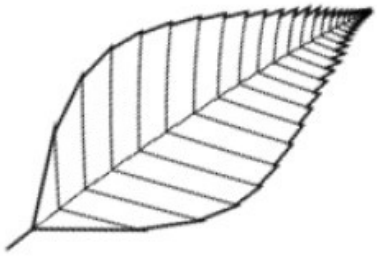
1. Получить параметры растения, представленного в виде L-системы (аксиому, набор продукционных правил, количество итераций роста, начальные данные, длину шага и т.д.)
2. Используя количество итераций, аксиому и правила получить предложение, по которому будет строиться итоговое растение.
3. Если в предложении есть вхождения некоторых поверхностей, убедится, что их представления заданы в программе или сгенерировать их.
4. “Прочитать” предложение с помощью черепашки и таблицы-расшифровки черепашьего языка
5. Получить итоговую модель растения, которую уже можно будет выводить на экран.

n=10, $\delta = 60°$

```
#include K /* flower shape specification */

ω  :  A~K
p1 :  A     :  *       → [-/~K][+/~K]I(0)/(90)A
p2 :  I(t)  :  !(t=2)  → FI(t+1)
p3 :  I(t)  :  t=2     → I(t+1)[-FFA][+FFA]
```

Рис. 8 L-система, описывающая сирень



n=25, δ=60°

```
#define LA 5      /* initial length - main segment */
#define RA 1.15   /* growth rate - main segment */
#define LB 1.3    /* initial length - lateral segment */
#define RB 1.25   /* growth rate - lateral segment */
#define LC 3      /* initial length - marginal notch */
#define RC 1.19   /* growth rate - marginal notch */

ω  :  [{A(0,0).}][{A(0,1).}]
p1 :  A(t,d)    :  d=0  →  .G(LA,RA).[+B(t)G(LC,RC,t).}]
                           [+B(t){.]A(t+1,d)
p2 :  A(t,d)    :  d=1  →  .G(LA,RA).[-B(t)G(LC,RC,t).}]
                           [-B(t){.]A(t+1,d)
p3 :  B(t)      :  t>0  →  G(LB,RB)B(t-1)
p4 :  G(s,r)    :  *    →  G(s*r,r)
p5 :  G(s,r,t)  :  t>1  →  G(s*r,r,t-1)
```

Рис. 9 L-система, описывающая лист, как часть растения

3. Технологическая часть

3.1 Выбор языка программирования и среды разработки

Для разработки данной программы был выбран язык Python. Данный выбор обусловлен следующими факторами:

- Этот язык предоставляет программисту широкие возможности реализации самых разнообразных алгоритмов.
- Python является объектно-ориентированным языком, что позволяет программисту естественным образом декомпозировать задачу и легко модифицировать программу в случае необходимости.
- Существует множество учебной литературы по этому языку.

В качестве среды разработки был выбран PyCharm. Некоторые факторы, по которым была выбрана данная среда:

- Включает весь основной функционал: отладчик, поддержка точек останова и т.д.
- Имеет удобный редактор кода со всеми полезными функциями: подсветкой синтаксиса, автоматическим форматированием, дополнением и отступами.
- Имеет удобный интерфейс для работы с git.
- Имеет большое количество плагинов.
- Данная среда разработки бесплатна для студентов.

3.2 Структура классов программы

Описание классов в программе:

- Primitive – класс, реализующий основные функции 3D-модели (трехмерные преобразования)
- Plant – класс, реализующий работу с растением
- PlantGenerate – класс, отвечающий за генерацию растения

- Camera – класс для работы с камерой. Хранит позицию камеры, угол ее наклона, скорость перемещения.
- Field – класс для работы со сценой. Характеризует набор объектов и их свойств.
- Shadow – класс, позволяющий работать с тенями
- Light – класс, позволяющий работать с источником света
- InputForm – класс для взаимодействия с пользователем
- FileWork – класс работы с файлами (ввод и вывод параметров растений, сохранение сцены как изображения)
- MainWindow – основной класс для работы с приложением.

3.3 Диаграмма классов

Диаграмма классов для данной работы представлена на рисунке 6.

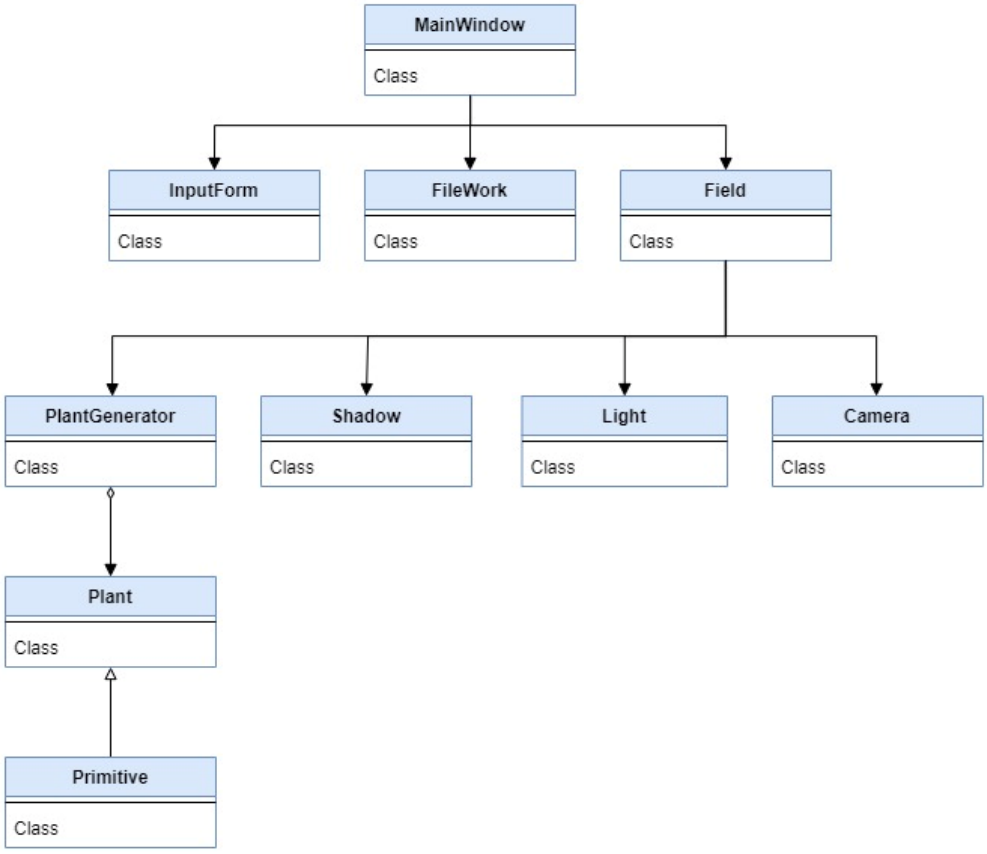


Рис. 6 Диаграмма классов

3.4 Интерфейс программного обеспечения

Формы ввода параметров растения:

- Initial conditions – ввод начальных координат x, y, z, откуда будет нарисовано растения, а также угла alpha – начальное направление роста растения.
- Plant parameters – ввод параметров растения:
  - axiom, rules – правила, задающие как генерировать растение
  - lengthStep – длина одного шага при генерации
  - delta – угол поворота при генерации
  - n – количество итераций для генерации
  - types – типы стандартных поверхностей для стебля, листьев и цветов
- Appearance – внешний вид растения и окружающей среды
  - Bg – цвет фона
  - Colors – основные цвета для растения

Отдельные кнопки:

- Load – считать параметры растения из файла.
- Save – записать текущие параметры растения в файл.
- Print Screen – сохранить текущую сцену как svg-изображение
- Draw – сгенерировать растение и вывести полученное изображения на экран
- Clean – очистка сцены

Меню:

- File – выпадающий список из load, save, printScreen, clean, описанных выше.
- Settings – выпадающий список, включающий возможность включить координатную сетку, а также поменять цвета картинки.

- Info – выпадающий список, включающий информацию о программе, информацию об авторе, краткую справку.
- Exit – выход из программы.

Также есть возможность взаимодействовать с программой при помощи стрелок вправо/влево/вверх/вниз на клавиатуре и при помощи мышки (нажать на сцену и, не отпуская кнопку, плавно ввести курсор вправо/влево). Интерфейс представлен на рисунке 7.

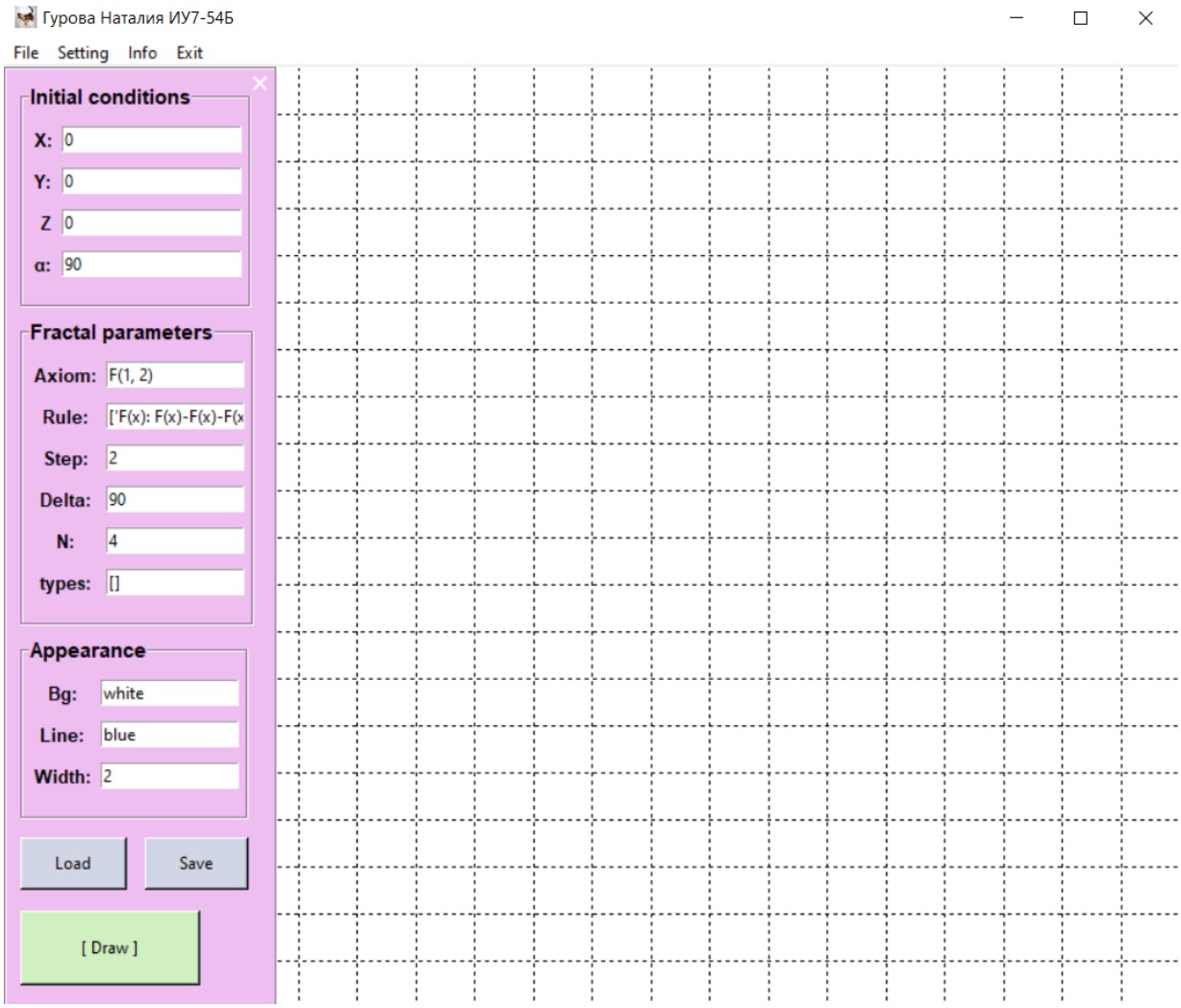


Рис. 7 Интерфейс

Заключение

Во время выполнения поставленной задачи были проанализированы основные способы представления и задания трехмерных моделей, а также рассмотрены основные алгоритмы удаления невидимых линий и методы освещения. Проанализированы достоинства и недостатки представленных алгоритмов и выбраны наиболее оптимальные для решения поставленной задачи.

Проделанная работа помогла закрепить полученные навыки в области компьютерной графики и проектирования программного обеспечения.

Литература

1. Роджерс Д., Алгоритмические основы машинной графики: пер. с англ.— М.: Мир, 1989.

2. Авдеева С.М., Куров А.В. Алгоритмы трехмерной машинной графики: учебное пособие. - М.: Издательство МГТУ им. Н.Э. Баумана, 1996.

3. Шикин Е.В., Боресков А.В. Компьютерная графика. Динамика, реалистические изображения. – М.: Диалог-МИФИ, 1995.

4. Przemyslaw Prusinkiewicz , Aristid Lindenmayer A. The alocithmic beauty of plants. [Электронные ресурс]. – Режим доступа: http://algorithmicbotany.org/papers/abop/abop.pdf, свободный - 2004

5. James Scott Hanan. Parametric L-systems and their application to the modelling and visualization of plants [Электронные ресурс]. – Режим доступа: http://algorithmicbotany.org/papers/hanan.dis1992.pdf, свободный - 1992