ВВЕДЕНИЕ.

Краткая постановка задачи сформулирована в техническом задании проекта. Цель введения — раскрыть и дополнить техническое задание (ТЗ), а также выделить абстракции, понятия и сокращения, которые будут действительны на протяжении всего дальнейшего изложения.

Приведу формулировку технического задания:

«Разработать программу моделирования и анимации открытого огня. Программа должна реалистично визуализировать процесс горения на трехмерной сцене, при наличии статического окружения. Необходимо также реализовать анимацию дыма, получаемого в процессе горения и дать возможность пользователю настраивать параметры анимации».

Техническое задание описывает требования к программному продукту. Можно выделить шесть компонентов, из которых этот продукт должен состоять:

1) Трехмерная сцена.

2) Несложное статическое окружение.

3) Получение реалистичного огня (пламя, всполохи, спектры цветов …).

4) Рисование дыма, получаемого при горении.

5) Выделение ключевых параметров анимации.

6) Пользовательский интерфейс.

Связи между компонентами неочевидны и будут рассматриваться при построении архитектуры, но уже сейчас может быть выделена главная задача проекта — рендеринг реалистичной анимации. Все остальные задачи являются второстепенными и будут реализованы как расширения главной задачи, реализацию которой в программном коде, в дальнейшем будем называть ядром.

Можно видеть, что система разбита на два уровня: уровень ядра и уровень подзадач, в свою очередь разбитый на подзадачи. Большой плюс подобного разбора в том, что его можно безболезненно перенести на построение архитектуры программного решения. В таком случае, выделенные подзадачи и ядро становятся доменами.

Замечу, что здесь представлена вся рассматриваемая система, хоть и в самом абстрактном её понимании. Никаких других задач или уровней в дальнейшем вводиться не будет. То есть, вся последующая работа проводится вглубь обозначенных пунктов.

Расчетно-пояснительная записка поделена на разделы, каждый из которых описывает определенные аспекты проведенной работы:

- Анализ предметной области, выделение базовых программных сущностей, выбор инструментов разработки — находятся в аналитическом разделе.

- Математическая модель ядра описана и проанализирована в конструкторском разделе.

- Программная реализация, тестирование и исследование программного решения составляют часть конструкторского раздела и полностью весь технический раздел.

Терминология, принятая в данном документе не расходится с общепринятой в области компьютерной графики. Однако стоит привести разъяснения по поводу слов «рендеринг» и «визуализация». В дальнейшем они будут употребляться как синонимы (второе чаще), в значении: процесс получения изображения по модели с помощью компьютерной программы.

2. АНАЛИТИЧЕСКИЙ РАЗДЕЛ.

**2.1. Анализ предметной области.**

В реальной жизни задача графической визуализации огня встречается достаточно часто. Однако проводить для этого реальные эксперименты дорого и опасно. Чтобы записать на видео красивый взрыв, необходимо выбрать удаленное от людей (а значит и от инфраструктуры) место, заготовить горючее и средства обеспечения безопасности, получить разрешение от властей. В итоге, полученные кадры будут эксклюзивными, ведь, чтобы переснять их в других ракурсах, потребуется повторение всего эксперимента. Тут и приходят на помощь программы-симуляторы.

Заказчиком технологии выступает в основном индустрия развлечений, которой огонь нужен в компьютерных играх и в кино, причем эти отрасли накладывают различные ограничения. Компьютерные игры требуют симуляции в реальном времени и с вычислительными ресурсами персонального компьютера. В то же время для киноиндустрии наиболее важна реалистичность. Ради получения реалистичных симуляций могут задействоваться большие вычислительные мощности и допустимо обрабатывать каждый кадр по несколько минут.

Министерство чрезвычайных ситуаций также использует программные симуляции природных явлений. Например, для таких задач как системы предсказания лесных пожаров и компьютерные системы обучения сотрудников.

В данной работе, решено сконцентрироваться на достижении довольно жестких требований, предъявляемых к компьютерным играм. Благодаря этому целевая аудитория созданного программного продукта будет наибольшей.

**2.2. Классификация существующих методов.**

Для рисования огня в компьютерной графике было изобретено достаточно много методов. Американский инженер H.R. Baum составил их классификацию в своей работе «Computer fire simulation»:

1) Particle system based.

Огонь представляется в виде большого количества частиц, которые движутся по случайным траекториям, управляемые стохастическими процессами. Каждая частица имеет свой единственный цвет и некоторую степень прозрачности. При большом количестве частиц, метод сильно нагружает компьютер, при этом не давая ни внешней ни поведенческой реалистичности. Преимущество данного метода - простота реализации.

2) Noise-synthesized.

Здесь огонь получается путем наложения друг на друга нескольких сеток шумов. Выглядит хорошо, однако трудно воспроизвести реальное поведение.

3) Geometry-skeleton.

Используется лучевой каркас, который затем покрывается некоторой поверхностью. Данный метод подкупает своей простотой реализации и простотой расчетов. Других преимуществ не имеет.

4) Data-driven.

Допустим имеется возможность наблюдать реальный огонь в реальном мире и фиксировать его с помощью фото или видеосъемки. (Что случается достаточно редко). Собранный материал можно использовать для построения компьютерной модели. Метод нуждается в большом хранилище данных и заведомо реалистичен. Однако, чтобы добавить новое поведение огня, нужно снова проводить реальные эксперименты и увеличивать объем базы данных.

5) Physically-based.

К последнему классу относятся наиболее трудные для моделирования и программирования методы. Они же дают самый качественный результат, то есть самое реалистичное изображение. Поэтому, исследование «физически-обоснованных» способов представляет огромный интерес для всех, кто нуждается в искусственной симуляции огня на персональном компьютере. В данном направлении работают такие исследователи как Jos Stam, Ronald Fedkiw, Duc Quang Nguyen.

Недостатки метода: сложная программная реализация, возможная расходимость применяемого метода расчета состояний модели и долгий расчет этой самой модели.

**2.3. Обзор и анализ существующих программных систем, и обоснование необходимости разработки.**

Для рендеринга огня могут быть использованы пакеты редактирования компьютерной графики: Autodesk Maya или Blender3D. Качество получаемых результатов для обоих приблизительно одинаково.

Blender3D – свободный пакет для создания трехмерной компьютерной графики, включающий в себя средства моделирования, анимации, рендеринга, постобработки видео, а также создания интерактивных игр. Имеет репутацию программы сложной для изучения.

Autodesk Maya – открытый для сторонних разработчиков редактор трехмерной графики. В настоящее время стал стандартом 3D графики в кино и телевидении.

Оба программных пакета включают в себя невероятно много функциональных возможностей, требуют знания специальных языков для настройки и работы (python для Blender, MEL основанный на Tcl, для Maya). Соответственно они требуют от пользователя долгого обучения, из-за чего использовать их могут лишь профессионалы либо специализированные студии. В то же время на рынке нет простого инструмента, который могли бы использовать небольшие команды, разрабатывающие компьютерные игры или симуляторы.

Созданный программный продукт призван заполнить собой данную нишу.

**2.2. Выбор, обоснование метода моделирования и алгоритма.**

Для реализации ядра системы, автором был выбран Physically-based method. Причиной выбора послужило то, что, хотя это и не самый сложный метод, но он дает самый реалистичный и качественный результат. Также данный метод позволяет решать и некоторые дополнительные задачи, к примеру, распространение огня, тление и затухание. Это значит, что в будущем разработанная система может быть расширена до полноценного Фреймворка.

Существует занимательный факт, который несколько упрощает реализацию ядра. Дело в том, что движение жидкостей, газов и плазмы в поле скоростей можно описать одной и той же системой векторных уравнений. Система называется уравнениями Навье-Стокса, в честь французского физика Навье и британского математика Стокса. Конечно, так можно делать только тогда, когда не требуется физическая достоверность моделирования. Действительно, проектируемая модель используется для рендеринга, а не для проведения замеров физических параметров и такой уровень точности просто не нужен.

Огонь, в первом приближении, является плазмой, а дым — газом. Значит оба явления можно описать с помощью одной и той же математической модели. Разница будет лишь в коэффициентах расчета, таких как вязкость, диффузия, цветовые спектры и некоторые другие. Замечу, что при реализации анимации огня другими методами, дым пришлось бы моделировать с нуля.

Для реализации ядра в этой работе, был взят за основу программный движок, описанный Jos Stam в его статье Real-time fluid dynamics for games.

3. КОНСТРУКТОРСКИЙ РАЗДЕЛ.

**3.1. Математические основы метода моделирования.**

Уравнения Навье-Стокса – система дифференциальных уравнений в частных производных для абстрактных векторных полей любой размерности, описывающая движение вязкой ньютоновской жидкости. Уравнения Навье-Стокса являются одними из важнейших в гидродинамике и применяются в математическом моделировании многих природных явлений и технических задач.

Пусть в момент времени t1 имеется n-мерная сетка, в каждой точке которой определены скорость U и плотность вещества R. Уравнения Навье-Стокса учитывают параметры среды и позволяют определить значения величин U, R для момента t2.

Здесь и далее, полем называется векторное поле, на котором применимы операторы векторной алгебры (градиент, дивергенция, лапласиан) и определены параметры среды. Сетка – это математическая абстракция, описывающая поле с некоторой точностью. Первое понятие применяется при моделировании, а второе при реализации модели.

Скорость перемещения вещества в огне и дыме намного меньше, чем скорость звука. Следовательно, можно допустить, что скорость звука в рассматриваемых средах бесконечна и любые возмущения передаются немедленно. Данное условие позволяет использовать модель несжимаемой жидкости, что сильно упростит систему уравнений. Ведь если жидкость несжимаемая, то плотность сохраняется при изменении давления и само давление можно не принимать в расчет.

В результате, система уравнений модели выглядит следующим образом.

(1)

Это нелинейная система векторных уравнений в частных производных. Верхнее описывает поведение поля скоростей и называется уравнением движения. Оно в точности описывает эволюцию поля скоростей в течение времени. Нижнее — движение вещества в поле скоростей. Система справедлива как для двумерного, так и для трехмерного случая (меняется только количество компонент векторов).

Само по себе поле скоростей не имеет визуального смысла, пока нет объектов, на которые оно воздействует. Движение этих объектов рассчитывается путем перевода окружающих их скоростей в силы. Простые объекты, вроде частиц, обычно просто двигаются вдоль скоростей, но в случае сплошных сред моделировать каждую частицу слишком ресурсозатратно. Поэтому частицы вещества заменим плотностью: непрерывной функцией, которая для каждой точки пространства определяет количество вещества в ней.

Рассмотрим сначала второе уравнение. Оно утверждает, что изменение скалярного поля плотностей во времени происходит из-за трех причин, соответствующих трем слагаемым в правой части. Первая — это так называемая адвекция (advection), в сущности, перемещение вещества под действием поля скоростей. Вторая — диффузия вещества, где каппа — коэффициент диффузии. И, наконец, третья причина — поле источников плотности, обозначаемое S.

Верхнее уравнение очень похоже на нижнее. И действительно, три слагаемых в правой части имеют ту же самую смысловую нагрузку: адвекция, диффузия, где ню — коэффициент, и источники поля (применительно для скоростей это поле сторонних сил; благодаря этому полю можно, например, моделировать ветер, сдувающий языки пламени). При этом уравнение для плотностей является линейным, в то время как второе нелинейно. Отсюда следует один важный вывод. Сначала следует разработать алгоритм для решения нижнего уравнения в фиксированном поле скоростей, и затем полученное решение применить для вычисления нового поля скоростей.

Ядро программного решения призвано решить данную систему для заданных коэффициентов и полей.

К сожалению, в настоящее время не существует общего аналитического решения уравнений Навье-Стокса. Существование и гладкость решений этих уравнений – одна из семи математических задач тысячелетия, сформулированных в 2000 году Математическим институтом Клэя. По состоянию на 2014 год подтвержденные решения найдены лишь в некоторых частных случаях, обусловленных простой геометрией. В остальных случаях применяются численные методы.

**3.2. Разработка алгоритма моделирования.**

Математические уравнения и абстракции очень полезны при проектировании модели. Однако при реализации, необходимо перейти к конечным представлениям полей. Общепринятым подходом является использование конечных сеток, которые состоят из одинаковых ячеек. Внутри каждой ячейки характеристики полей являются постоянными величинами. Такое представление позволяет применить численные методы для решения уравнений Навье-Стокса, а также производить рендеринг на растровых дисплеях.

Замечу, что благодаря введению сеток мы получаем первый инструмент для оценки реалистичности получаемого изображения: количество пикселей на одну ячейку. Подробнее об этом будет сказано в экспериментальном разделе.

**3.3. Разработка алгоритма анимации.**

Алгоритм симуляции огня представляет собой безусловный цикл по времени. Внутри цикла выполняется перерасчет полей плотности и скорости, а также отрисовка огня на сцене. Помимо этого, производится калибровка поля скоростей, для того чтобы эмулировать действие температуры. Это необходимо делать, потому что температура не входит явно в систему уравнений (1).

Алгоритм изображен на рисунке 1.

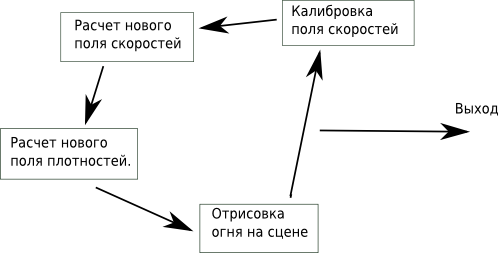


Рисунок 1.

**3.3. Разработка и обоснование используемых типов и структур данных.**

**3.4. Разработка структуры программного комплекса.**

**3.5. Применяемые инструменты.**

4. ТЕХНИЧЕСКИЙ РАЗДЕЛ.