**Санкт-Петербургский Государственный Университет**

**Математико-механический факультет**

Кафедра системного программирования

**Разработка 3D пиксельного графического движка**

Курсовая работа студента 345 группы  
Байцеровой Юлии Сергеевны

Научный руководитель **Оносовский В.В.**

(Исполнительный директор Exigen Services,

выпускник кафедры системного программирования 1985 г.)

Санкт-Петербург

2012

**Оглавление**

Введение…………………………………………………………………………………....3

Постановка задачи…………………………………………………………………………3

Принципы формирования графических изображений……………………………….4

Графическое средство……………………………………………………………………8

Основные возможности OpenGL……………………………………………………….10

Предлагаемое решение…………………………………………………………………..10

Описание решения……………………………………………………………………….11

Проблемы……………………………………………………………………………….11

Выводы……………………………………………………………………………………12

Результаты…………………………………………….……..………………………….12

Список литературы……………………………………………………………………..13

**Введение**

Считается, что 80% информации об окружающей среде человек получает посредством зрения. Но, к сожалению, мы зачастую не имеем возможность увидеть желаемое. К решению этой проблемы в разные времена подходили по-разному: рисовали картины, лепили из глины, складывали из бумаги. В современном же мире решением стали компьютерные игры. В настоящее время рынок компьютерных игр удивляет нас своими возможностями, графикой и разнообразием тематик. И первое неизгладимое впечатление от игры создает, конечно же, ее графическая составляющая. Выбор графических технологий и инструментов широк и разнообразен. Остается лишь выбрать подходящий и творить!

**Постановка задачи**

В рамках данной курсовой мной разрабатывался 3D пиксельный графический движок для игры «Лабиринт» для мобильного телефона на платформе Android. Лабиринт представляет собой трехмерное многоуровневое поле, по которому перемещается шарик. Благодаря большому числу графических средств и методов, возникают естественные вопросы, такие как:

1. Как отобразить лабиринт?
2. Какие графические средства для этого выбрать?
3. Как легко сгенерировать поверхность уровня, не потеряв при этом производительности?

Таким образом, задача разбивается на несколько подзадач:

1. Изучение принципов формирования графических изображений
2. Растровая компьютерная графика
3. Векторная компьютерная графика
4. Фрактальная компьютерная графика
5. Выбор принципа отображения для дальнейшего использования
6. Выбор графического средства
7. Изучение существующих графических библиотек
8. Сравнение их возможностей
9. Выбор инструмента
10. Детальное изучение выбранной библиотеки
11. Изучение методов генерации данных
12. Оптимизация
13. Создание реалистичности модели
14. Интеграция с другими участниками команды разработчиков с целью получения работающего приложения

**Цель:** Написать графическое отображение лабиринта – набора растровых отображений, для телефонов на базе Android, используя библиотеку OpenGL ES.

**Принципы формирования графических изображений**

**Растровое изображение** — это изображение, составленное из пикселов — маленьких цветных квадратиков, размещенных в прямоугольной сетке. Пиксел — это самая маленькая единица цифрового изображения.

#### Достоинства растровой графики:

1. Возможность воспроизведения изображений любого уровня сложности. Количество деталей, воспроизводимых на изображении, во многом зависит от количества пикселов.
2. Точная передача цветовых переходов.
3. Наличие множества программ для отображения и редактирования растровой графики. Абсолютное большинство программ поддерживают одинаковые форматы файлов растровой графики. Растровое представление, пожалуй, самый «старый» способ хранения цифровых изображений.

#### Недостатки растровой графики:

1. Большой размер файла. Фактически для каждого пиксела приходится хранить информацию о его координатах и цвете.
2. Невозможность масштабирования (в частности, увеличения) изображения без потери качества (рис.6.2)



**Рис. 1.** Растровое изображение: 1) в уменьшенном масштабе; 2) увеличенном масштабе

Еще один несомненный плюс растрового изображения – по нему можно построить объемную 3D модель с освещением. Ввиду всех плюсов растрового изображения, в данной курсовой был выбран именно этот принцип отображения.

**Векторная графика**

Изображение, созданное в векторных программах, основывается на математических формулах, а не в координатах пикселов. Поэтому векторные файлы содержат наборы инструкций для построения геометрических объектов – линий, эллипсов, прямоугольников, многоугольников и дуг. В соответствии с этим основу векторных изображений составляют разнообразные линии или кривые, называемые векторами, или, по-другому, контурами. Каждый контур представляет собой независимый объект, который можно редактировать: перемещать, масштабировать, изменять. В соответствии с этим векторную графику часто называют также объектно-ориентированной графикой.

**Преимущества векторного способа описания графики над растровой графикой**

* Размер, занимаемой описательной частью, не зависит от реальной величины объекта, что позволяет, используя минимальное количество информации, описать сколько угодно большой объект файлом минимального размера
* В связи с тем, что информация об объекте хранится в описательной форме, можно бесконечно увеличить графический примитив, например, дугу окружности, и она останется гладкой. С другой стороны, если кривая представлена в виде ломаной линии, увеличение покажет, что она на самом деле не кривая
* Параметры объектов хранятся и могут быть легко изменены. Также это означает что перемещение, масштабирование, вращение, заполнение и т. д. не ухудшает качества рисунка. Более того, обычно указывают размеры в аппаратно-независимых единицах, которые ведут к наилучшей возможной растеризации на растровых устройствах
* При увеличении или уменьшении объектов толщина линий может быть задана постоянной величиной, независимо от реального контура

**Фундаментальные недостатки векторной графики**

* Не каждый объект может быть легко изображен в векторном виде — для подобного оригинальному изображению может потребоваться очень большое количество объектов и их сложности, что негативно влияет на количество памяти, занимаемой изображением, и на время для его отображения (отрисовки)
* Перевод векторной графики в растр достаточно прост. Но обратного пути, как правило, нет — трассировка растра, при том что требует значительных вычислительных мощностей и времени, не всегда обеспечивает высокое качество векторного рисунка
* Преимущество векторной картинки — масштабируемость — пропадает, когда начинаем иметь дело с особо малыми разрешениями графики (например, иконки 32×32 или 16×16). Чтобы не было «грязи», картинку под такие разрешения приходится подгонять вручную



**Рис. 2.** Увеличение масштаба (2) векторного изображения (1) не приводит к ухудшению его качества (в отличие от растрового изображения)

**Фрактальная графика**

Понятия фрактал, фрактальная геометрия и фрактальная графика, появившиеся в конце 70-х, сегодня прочно вошли в обиход математиков и компьютерных художников. Слово фрактал образовано от латинского fractus и в переводе означает «состоящий из фрагментов». Оно было предложено математиком Бенуа Мандельбротом в 1975 году для обозначения нерегулярных, но самоподобных структур, которыми он занимался.

Фракталом называется структура, состоящая из частей, которые в каком-то смысле подобны целому. Одним из основных свойств фракталов является самоподобие. Объект называют самоподобным, когда увеличенные части объекта походят на сам объект и друг на друга. Перефразируя это определение, можно сказать, что в простейшем случае небольшая часть фрактала содержит информацию обо всем фрактале.

Фрактальная графика, также как векторная и трёхмерная, является вычисляемой. Её главное отличие в том, что изображение строится по уравнению или системе уравнений. Поэтому в памяти компьютера для выполнения всех вычислений, ничего кроме формулы хранить не требуется.

Только изменив коэффициенты уравнения, можно получить совершенно другое изображение. Эта идея нашла использование в компьютерной графике благодаря компактности математического аппарата, необходимого для ее реализации. Так, с помощью нескольких математических коэффициентов можно задать линии и поверхности очень сложной формы.

Её возможности трудно переоценить. Фрактальная компьютерная графика позволяет создавать абстрактные композиции, где можно реализовать такие композиционные приёмы как, горизонтали и вертикали, диагональные направления, симметрию и асимметрию и др. Сегодня немногие компьютерщики в нашей стране и за рубежом знают фрактальную графику. С чем можно сравнить фрактальное изображение? Ну, например, со сложной структурой кристалла, со снежинкой, элементы которой выстраивается в одну сложную структуру. Это свойство фрактального объекта может быть удачно использовано при составлении декоративной композиции или для создания орнамента. Сегодня разработаны алгоритмы синтеза коэффициентов фрактала, позволяющего воспроизвести копию любой картинки сколь угодно близкой к исходному оригиналу.

С точки зрения машинной графики фрактальная геометрия незаменима при генерации искусственных облаков, гор, поверхности моря. Фактически благодаря фрактальной графике найден способ эффективной реализации сложных неевклидовых объектов, образы которых весьма похожи на природные. Геометрические фракталы на экране компьютера (рис. 3) — это узоры, построенные самим компьютером по заданной программе.



**Рис. 3.** Пример фрактального изображения

**Графическое средство**

Графическая система OpenGL – это программный интерфейс к графическому оборудованию (GL обозначает Graphics Library – графическая библиотека). Она позволяет создавать интерактивные программы, которые формируют движущиеся цветные изображения в трехмерном пространстве. С помощью OpenGL можно управлять технологиями машинной графики, создавая изображения реальных объектов или воплощая другие плоды фантазии.

На данный момент OpenGL является одним из самых популярных графических стандартов во всём мире. Ещё в 1982 г. в Стенфордском университете была разработана концепция графической машины, на основе которой фирма Silicon Graphics в своей рабочей станции Silicon IRIS реализовала конвейер рендеринга. Таким образом была разработана графическая библиотека IRIS GL. На основе библиотеки IRIS GL, в 1992 году был разработан и утверждён графический стандарт OpenGL. Разработчики OpenGL - это крупнейшие фирмы разработчики как оборудования так и программного обеспечения: Silicon Graphics, Inc., Microsoft, IBM Corporation, Sun Microsystems, Inc., Digital Equipment Corporation (DEC), Evans & Sutherland, Hewlett-Packard Corporation, Intel Corporation и Intergraph Corporation.

OpenGL переводится как «Открытая Графическая Библиотека» (Open Graphics Library), это означает, что OpenGL - это открытый и мобильный стандарт. Программы, написанные с помощью OpenGL можно переносить практически на любые платформы, получая при этом одинаковый результат, будь это графическая станция или суперкомпьютер. OpenGL освобождает программиста от написания программ для конкретного оборудования. Если устройство поддерживает какую-то функцию, то эта функция выполняется аппаратно, если нет, то библиотека выполняет её программно.

Что же представляет из себя OpenGL? С точки зрения программиста OpenGL - это программный интерфейс для графических устройств, таких как графические ускорители. Он включает в себя около 150 различных команд, с помощью которых программист может определять различные объекты и производить рендеринг. Говоря более простым языком, вы определяете объекты, задаёте их местоположение в трёхмерном пространстве, определяете другие параметры (поворот, масштаб), задаете свойства объектов (цвет, текстура, материал), положение наблюдателя, а библиотека OpenGL позаботится о том чтобы отобразить всё это на экране.

OpenGL имеет хорошо продуманную внутреннюю структуру и довольно простой процедурный интерфейс. Несмотря на это с помощью OpenGL можно создавать сложные и мощные программные комплексы, затрачивая при этом минимальное время по сравнению с другими графическими библиотеками.

В некоторых библиотеках OpenGL (например под X Windows) имеется возможность изображать результат не только на локальной машине, но также и по сети. Приложение, которое вырабатывает команды OpenGL называется клиентом, а приложение, которое получает эти команды и отображает результат - сервером. Таким образом можно строить очень мощные воспроизводящие комплексы на основе нескольких рабочих станций или серверов, соединённых сетью.

**Основные возможности OpenGL**

Что предоставляет библиотека в распоряжение программиста? Основные возможности:

* Геометрические и растровые примитивы. На основе геометрических и растровых примитивов строятся все объекты. Из геометрических примитивов библиотека предоставляет: точки, линии, полигоны. Из растровых: битовый массив(bitmap) и образ(image)
* Видовые и модельные преобразования. С помощью этих преобразований можно располагать обьекты в пространстве, вращать их, изменять форму, а также изменять положение камеры из которой ведётся наблюдение.

**Работа с цветом.** OpenGL предоставляет программисту возможность работы с цветом в режиме RGBA (Red-Green-Blue-Alpha) или используя индексный режим, где цвет выбирается из палитры.

* Удаление невидимых линий и поверхностей. Z-буферизация.
* Двойная буферизация. OpenGL предоставляет как одинарную, так и двойную буферизацию. Двойная буферизация используется для того, чтобы устранить мерцание при мультипликации, т.е. изображение каждого кадра сначала рисуется во втором(невидимом) буфере, а потом, когда кадр полностью нарисован, весь буфер отображается на экране.
* Наложение текстуры. Позволяет придавать объектам реалистичность. На объект, например шар, накладывается текстура (просто какое-то изображение), в результате чего наш объект теперь выглядит не просто как шар, а как разноцветный мячик.
* Освещение. Позволяет задавать источники света, их расположение, интенсивность, и т.д.
* Атмосферные эффекты. Например туман, дым. Всё это также позволяет придать объектам или сцене реалистичность, а также "почувствовать" глубину сцены.
* Прозрачность объектов.
* Использование списков изображений.

**Предлагаемое решение**

* 1. Составление карты высот
  2. Использование графических примитивов – треугольников – для построения полигонной сетки
  3. Придание модели реалистичности благодаря освещению

**Описание решения**

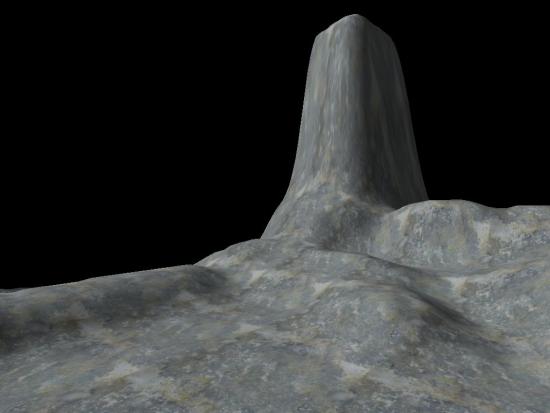
1. Генерация вершин по карте высот путем деления исходного растрового изображения на пиксели, где в качестве одной из координат вершин берется одна компонента цвета.
2. Построение полигонной сетки уровня по вершинам, построенным в первом пункте.
3. Текстурирование получившейся сетки несколькими текстурами, учитывая ее рельеф

**Проблемы**

Среди встретившихся проблем стоит упомянуть следующее:

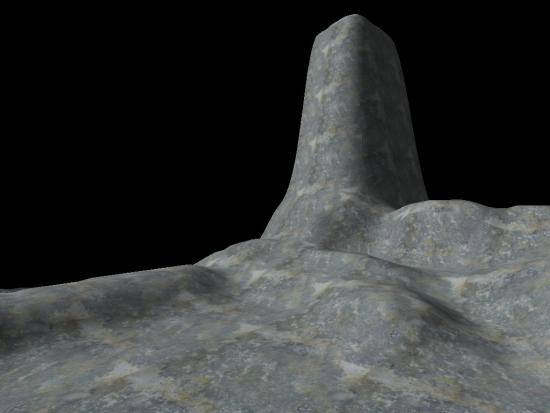
* Растягивание текстуры

Из-за большой разницы между высотами соседних треугольников, текстура, натягиваемая на них, может растягиваться (рис. 4).



**Рис. 4.** Растянутая текстура

Возникает естественная потребность исправить этот дефект. Он устраняется путем уложения текстуры в этот промежуток несколько раз (Рис. 5).



**Рис. 5.** Текстура без растяжения

**Выводы**

* Реализованное решение позволяет создавать сложную геометрию по простым картам
* Поддерживается несколько «этажей» в одном уровне
* Поддерживается возможность использования нескольких материалов

**Результаты**

Реализовано построение 3D модели поверхности уровня, текстурирование поверхности несколькими текстурами, освещение модели. Проведена оптимизация генерации вертексного, индексного буферов и буфера нормалей.

**Список литературы**

* 1. M. Woo, T. Davis, J. Neider, D. Shreiner. OpenGL, Programming Guide, 4th edition, 2006г.
  2. М. Н. Петров, В. П. Молочков. Компьютерная графика, 2003г.
  3. <http://blog.jayway.com/author/pererikbergman/> - OpenGL ES Tutorial For Android
  4. <http://pmg.org.ru/nehe/index.html>
  5. <http://www.gamedev.ru/code/terms/tags/OpenGL>
  6. <http://www.opengl.org/>