Задание на курсовой проект

Разработать программу моделирующую жизненный цикл дерева. В каждый определенный промежуток времени дерево будет представлять собой полноценный 3Д-объект. Пользователь должен иметь возможность наблюдать за ростом дерева как в прямом, так и в обратном порядке; замедлять рост дерева, и, наоборот, ускорять его. В любой промежуток времени пользователь должен иметь возможность приостановить процесс жизненного цикла дерева и посмотреть, как оно выглядит в данный момент.

Сам процесс моделирования жизненного цикла дерева будет достигаться путем смены одной 3Д-модели на другую через каждую единицу времени - в программе будет храниться массив 3Д-объектов на каждый определенный промежуток времени. Пользователь должен иметь возможность задавать точку и направление обзора, управлять одним источником освещения. Источник задается своим направлением света и интенсивностью.

Содержание

Введение

В настоящее время с помощью компьютерной графики реализовано большое количество способов моделирования деревьев. Моделирование деревьев широко используется в разработке компьютерных игр, в реализации спецэффектов в фильмах, в различных биологических исследованиях. Однако большая часть исследований, посвященных моделированию деревьев, происходит за рубежом и не переведено на русский язык [1].

Целью данного курсового проекта является анализ методов моделирования трехмерных изображений, а также методов генерации реалистичных деревьев, их сравнение и реализация наиболее подходящего из них для решения поставленного индивидуального задания.

Для реализации поставленной цели должны быть выполнены следующие задачи:

1) анализ существующих решений по моделированию деревьев;
1) анализ этапов моделирования 3D-изображения;
1) выбор наиболее эффективных алгоритмов моделирования 3D-изображения;
1) исследование и реализация геометрических преобразования для 3D-изображений;
1) реализация всей модели в приложении.

1 Аналитический раздел

В данном разделе будет произведён анализ предметной области и выбор оптимальных методов и алгоритмов, необходимых для решения поставленной задачи.

1.1 Модель представления объектов сцены

Выбор модели представления объектов на сцене играет важную роль, так как от этого зависит, какие алгоритмы можно использовать в программе. Большинство трехмерных алгоритмов работают исключительно с многоугольниками, поэтому для описания формы многогранного трёхмерного объекта удобно использовать полигональную сетку, которая состоит из вершин, рёбер и граней. На рисунке 1 продемонстрирован пример трехмерного объекта, представленного в виде треугольной полигональной сетки.

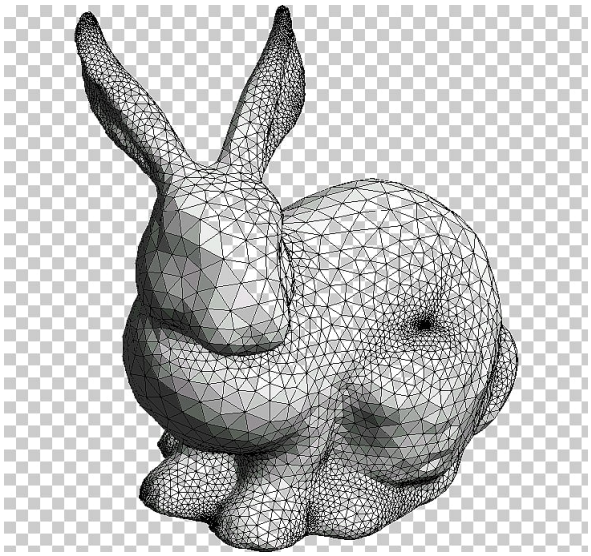


Рис. 1: Пример треугольной полигональной сетки

Гранями полигональной сетки обычно являются треугольники, четырёхугольники или другие простые выпуклые многоугольники. У треугольника есть ряд особенностей, которые делают его наиболее подходящим для применения в моделировании объектов:

1) любой треугольник является выпуклым, поэтому разбиение тел на треугольники упрощает рендеринг;

2) любой многоугольник может быть разбит на некоторое определённое количество треугольников;

3) в треугольнике относительно легко можно восстановить нормаль к исходной поверхности, что актуально для алгоритмов закраски;

4) невырожденный треугольник всегда задает одну плоскость и расположен в одной плоскости.

Существуют самые различные способы хранения полигональной сетки трёхмерного объекта. Но так как основная цель данной работы заключается в изучении моделирования реалистичных деревьев, то полигональная сетка будет представлять из себя массив полигонов - треугольников, задающимися координатами вершин.

1.2 Алгоритмы построения деревьев

Моделирование объектов природы, в частности лесной растительности, является нетривиальной задачей для большинства известных пакетов 3D-моделирования. В традиционных CAD-системах мы можем получать объекты строгой геометрической формы, которые достаточно хорошо описываются математическими моделями. Объектам живой природы присуща стохастичность в их структуре и огром -ное количество параметров (свет, ветер, гравитация, тип ветвления, осадки), влияющих на их внешний вид. Все это доказывает то, что при моделировании растительности необходимо использовать собственные методы, алгоритмы и подходы.

Все существующие подходы к моделированию деревьев и растений можно условно разделить на три группы:

- по определенным математическим правилам и грамматикам;
- на основе исходного скелета (эскиза) объекта;
- на основе исходного изображения объекта.

Критерием разбиения служит тип входных данных, используемых при моделировании.

При моделировании по определенным математическим правилам и грамматикам входными данными является набор числовых параметров: количество итераций, углы ветвления, радиус ствола, ветвей, коэффициенты уменьшения радиусов и длины ветвей и пр. По данному набору параметров в соответствии с законами ветвления моделируется дерево. Простейшим законом ветвления является модель Хонды.

1.2.1 Модель Хонды

Основная идея модели Хонды состоит в следующем:

- сегменты дерева прямые, площадь их поперечного сечения не рассматривается;

- в течение итерации материнский сегмент производит два дочерних;
- длина двух дочерних сегментов короче материнского в $r1$ и $r2$ раз;
- материнский сегмент и два его дочерних находятся в одной плоскости ветвления. Дочерние сегменты выходят из материнского под углами ветвления $\alpha 1$ и $\alpha 2$
- в связи с действием гравитационной силы, плоскость ветвления является «ближайшей к горизонтальной плоскости», иными словами, линия, перпендикулярная материнскому сегменту и лежащая в плоскости ветки — горизонтальная. Исключение делается для веток, присоединенных к главному стволу. В этом случае используется постоянный угол расхождения α.

На рисунке 2 продемонстрировано ветвление веток из ствола дерева, а также ветвление произвольной ветки.

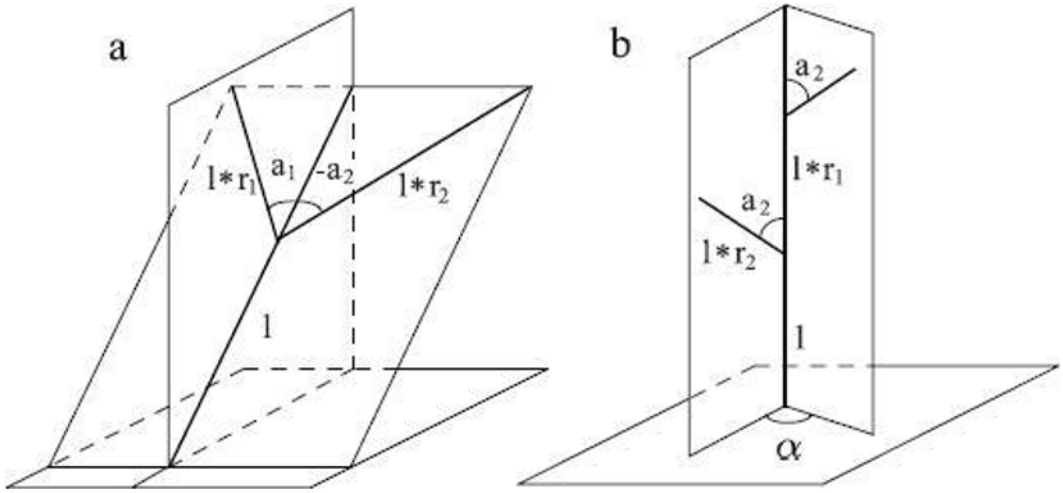


Рис. 2: Ветвление дерева согласно модели Хонды

1.2.2 Модель Эоно и Кьюниай

Модель Эоно и Кьюниай является продолжением модели Хонды. Они предложили несколько улучшений, самым важным из которых был поворот сегментов в определенных направлениях, соответствующих стремлению веток к солнцу, учёту ветра и гравитации. Эоно и Кьюниай использовали прямые линии постоянной или переменной ширины для построения «древесного скелета».

1.2.3 Стохастические модели

Эти модели построены по-другому, они разделяют общую парадигму описания структуры деревьев, в частности, расчёт возможностей формирования веток.

1.2.4 Применение L-systems

В 1968г. Венгерский биолог и ботаник Аристид Линденмайер предложил математическую модель для изучения развития простых многоклеточных организмов, которая позже была расширена и используется для моделирования сложных ветвящихся структур — разнообразных деревьев и цветов. Эта модель получила название L-System. Основная идея L — постоянная перезапись (rewriting) элементов строки. Другими словами, получение сложных объектов путем замены частей простого начального объекта по некоторым правилам. Классическим примером является снежинка. На рисунке 3 initiator — это начальный объект, грани которого заменяются на generator. Далее с новым объектом проделывается то же самое.

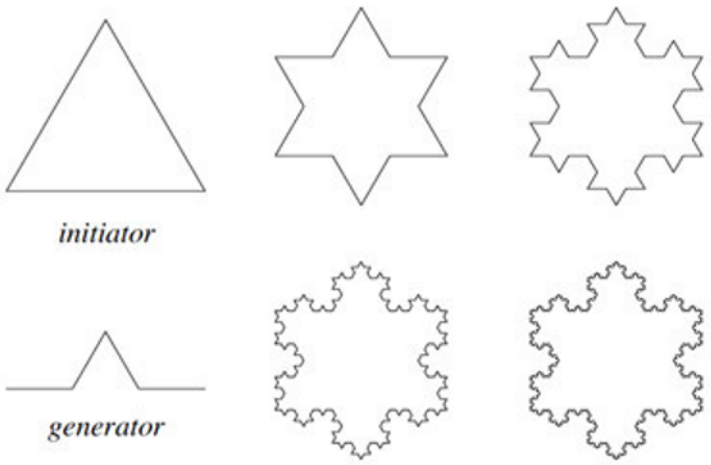


Рис. 3: Классический пример L - снежинка

Проводя аналогию с фракталами, можно сказать, что L оперирует со строкой символов по специальным правилам, начиная с первоначальной простой аксиомы.

По сути L является грамматикой, но фундаментальное отличие L от формальных грамматик состоит в том, что правила применяются одновременно ко всей строке, к каждому символу, плюс, нет понятий терминальных и нетерминальных символов. То есть «вывод» по этой грамматике может продолжаться бесконечно. Одновременность применения правил очень быстро становится понятна, если учесть откуда пришла эта модель. В биологии каждая клетка растет, делится и развивается параллельно во времени.

Применение L-systems для генерирования деревьев было впервые рассмотрено Эоно и Кьюниай. Для начала, основываясь на формальном определении L, они доказали её непригодность для моделирования высших растений. Но доказательство не относилось к параметрическим

L-systems с черепашьей интерпретацией строк. К примеру, L-systems на рисунке 1 выполняет те модели Хонды, в которых один из углов ветвления равен нулю, податливая моноподиальная структура и ясно выраженные главная и боковые оси.

Таким образом, технология L-systems могут играть важную роль как инструмент для биологически-корректного моделирования деревьев и синтеза фотореалистичных изображений.

1.2.5 Обоснование выбора алгоритма генерирования каркаса дерева

Поскольку главной целью данной курсовой работы является анализ моделирования различных 3Д-изображений, то для реализации поставленной задачи используется метод Хонды. В дальнейшем, программу можно будет усовершенствовать, добавляя в нее другие, более реалистичные методы генерации деревьев.

1.3 Генерация жизненного цикла дерева на основе полученного каркаса

1.3.1 Задание индивидуальных характеристик дерева

Для того, чтобы иметь возможность генерировать различные виды деревьев, алгоритм генерации деревьев методом Хонда обладает следующими индивидуальными характеристиками дерева:

- точка, из которой будет расти дерево;
- угол между двумя главными ветками дерева;
- угол между веткой и исходящей из нее веткой;
- отношение длины ветки к длине исходящей из нее ветки;
- точка на ветке, из которой будет расти следующая ветка;
- длина ствола дерева;
- радиус ствола;
- радиус веток;
- цвет дерева.

Пояснения к характеристикам:

Главными ветками называются ветки, растущие из ствола дерева.

Точка на ветке, из которой будет расти следующая ветка, рассчитывается в алгоритме, используя параметр $start$, определяемый пользователем. Он равен отношению длины ветки к длине части этой ветки. Часть ветки определяет точку, из которой будет расти новая ветка.

Радиус каждого последующего конуса будет вычисляться так, чтобы самая широкая часть этого конуса (ветка дерева) не выходила за границы того конуса, из которого она исходит.

1.3.2 Генерация массива каркасов дерева для каждого момента времени

Для моделирования процесса роста дерева было решено уменьшать пропорционально индивидуальные характеристики дерева, а именно: радиус ствола, длину ствола, угол между главными ветками, углы между веткой и исходящей из нее веткой, до тех пор пока они не станут меньше заранее заданного минимального значения.

Таким образом будет сгенерировано определенное количество деревьев с пропорционально уменьшающимися параметрами.

1.4 Построение 3Д-модели на основе получившегося каркаса дерева

Для визуализации получившегося каркаса дерева каждая ветка и ствол дерева будут представляться в виде конусов, задающихся следующими параметрами: радиусом основания, центром основания, а также вершиной конуса.

Он же в свою очередь будет разбит на определенное количество полигонов (треугольников) 1.1, задаваемое пользователем.

1.5 Удаление невидимых поверхностей

Одной из наиболее сложных задач в машинной графике является удаление невидимых линий и поверхностей. Все алгоритмы, решающие данную задачу, включают в себя процесс сортировки. Это либо сортировка по расстоянию от точки наблюдения до тела, либо сортировка, основанная на загораживании тел друг другом. Эффективность алгоритмов в большой степени зависит от эффективности алгоритма сортировки.

Существует связь между скоростью работы алгоритма и детальностью его результата – ни один алгоритм не может достигнуть хороших показателей в том или другом одновременно.

Алгоритмы делятся на 2 типа:

1) работающие в объектном пространстве;

2) работающие в пространстве изображений.

Алгоритмы, работающие в объектном пространстве использующте мировую системму координат, обладают высокой точностью. А у алгоритмов, работающих в пространстве изображений система координат связана с дисплеем, и поэтому точность ограничена разрешающей способностью дисплея. Сложность таких алгоритмов составляет $\sim$O(N^2) и $\sim$O(N$\star$M) соответственно, при количество объектов - N и количестве пикселей дисплея - M. Но несмотря на то, что $N^2 <$ N$\star$M, алгоритмы, работающие в пространстве изображений могут быть эффективнее в силу того, что при растровой реализации близко расположенные пиксели часто обладают одинаковыми свойствами [4]. Рассмотрим наиболее известные алгоритмы удаления невидимых поверхностей.

1.5.1 Алгоритм Робертса

Алгоритм Робертса позволяет произвести удаление невидимых линий при помощи математических вычислений. Алгоритм работает в объектном пространстве и состоит из следующих этапов:

1) подготовка исходных данных;
2) удаление невидимых линий, экранируемых самим телом (для одного тела на этом работа заканчивается);
3) удаление линий, экранируемых другими телами;
4) удаление линий пересечения тел, экранируемых самими телами, связанными отношением протыкания, и другими телами.

Недостатки алгоритма:

1) для работы алгоритма Робертса необходимо, чтобы все объекты сцены были выпуклыми, если имеются невыпуклые тела, необходимо сначала разбить их на выпуклые;
2) вычислительная трудоемкость алгоритма Робертса растет теоретически, как квадрат числа объектов;
3) невозможна реализация теней без использования посторонних методов.

Достоинством алгоритма является точность и быстрота работы математических методов [5].

1.5.2 Алгоритм Варнока

В пространстве изображения рассматривается окно и решается вопрос о том, пусто ли оно, или его содержимое достаточно просто для визуализации. Если содержимое слишком сложно, то окно разбивается

на фрагменты до тех пор, пока содержимое фрагмента не станет достаточно простым для визуализации или его размер не достигнет требуемого предела разрешения. В последнем случае информация, содержащаяся в окне, усредняется, и результат изображается с одинаковой интенсивностью или цветом. Пример работы алгоритма Варнока представлен на рис. 4.

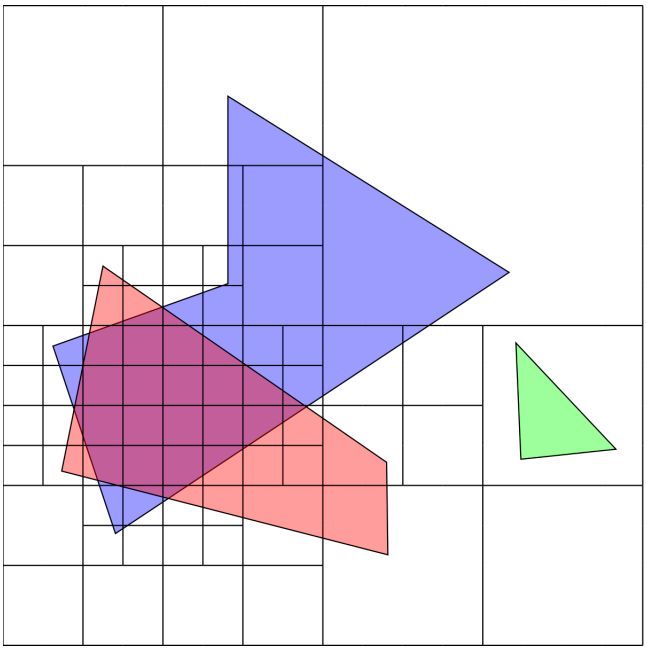


Рис. 4: Пример работы алгоритма Варнока

Недостаток алгоритма Варнока заключается в том, что в случае, когда невозможно сразу определить параметры закраски области, разбиение области может проводиться многократно, до тех пор, пока размер области не достигнет одного пиксела. При этом для каждой из полученных подобластей применяется процедура принятия решения.

Алгоритм достаточно прост с точки зрения понимания, однако может потребоваться большое количество разбиений, соответственно потребуется много времени на анализ и отображение содержимого всего окна. В случае отрисовки сложных сцен со множеством объектов алгоритм становится неэффективным.

1.5.3 Алгоритм, использующий Z-буфер

Данный алгоритм является одним из простейших алгоритмов удаления невидимых линий и поверхностей. Алгоритм работает в пространстве изображений.

Идея Z-буфера является простым обобщением идеи о буфере кадра.

Буфер кадра используется для запоминания атрибутов (интенсивности) каждого пиксела в пространстве изображения. Z-буфер - это отдельный буфер глубины, используемый для запоминания координаты Z или глубины каждого видимого пиксела в пространстве изображения. В процессе работы глубина или значение Z каждого нового пиксела, который нужно занести в буфер кадра, сравнивается с глубиной того пиксела, который уже занесен в Z-буфер. Если это сравнение показывает, что новый пиксел расположен впереди пиксела, находящегося в буфере кадра, то новый пиксел заносится в этот буфер и, кроме того, производится корректировка Z-буфера новым значением Z. Если же сравнение дает противоположный результат, то никаких действий не производится. Вычислительная сложность данного алгоритма, которая равна O(N*M*K), где M*N – количество пикселей в буфере кадра, K – количество многоугольников. На рис.5 представлен пример работы алгорима, использующего z-буфер.

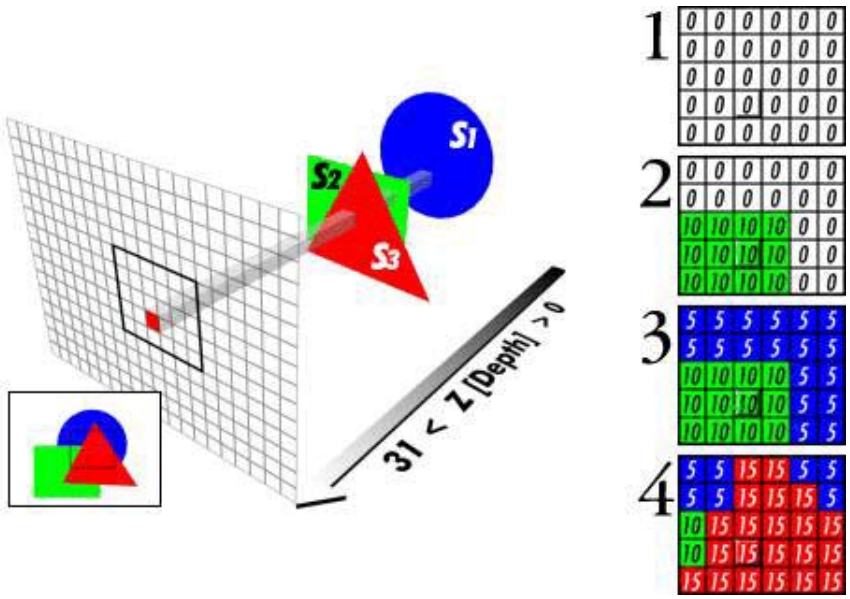


Рис. 5: Пример работы алгоритма, использующего z-буфер

Главное преимущество алгоритма – его простота. Также, этот алгоритм делает тривиальной визуализацию пересечений поверхностей. Кроме того, объекты сцены могут рассматриваться в произвольном порядке, что позволяет не тратить время на сортировку объектов по глубине.

Недостатком алгоритма является большой объем требуемой памяти для хранения буферов, трудоемкость и высокая стоимость устранения лестничного эффекта и трудоемкость реализации эффектов прозрачности и просвечивания.

1.5.4 Обоснование выбора алгоритма удаления невидимых линий и поверхностей

В проекте используется алгоритм, использующий Z-буфер, так как данный алгоритм позволяет строить сцены любой сложности, а также достаточно быстро работает за счет отсутствия сложных вычислений. Так как на сегодняшний день размер оперативной памяти достаточно велик, использование большого объема памяти не является существенным недостатком применения данного алгоритма в программном продукте.

1.6 Алгоритмы закраски

Изображение формируется в результате отражения падающего на поверхность объектов света, интенсивность и цвет которого необходимо рассчитать. Для вычисления этих параметров используют методы закраски поверхностей.

1.6.1 Алгоритм плоской закраски

Алгоритм плоской закраски довольно прост. Сперва вычисляется нормаль к текущему многоугольнику, зачем на основе значения нормали в соответствии с выбранной моделью освещения считается интенсивность цвета всего многоугольника. Такой метод закрашивания обладает высокой скоростью работы, но на визуализированной модели заметны переходы между гранями.
Недостаток данного алгоритма заключается в том, что его применение ограничено тремя условиями:

1) источник света находится в бесконечности;

2) наблюдатель находится в бесконечности;

3) закрашиваемый многоугольник является реально существующим многоугольником, а не результатом аппроксимации поверхностей.

1.6.2 Алгоритм закраски Гуро

Метод Гуро основан на идее закрашивания грани не сплошным цветом, а плавно изменяющимися оттенками. При данном виде закраски интерполируется значение интенсивности, что позволяет получить сглаженное изображение [5].

Процесс закраски Гуро можно разделить на 4 этапа:

1) вычисление векторов нормалей к каждой грани;

2) вычисление векторов нормалей к каждой вершине грани путем усреднения нормалей к граням;

3) вычисление значений интенсивности в вершинах грани;

4) закраска многоугольника путем интерполяции значений интенсивности в вершинах сначала вдоль каждого ребра, а затем и между ребрами вдоль каждой сканирующей строки.

Закраска методом Гуро обеспечивает непрерывность значений интенсивноти вдоль границ многоугольников, но не обеспечивает непрерывности изменения интенсивности, поэтому при использовании данного алгоритма возможно появление полос Маха. Также на некоторых участках поверхность будет выглядеть плоской, поскольку при линейной интерполяции значение интенсивности на этих участках может получится равной.

1.6.3 Алгоритм закраски Фонга

Метод Фонга базируется на той же идее, что и метод Гуро, но при закраске здесь интерполируется не значения интенсивности, а вектор нормали. Затем он используется в моделе освещения для вычисления интенсивности. Это позволяет уменьшить количество полос Маха и добиться лучшей локальной аппроксимации кривизны поверхности. Также метода Фонга, работая с нематовыми поверхностями, выдаёт более реалистичное изображение по сравнению с методом Гуро за счёт наличия зеркальных бликов [5].

Процесс закраски Фонга можно разделить на 5 этапов:

1) вычисление векторов нормалей к каждому полигону;

2) вычисление векторов нормалей к каждой вершине;

3) вычисление значений нормалей вдоль ребер путем интерполяции значений интенсивности в вершинах;

4) вычисление нормали вдоль сканирующей строки;

5) закраска многоугольника путем использования полученной нормали.

Закраска методом Фонга позволяет частично устранить недостатки закраски методом Гуро и получить более реалистичное изображение, однако она сложнее в реализации и требует больших вычислительных затрат.

1.6.4 Обоснование выбора алгоритма закраски

В курсовом проекте используется алгоритм тонировки Гуро, так как он требует меньшего количества вычислений, чем алгоритм закраски Фонга и не сильно уступает в получении реалистичностого изображения по сравнению с плоским алгоритмом закраски.

1.7 Требования к программному обеспечению

Программное обеспечение должно реализовывать возможность генерации, а также моделирования дерева, задающееся следующими индивидуальными характеристиками (подробнее в: 1.3.1):

- точка, из которой будет расти дерево;
- угол между двумя главными ветками дерева;
- угол между веткой и исходящей из нее веткой;
- отношение длины ветки к длине исходящей из нее ветки;
- точка на ветке, из которой будет расти следующая ветка;
- длина ствола дерева;
- радиус ствола;
- радиус веток;
- цвет дерева.

Моделирование дерева должно использовать такие алгоритмы как закраска Гуро и z-буфер.

Пользователь должен иметь возможность загружать, сохранять индивидуальные характеристики дерева, перемещать, вращать и масштабировать как сам 3Д-объект, так и камеру. Также должна быть возможность добавления новых камер (точек обзора). Пользователь должен иметь возможность свободно переключаться между этими камерами, чтобы удобно рассматривать дерево с разных ракурсов.

Помимо этого, пользователь должен иметь возможность менять вектор направления света, а также его интенсивность.

1.8 Выводы по аналитическому разделу

В данном разделе был произведён анализ предметной области, методов и алгоритмов, которые необходимы для решения поставленной задачи.

В качестве модели объектов была выбрана полигональная сетка, хранящаяся в массиве полигонов - треугольников, задающимися координатами вершин.

В качестве алгоритма удаления невидимых граней был выбран алгоритм, использующий z-буфер, который работает достаточно быстро за счет отсутствия сложных вычислений. Алгоритм z-буфера имеет недостаток, заключающийся в использовании большого объема памяти. Однако его можно считать несущественным, поскольку размер оперативной памяти на сегодняшний день считается достаточно большим.

В качестве алгоритма закраски был выбран алгоритм тонировки Гуро, так как он требует меньшего количества вычислений, чем алгоритм закраски Фонга и не сильно уступает в получении реалистичностого изображения по сравнению с плоским алгоритмом закраски.

В качестве алгоритма генерации реалистичного каркаса дерева был выбран алгоритм, основывающийся на модели Хонды.

Каждая ветка дерева будет представлять из себя конус, разбитый на определенное количество полигонов.

2 Конструкторский раздел

В данном разделе будут более подробно описаны выбранные в аналитическом разделе алгоритмы, будут перечислены основные типы и структуры данных, использующиеся в проекте, и представлена диаграмма классов архитектуры проекта.

2.1 Процесс отрисовки изображения

2.1.1 Z-буфер

В процессе работы алгоритма z-буфера анализируется глубина каждого пикселя в пространстве изображения. Если пиксели двух рисуемых объектов перекрываются, то их значения глубины сравниваются. Если это сравнение показывает, что новый пиксел расположен ближе к наблюдателю, чем пиксел, уже находящийся в буфере кадра, то новое значение координаты z заносится в z-буфер, корректируя значение интенсивности, находящееся в буфере кадра. Если же сравнение дает противоположный результат, то никаких действий не производится. По сути, алгоритм является поиском по x и y наибольшего значения функции z(x,y).
Формальное описание алгоритма z-буфера:

1) заполнить буфер кадра фоновым значением интенсивности или цвета;

2) заполнить z-буфер минимальным значением z;

3) преобразовать каждый многоугольник в растровую форму в произвольном порядке;

4) для каждого пикселя в многоугольнике вычислить его глубину z(x,y);

5) сравнить глубину z(x,y) со значением z-буфера, хранящимся в z-буфере в этой же позиции;

6) 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). В противном случае никаких действий не производить.

На рис. 6 представлен полный процесс отрисовки изображения.

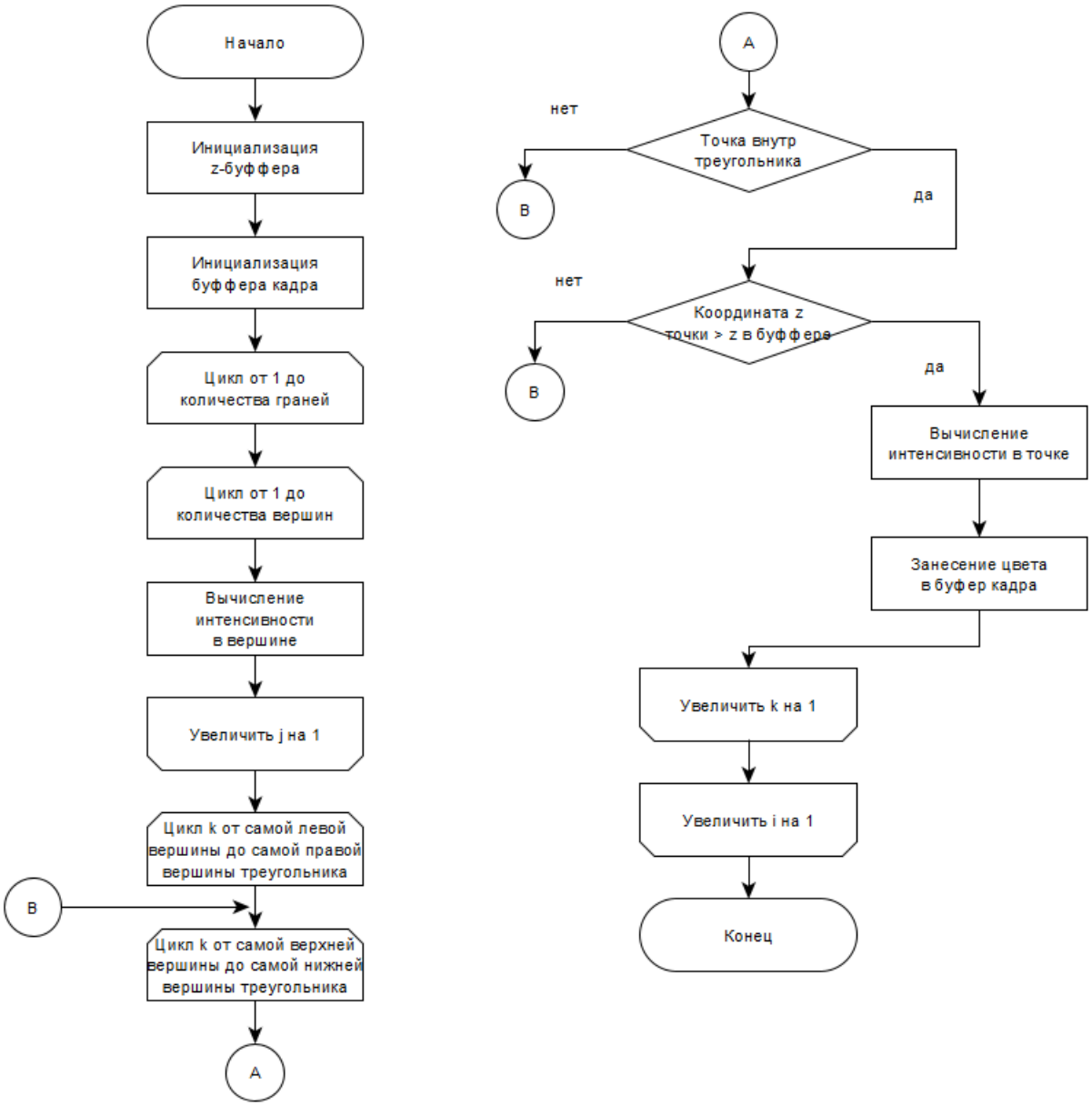


Рис. 6: Процесс отрисовки изображения

2.1.2 Закраска Гуро

Метод закраски, который основан на интерполяции интенсивности и известен как метод Гуро (по имени его разработчика), позволяет устранить дискретность изменения интенсивности. Процесс закраски по методу Гуро осуществляется в четыре этапа:

1. Вычисляются нормали ко всем полигонам;
2. Определяются нормали в вершинах путем усреднения нормалей по всем полигональным граням, которым принадлежит вершина;
3. Используя нормали в вершинах и применяя произвольный метод закраски, вычисляются значения интенсивности в вершинах;

4. Каждый многоугольник закрашивается путем линейной интерполяции значений интенсивностей в вершинах сначала вдоль каждого ребра, а затем и между ребрами вдоль каждой сканирующей строки.

Линейная интерполяция осуществляется следующим образом: Интенсивность вдоль отрезка AB в некоторой точке C вычисляется по формуле [8]:

Схема алгоритма закраски Гуро приведена на рисунке 7.

Рис. 7: Схема алгоритма закраски методом Гуро

2.2 Генерация каркаса дерева

На рисунке 8 представлена схема алгоритма генерации дерева, использующего модель Хонды.

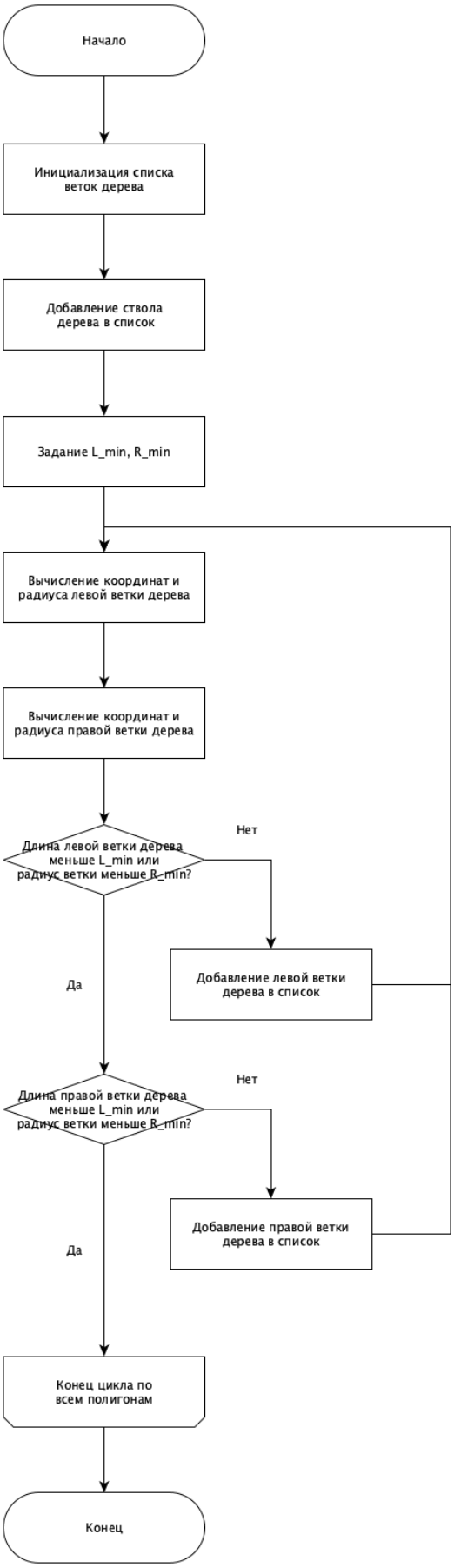


Рис. 8: Схема алгоритма генерации дерева, использующего модель Хонды

2.3 Построение 3Д-модели на основе получившегося каркаса дерева

Для представления получившегося каркаса дерева в виде совокупности конусов, разбитых на определенное количество полигонов, необходимо:

- переместить i-ую ветку в начало координат;
- повернуть ее так, чтобы она совпала с осью OZ [3];
- Разбить конус на определенное количество треугольников.

2.4 Трехмерные аффинные преобразования

В процессе работы программы возникает необходимость преобразования объектов сцены – их поворота. Для трехмерного пространства любое аффинное преобразование может быть представлено последовательностью простейших операций.

Ниже приведены уравнения поворота относительно осей x (1), y (2) и z (3) на угол ϕ.

$$\begin{cases} X = x \\ Y = y \cdot \cos\phi + z \cdot \sin\phi \\ Z = -y \cdot \sin\phi + z \cdot \cos\phi \end{cases} \tag{1}$$

$$\begin{cases} X = x \cdot \cos\phi + z \cdot \sin\phi \\ Y = y \\ Z = -x \cdot \sin\phi + z \cdot \cos\phi \end{cases} \tag{2}$$

$$\begin{cases} X = x \cdot \cos\phi + y \cdot \sin\phi \\ Y = -x \cdot \sin\phi + y \cdot \cos\phi \\ Z = z \end{cases} \tag{3}$$

Поворот объекта относительно одной из координатных осей также можно представить в виде матриц. Ниже представлены три матрицы поворота объекта относительно оси x, y и z соответственно.

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\phi & \sin\phi & 0 \\ 0 & -\sin\phi & \cos\phi & 0 \\ 0 & 0 & 0 & 1 \\ & & & \end{pmatrix}$$

$$\begin{pmatrix} \cos\phi & 0 & \sin\phi & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\phi & 0 & \cos\phi & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$\begin{pmatrix} \cos\phi & \sin\phi & 0 & 0 \\ -\sin\phi & \cos\phi & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

2.5 Используемые типы и структуры данных

В реализованной программе используются некоторые базовые типы C++, такие как integer и double. Также применяется тип данных QVector для хранения объектов сцены, буфера кадра и реализации алгоритма, использующего z-буфера.

Кроме того, в программе используются структуры данных, описывающие матрицу и математический вектор, который предусматривает операции векторного и скалярного умножения, нахождения длины вектора, нормализации; а также бинарное дерево - для реализации алгоритма генерации деревьев.

2.6 Общая архитектура проекта

Архитектура проекта сформирована в соответствии с канонами объектноориентированного программирования. На рис. 9 представлена диаграмма классов архитектуры проекта.

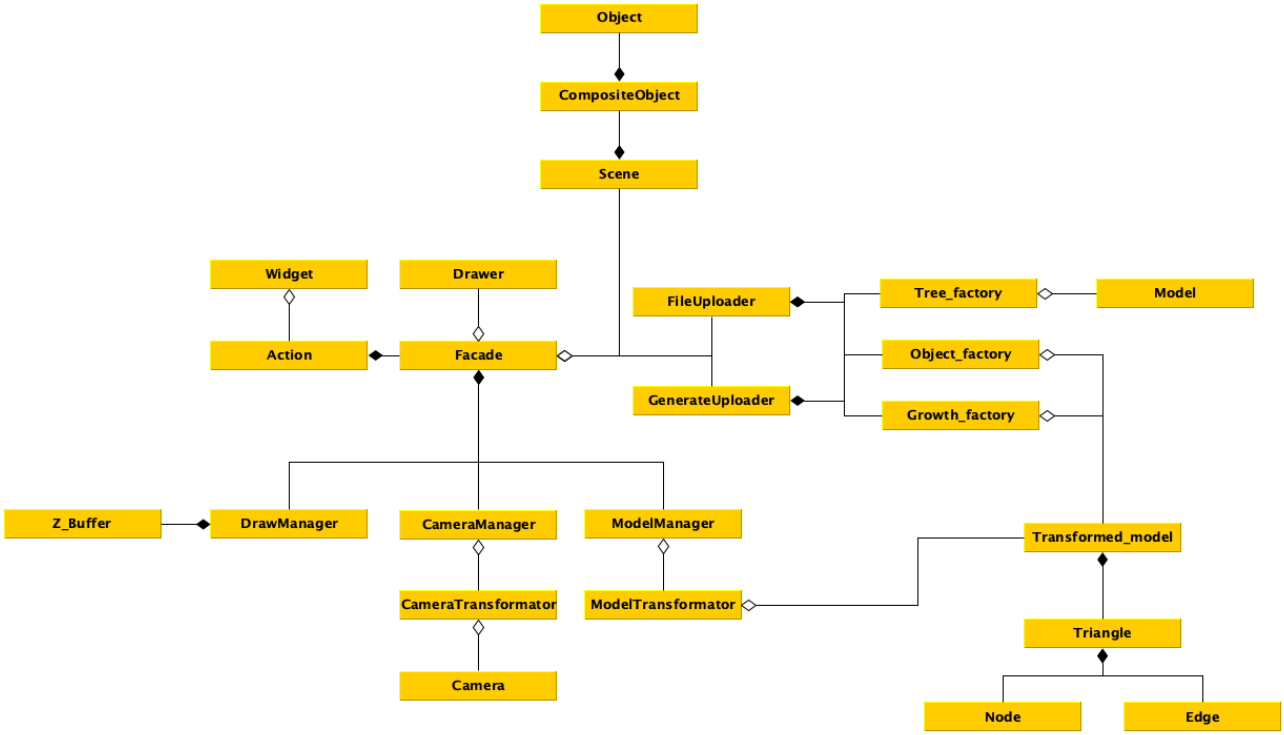


Рис. 9: Диаграмма классов программы

- Widget – основной класс, работающий с окном приложения;
- Growth_factory - класс генерации жизненного цикла дерева, реализующий шаблон проектирования «Фабрика» [9].
- Object_factory - класс построения 3Д-модели на основе получившегося каркаса дерева, реализующий шаблон проектирования «Фабрика» [9].
- Tree_factory - класс генерации дерева, реализующий шаблон проектирования «Фабрика» [9].
- Action - класс, содержащих информацию о команде, которая передается через Facade.
- Facade - класс, реализующий шаблон проектирования «Фасад» [9].
- Drawer - класс отрисовщика, содержащий методы, которые реализуют отображение буфера кадра на экране.
- DrawManager - класс менеджера отрисовщика; содержит методы отрисовки модели.

- ModelManager - класс, применяющий преобразования перемещения, масштабирования и поворота к моделям через ModelTransformator.
- CameraManager - класс, применяющий преобразования перемещения, масштабирования и поворота к камерам через CameraTransformator.
- Camera - класс камеры, позволяющий изменять параметры перемещения, масштабирования и поворота камеры.
- CompositeObject - класс, реализующий шаблон проектирования «Компоновщик» [9].
- Edge - класс ветки дерева, содержит номера вершин веток и радиус ветки.
- Model - класс каркаса дерева, содержит Edge и Node.
- Node - класс точки, содержит вершину точки и цвет.
- Transformed_model - класс, содержащий полигональную модель дерева.
- Triangle - класс треугольника, содержит вершины треугольника и цвет вершин.
- Scene - класс, позволяющий добавлять, удалять модели, камеры сцены, обеспечивает взаимодействие с CompositeObject.
- FileUploader - класс загрузщика модели из файла; содержит методы, реализующие работу с файлом; выполняет загрузку считанных параметров в класс Tree_factory.
- GenerateUploader - класс созданий модели, выполняет загрузку параметров в класс Tree_factory.
- ModelTransformator - класс применяющий преобразования перемещения, масштабирования и поворота к модели.
- CameraTransformator - класс применяющий преобразования перемещения, масштабирования и поворота к камере.
- Z_Buffer - содержит методы, реализующие удаление невидимых линий и поверхностей алгоритмом z-буфера.

2.7 Выводы по конструкторскому разделу

В данном разделе были даны формальные описания алгоритмов z-буфера, Гуро, Хонды, полигонализации конусов; выбранных в аналитическом разделе.

Также были приведены математические формулы поворота объекта относительно каждой из трех координатных осей. Были перечислены основные типы и структуры данных, использующиеся в проекте, и представлена диаграмма классов архитектуры проекта.

3 Технологический раздел

В данном разделе будут рассмотрены средства, использованные в процессе разработки для реализации поставленных задач, информация о сборке и запуске проекта, будут представлены основные функции интерфейса, а также приведены примеры работы программы при различных параметрах.

3.1 Средства реализации

Для реализации программы был использован язык программирования C++, стандарт C++17 [6]. C++ имеет достаточный функционал в стандартных библиотеках и является высокопроизводительным языком, что очень важно при реализации задач компьютерной графики из-за высокой нагрузки на аппаратное обеспечение.

Проект был выполнен в среде QtCreator[7]. В качестве графической оболочки была выбрана QtGraphics, которая предоставляет поверхность для управления и взаимодействия с большим количеством пользовательских графических 2D объектов, что важно при отрисовки большого количества многоугольников полигональной сетки. Платформа включает в себя архитектуру распространения событий, которая позволяет использовать возможности взаимодействия элементов на сцене.

3.2 Сборка и запуск проекта

Для компиляции и сборки проекта используется gcc и системы сборки qmake[7], которая автоматически генерирует make-файлы, основываясь на информации в файлах проекта. Также для корректной сборки приложения необходимы основные библиотеки QtGraphics, стандартные библиотеки C++ и доступ к пространству имен std.

3.3 Интерфейс программы

Интерфейс программы состоит из окна вывода изображения, имеющего фиксированный размер 700x700 пикселей, и набора виджетов, позволяющих пользователю взаимодействовать с программой (рисунок 10).

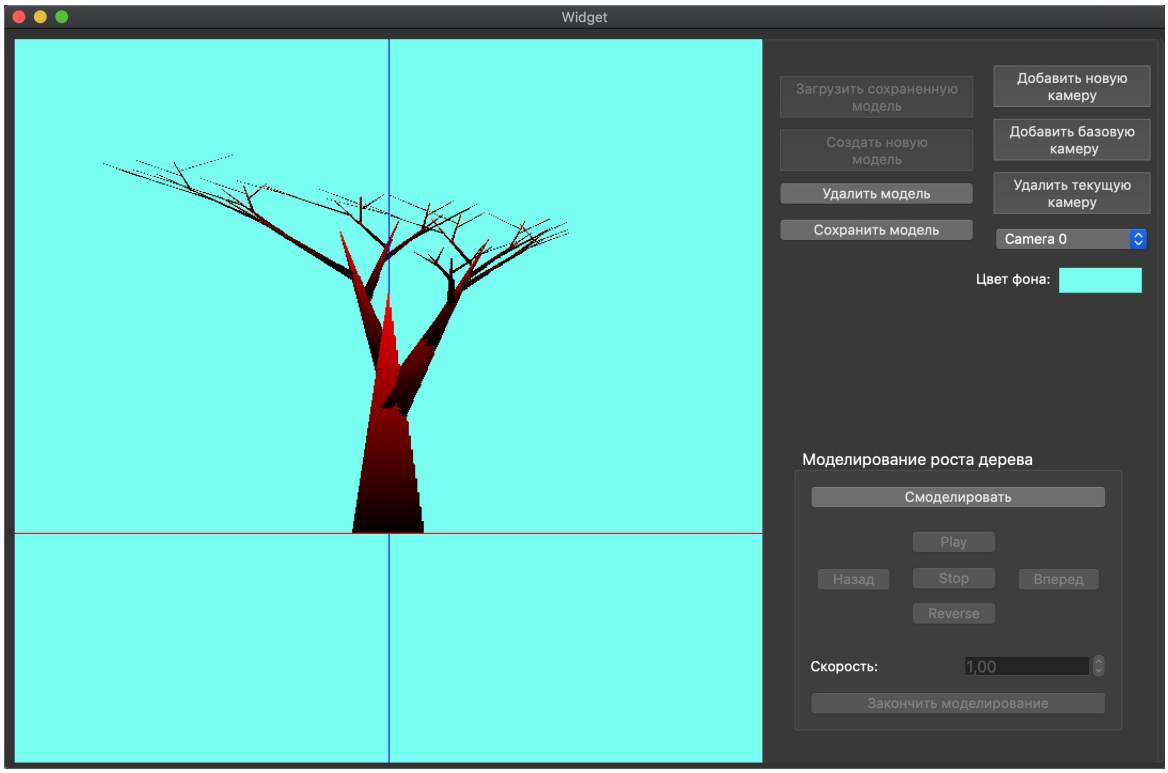


Рис. 10: Интерфейс программы

При запуске программы пользователю предоставлется возможность выбрать, хочет ли он загрузить готовые параметры дерева, или же он хочет задать их вручную. Если он выбирает загрузку параметров, то перед ним выскакивает каталог файлов, из которых ему необходимо будет выбрать модель. В другом случае перед ним появится окно с параметрами дерева, которое будет сгенерировано, их необходимо заполнить (рисунок 11).

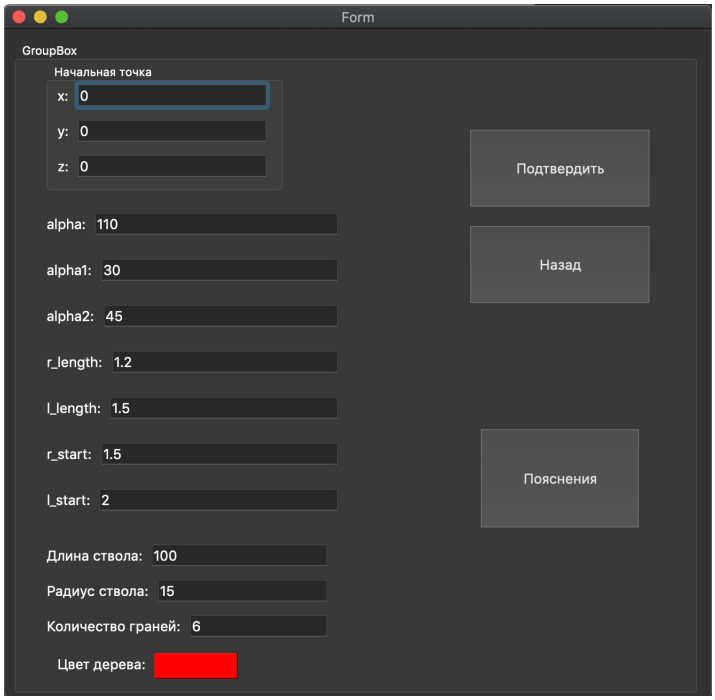


Рис. 11: Параметры дерева

После нажатия кнопки "Подтвердить"на экран выводится сгенерированное дерево, которое можно поворачивать, масштабировать и перемещать, используя клавиши на клавиатуре (подробнее в горячих клавишах программы: 3.4).

Помимо вращения самой модели, обеспечивается возможность добавление и удаление новых камер, а также переключение между ними с помощью кнопок:

1. «Добавить камеру» - добавляет камеру, зафиксированную в положении, что и предыдущая камера;
2. «Удалить камеру» - удаляет текущую камеру;
3. «Добавить базовую камеру» - добавляет камеру, направленную в начало координат;
4. «Camera$i$» - позволяет переключаться между камерами.

Камеры как и сам объект можно вращать, приближать и перемещать используя те же горячие клавиши после однократного нажатия

клавищи Command (подробнее о горячих клавишах программы: 3.4).

Программа предоставляет возможность пользователю изменять вектор направления падения лучей источника света и интесивность источника света, используя те же горячие клавиши после двухкратного нажатия клавищи Command (подробнее о горячих клавишах программы: 3.4).

В нижней правой части экрана присутствует необходимое меню для моделирования роста дерева. При нажатии на кнопку "Смоделировать"генерируется дерево "в более ранних стадиях его развития после чего "зерно"дерева выводится на экран.

После нажатия на кнопку "Play"происходит визуализация жизненного цикла дерева путем последовательной смены 3Д-объектов одного за другим. При этом пользователь все также имеет возможность перемещать, поворачивать, масштабировать само дерево или камеру в процессе визуализации. Пользователь может в любой момент нажать на кнопку "Stop после чего моделирование жизненного цикла дерева будет приостановлено.

После нажатия на кнопку "Reverse"происходит обратная визуализация жизненного цикла дерева путем последовательной смены 3Д-объектов одного за другим. Пользователь может замедлить или увеличить скорость роста дерева используя счетчик, расположенный рядом с кнопкой "Reverse".

В качестве альтернативы кнопкам "Play"и "Reverse"пользователь может воспользоваться кнопками "Вперед"или "Назад чтобы более детально рассмотреть каждый этап роста дерева.

3.4 Горячие клавиши программы

Ниже приведен список кнопок на клавиатуре, выполняющих ту или иную функцию:

1. A - перемешение вдоль оси OX в отрицательном направлении;
2. D - перемешение вдоль оси OX в положительном направлении;
3. S - перемешение вдоль оси OY в положительном направлении;
4. W - перемешение вдоль оси OY в отрицательном направлении;
5. Q - перемешение вдоль оси OZ в положительном направлени;
6. E - перемешение вдоль оси OZ в отрицательном направлени;
7. R - масштабирование (увеличение);
8. T - масштабирование (уменьшение);
9. Z - поворот вокруг оси OX в положительном направлении;
10. X - поворот вокруг оси OY в положительном направлении;
11. C - поворот вокруг оси OZ в положительном направлении;
12. V - поворот вокруг оси OX в отрицательном направлении;
13. B - поворот вокруг оси OY в отрицательном направлении;
14. N - поворот вокруг оси OZ в отрицательном направлении;
15. Command - при первом нажатии этой кнопки все предыдущие кнопки начинают работать с камерой, а не с объектом (до этого перемещался, масштабировался и вращался объект с помощью предыдущих кнопок); при втором нажатии этой кнопки кнопки с Z-N начинают работать с освещением, при третьем нажатии - с объектом и так далее;
16. O - увеличить яркость освещения;
17. L - уменьшить яркость освещения;

3.5 Тестирование программы

На рисунках 12, 13, 14, 15 и 16 представлены примеры работы программы для различных параметров генерации дерева. Примеры демонстрируют результаты выполнения программы при различных положениях объектов, камер и источника света.

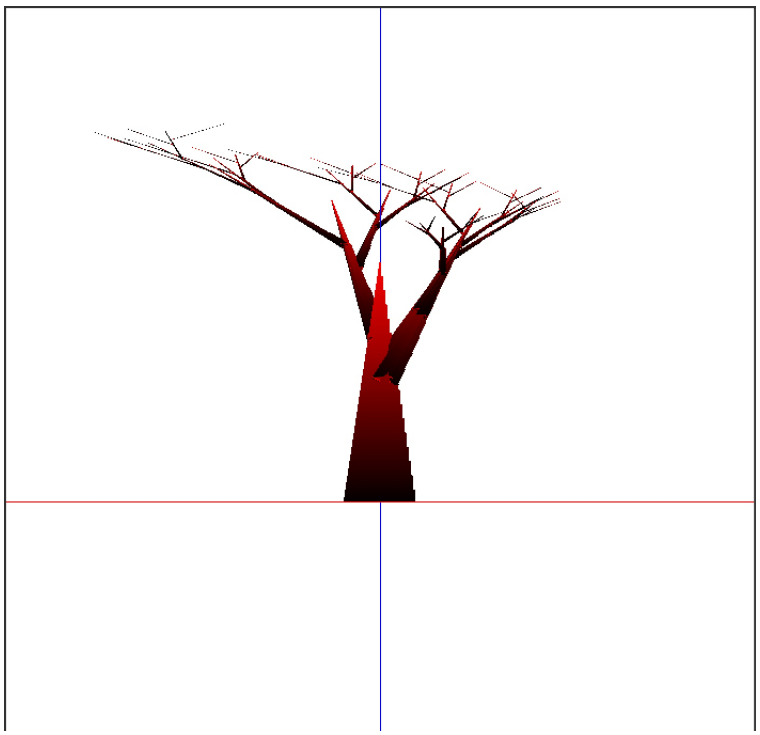


Рис. 12: Пример работы программы

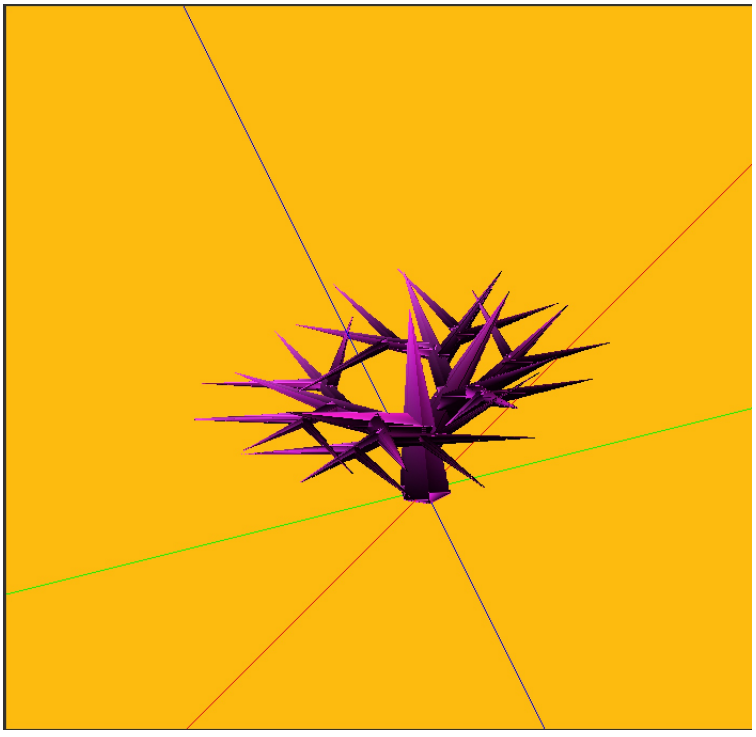


Рис. 13: Пример работы программы

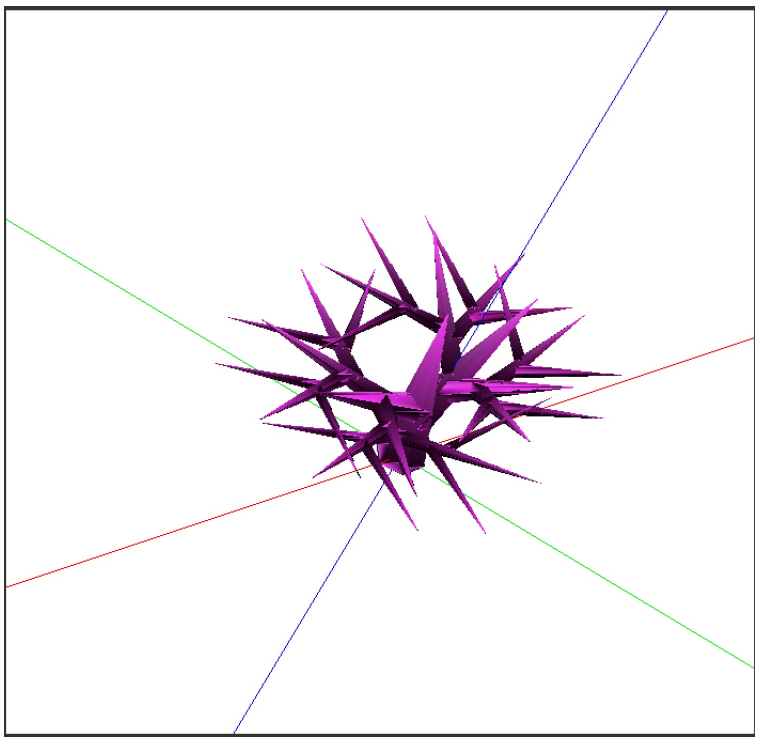


Рис. 14: Пример работы программы

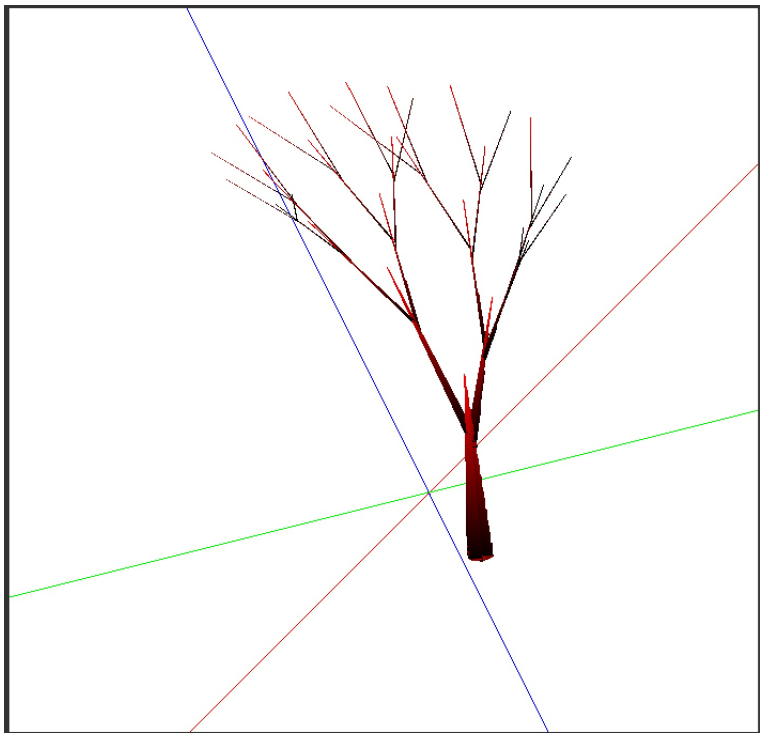


Рис. 15: Пример работы программы

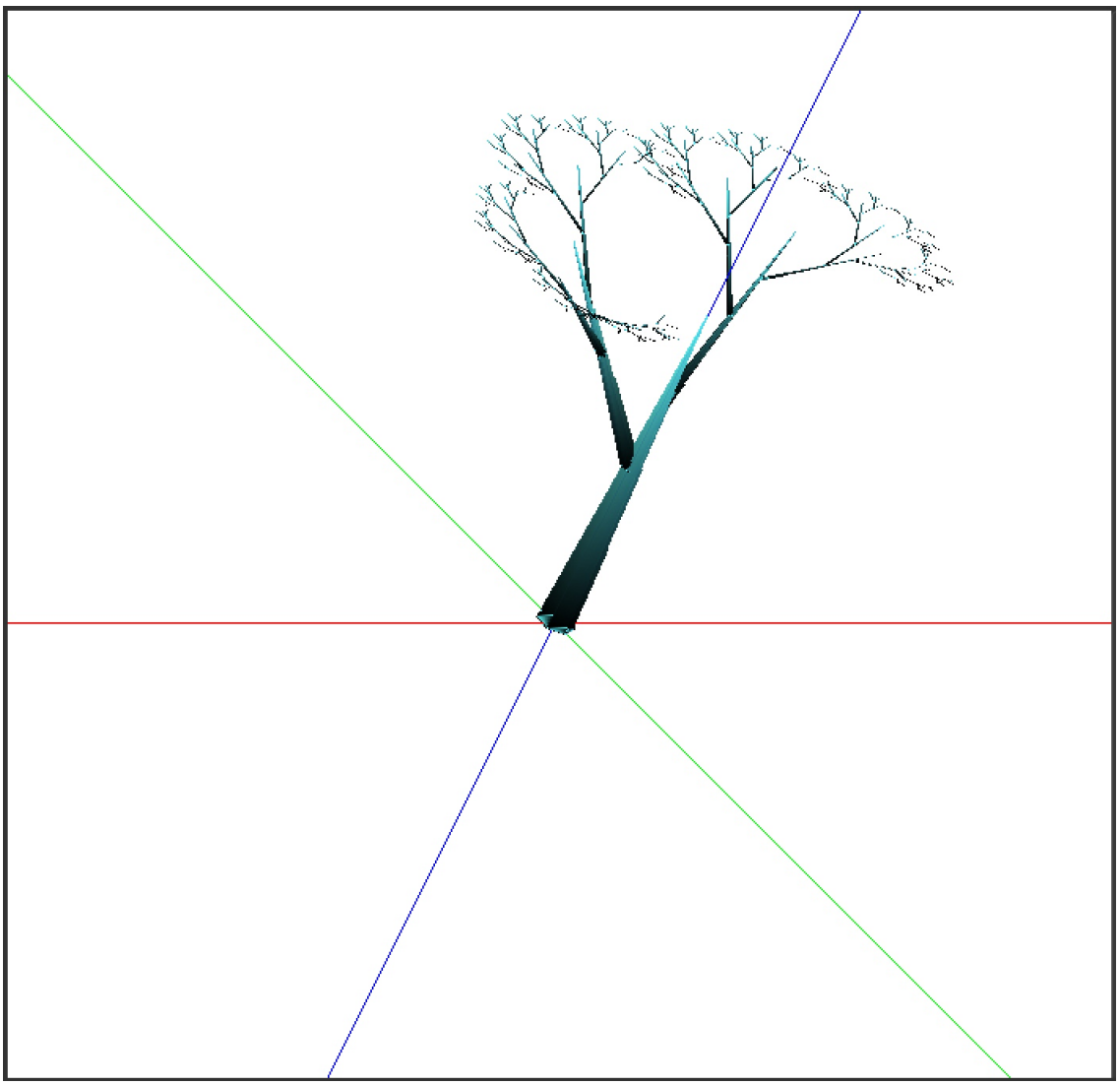


Рис. 16: Пример работы программы

3.6 Выводы по технологическому разделу

В данном разделе были рассмотрены средства, использованные в процессе разработки для реализации поставленных задач, информация о сборке и запуске проекта, были представлены основные функции интерфейса, а также приведены примеры работы программы при различных параметрах.

4 Исследовательский раздел

В данном разделе будет проведено исследование временных затрат разработанного программного обеспечения в зависимости от количества потоков.

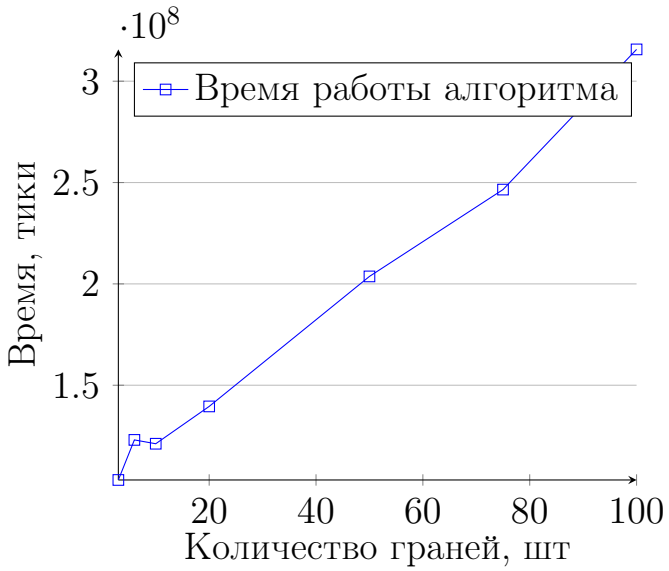
4.1 Исследование скорости работы алгоритма

Для исследования скоростных характеристик был использован компьютер на базе процессора Intel Core i7, с оперативной паматью 8 ГБ. Модуль тестирования запускался под операционной системой macOS Mojave.

В таблице представлены данные, полученные в ходе замера времени работы алгоритма отображения одного и того же объекта при различном количестве граней конусов ствола и веток.

Количество граней	Время, тики
3	103259868
6	120575960
10	121135023
20	139548342
50	203685764
75	246526176
100	315689052

Таблица 1: Время работы алгоритма при разном количестве граней конусов ствола и веток



Очевидно, программа работает быстрее с меньшим количеством граней. Из данных графика можно сделать вывод, уже при 75 гранях программа работает почти 2.04 раз быстрее, чем при 6 гранях.

4.2 Выводы по исследовательскому разделу

В данном разделе было проведено исследование временных затрат разработанного программного обеспечения в зависимости от количества

полигонов, на которые разбивается объект. Оно показало, что уже при разбиении конусов на 75 граней, алгоритм работает в 2 раза медленнее чем при 6 гранях.

Заключение

В результате выполнения данного курсового проекта были изучены и реализованы генерация и моделирование деревьев. Была определена и описана предметная область работы, рассмотрены существующие методы и алгоритмы, из которых были выбраны подходящие для решения поставленной задачи, была сформирована архитектура проекта, спроектирован пользовательский интерфейс и реализовано требуемое ПО.

В аналитическом разделе был произведён анализ предметной области, методов и алгоритмов, которые необходимы для решения поставленной задачи. В конструкторском разделе были подробно описаны эти алгоритмы, были даны формулы поворота относительно каждой из трех осей, перечислены основные типы и структуры данных, использующиеся в проекте, а также представлена диаграмма классов архитектуры проекта. В технологическом разделе были рассмотрены средства, использованные в процессе разработки для реализации поставленных задач, информация о сборке и запуске проекта, были представлены основные функции интерфейса, а также приведены примеры работы программы при различных параметрах.

Список литературы

[1] The Algorithmic Beauty of Plants. - Режим доступа: http://algorithmicbotany.org/papers/#abop, свободный - (Дата обращения: 01.10.2019)

[2] L-systems. Моделирование деревьев. [Электронный ресурс] - Режим доступа: https://habr.com/ru/post/83373/, свободный - (Дата обращения: 11.12.2019)

[3] Поворот вокруг произвольной оси в пространстве - Режим доступа: http://scask.ru/a_book_mm3d.php?id=60, свободный - (Дата обращения: 11.10.2019)

[4] Роджерс Д., «Алгоритмические основы машинной графики»/Роджерс Д. - Москва: Издательство «Мир». Редакция литературы по математическим наукам, 1989

[5] Куров А.В. Курс лекций по машинной графике. – М., 201

[6] Докуменация C++ [Электронный ресурс]. - Режим доступа: https://cppreference.com/, свободный. (Дата обращения: 20.09.2019 г.)

[7] Документация QtCreator [Электронный ресурс]. - Режим доступа - https://doc.qt.io/, свободный. (Дата обращения: 20.09.2019 г.)

[8] Основы освещения. [Электронный ресурс]. - Режим доступа: https://devburn.ru/2015/09/01/основы-освещения/, свободный - (Дата обращения: 14.09.2019)

[9] Паттерны проектирования [Электронный ресурс]. – Режим доступа: https://refactoring.guru/ru/design-patterns, свободный – (18.11.2019)