Санкт-Петербургский Государственный Университет
Математико-механический факультет

Кафедра системного программирования

Генерирование 3D ландшафта

Курсовая работа студента 344 группы
Ефремовой Варвары Андреевны

Научный руководитель: К. В. Праздников

Санкт-Петербург
2015

Оглавление

Введение

Компьютерные игры, кинематорграф, архитектура - это далеко не все сферы применения трёхмерной графики.

Одним из вариантов её использования также являются приложения для моделирование различных процессов и объектов. А поскольку люди привыкли видеть объёмные фигуры вокруг себя, то и создатели приложений стремятся предоставить пользователю трёхмерные объекты, наиболее приближенные к окружающему миру.

Один из примеров моделирования — приложение CoachingSpaces[1], которое предоставляет инструмент для удаленного коучинга[2]. Идея приложения состоит в том, что используя различные объекты (геометрические фигуры, модели людей, животных и другие различные предметы окружающего мира), пользователь может смоделировать какую-либо ситуацию (Рис. 1).

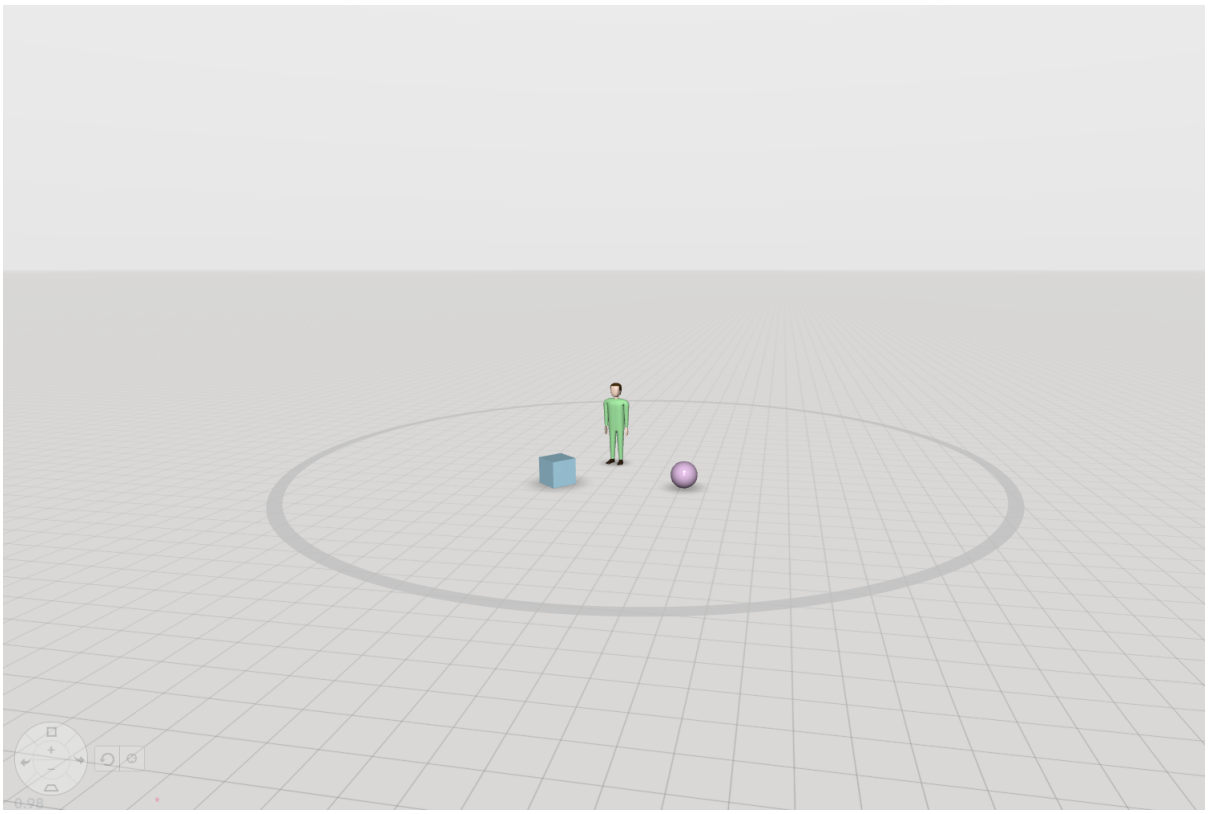


Рис.1: Пример сцены в приложении CoachingSpaces

Но для создания настроения хочется создать какой-то окружающий мир. Например, пустыню, снег или зеленый луг. И дать пользователю возможность его настройки: выбор типа ландшафта, цвета, освещения.

Это и является задачей данной курсовой работы.

Постановка задачи

Целью данной работы является генерирование 3D ландшафта в приложении CoachingSpaces.

В связи с этим были выделены следующие задачи:

1. Изучение уже существующих способов построения местности, включающие в себя моделирование при помощи графического редактора, а так же изучение различных графических библиотек.
2. Изучение технологий, необходимых для реализации решения.
3. Анализ и проработка существующих алгоритмов построения рельефа.
4. Разработка и реализация алгоритма построения рельефа.
5. Создание модели, позволяющей настроить различные параметры ландшафта.

Delightex project

CoachingSpaces — Web-приложение для моделирования различных ситуаций. Так же недавно вышла мобильная версия данного приложения.

Приложение написано на языке Java с применением технологии OpenGL[3] для отрисовки графики. Отображениие в HTML происходит с помощью технологии Google Web Toolkit (GWT)[4], которая позволяет транслировать программный код Java[5] в язык JavaScript[6] и запускать его в браузере клиента.

Именно в этот проект было интегрировано полученное решение.

Обзор существующих решений

Существует несколько вариантов создания ландшафта.

Графические редакторы

С помощью графических редакторов, таких, как 3ds Max, Blender и др., можно создавать объекты, изменение координат точек которых возможно лишь в самом редакторе. А в данной задаче хотелось иметь возможность изменять различные физические параметры рельефа со стороны клиента. Именно по этой причине использование графических редакторов представлялось невозможным.

Готовые графические библиотеки

Было изучено несколько графических библиотек, которые позволяют решить данную задачу.

Virtual Terrain Prject[7] — инструмент для построения любой части ореального мира в цифровом 3D формате. Но он требует большое количество памяти. А для данного приложения хочется найти оптимальное решение, не требующее больших ресурсозатрат.

World Machine 3D[8] — также является инструментом для создания реалистичной местности большого масштаба. В основном предназначается для создания карт для игр. Здесь так же встал вопрос производительности; также эта технология не предоставляла возможность создания пользовательского интефейса для настройки различных параметров ландшафта.

После изучения различных существующих решений для построения ландшафта было принято решение разработать подходящую модель местности для данной задачи.

Этапы работы

Вся задача была разделена на несколько этапов:

1. Построение карты высот.
2. Раскраска и реализация световой модели Фонга.
3. Интегрирование решения в приложение.

Именно первый этап потребовал наибольшее количество времени и усилий. Далее рассматриваются варианты решения задачи построения карты высот.

Обзор алгоритмов для построения карты высот

Random

Самое первое, что приходит в голову, это использование встроенной функции для генерации пвсевдослучайной равномерно распределенной величины для задания высоты в каждой точке. Но это дает совершенно неприемлемый результат. Получается не карта высот, а хаотичное положение точек в пространстве.

Другой возможный вариант — использование этой функции с различными весами. Например, если точка находится ближе к центру сцены, то вес — меньше, преобладание Random’а меньше; чем дальше от центра — тем вес больше, случайные значение пребладают.

Но это не сильно изменило предыдущий результат.

Шум Перлина

Шум Перлина — математический алгоритм для генерации процедурной текстуры псевдо-случайным методом.

Этот способ дает более реалистичные результаты, но только для более ровных поверхностей, а в создаваемой модели хотелось иметь возможность генерирования горного рельефа.

Холмовой алгоритм

Холмовой алгоритм генерирует холм на регулярной сетке, получив на вход его радиус. В зависимости от радиуса можем получить более скалистый подъем или небольшой пологий холмик.

Основная формула для получения холма:

$$z = R^2 - ((x_2 - x_1)^2 + (y_2 - y_1)^2), \text{ где}$$
$$(x_1, y_1) - \text{ координаты точки на плоскости } z = 0,$$
$$z - \text{ высота в этой точке,}$$
$$(x_2, y_2) - \text{ координаты точки пика холма на плоскости } z = 0,$$
$$R - \text{ радиус холма}$$

Но для генерирования большой карты высот необходимо большое количество итераций данного алгоритма, что отрицательно повлияло на производительность.

Моделирование эрозии

Так же одиним из вариантов построения карты высот — это моделирование физического поведения рельефа во времени. Генерируется произвольная карта высот. Далее большим количеством итераций моделируется изнашивание поверхности со временим. В этом способе построения остро встает проблема производительности.

Алгоритм Diamond Square

Diamond square алгоритм оказался одним из наиболее подходящих для решения поставленной задачи. Он генерирует карту высот для регулярной сетки, тогда как предыдущие алгоритмы позволяют делать это для нерегулярной сетки. Основная идея алгоритма заключается в прохождении по сетке уменьшающимися квадратами и вычислении высот в центре квадрата и на серединах его сторон по значениям в угловых точках с добавлением случайного значения.

Пример работы алгоритма:

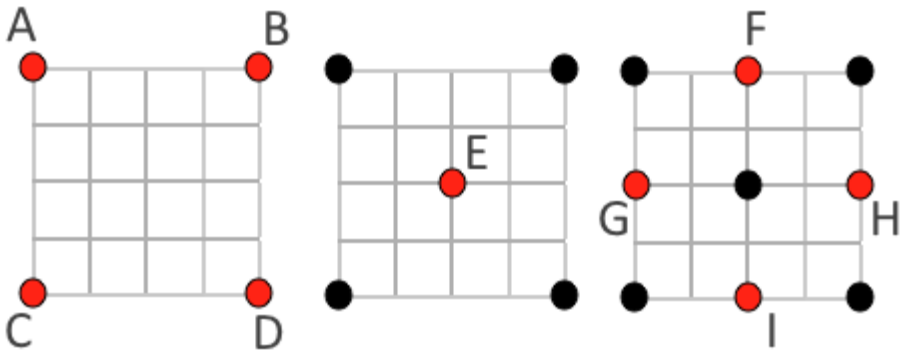
1) Задаются высоты в угловых точках (Рис. 2(а)).
2) Считается значение в центральной точке квадрата (Рис. 2(б)):

$$E = \frac{A+B+C+D}{4} + rand_1$$

3) Считаются значения на серединах сторон (Рис. 2(в)):

$$F = \frac{A+B}{2} + rand_2,\ G = \frac{A+C}{2} + rand_3,$$
$$H = \frac{B+D}{2} + rand_4,\ I = \frac{C+D}{2} + rand_5$$

4) Рассматриваются меньшие квадраты AFEG, BFEH, CGEI, DHEI. Для них повторяются шаги алгоритма, пока квадраты не станут нужного размера.



а) б) в)

Рис. 2

Алгоритм подходил нам по произовдительности и использованию ресурсов. Но хотелось получить и равнину с горами по краям. И данный алгоритм почти позволял это сделать.

Модификация алгоритма Diamond Square

В рассмотрении алгоритма генерирования карты с помощью Random() одной из его модификаций было генерирование значений высот с весом, зависящим от удаленности точки от центра сцены. Данная модицикация была успешна применена в алгоритме diamond square. Псевдокод функции веса:

```
float feild_radius  = … ; // feild_radius  - радиус окружающей поляны
float weight(float radius) {          // radius - расстояние рассматриваемой точки
      return radius / feild_radius;       //                              от цента сцены
}
```

Полученный результат был близок к тому, что хотелось получить. В ходе работы возникли препятствия, связанные со спецификой поставленных требований.

Рендеринг 3D объектов происходит с помощью отрисовки полигонов (в данном проекте полигоны — это треугольники). То есть после создания карты высот на регулярной сетке, квадраты разбиваются на треугольники и происходит их отрисовка.

Но при использовании регулярной сетки и создании на ней карты высот, возникал скачок высот на границе поле-горы. И появлялись длинные остроугольные треугольники, которые портили вид.

Решением это проблемы стал последующий отказ от регулярной сетки. Остроугольные треугольники объединялись между собой, и в область между ними добавлялись новые точки. Большое количество точек, находящихся на довольно плоской части рельефа, никак не влияло на визуальный эффект, а их обработка занимала значительную часть

времени работы алгоритма. Было принято решение удалить их. Таким образом у нас появилась хорошая карта высот, но она была несвязной. А для отрисовки модели нам нужны именно точки, объединенные в треугольники.

Триангуляция Делоне

Триангуляция Делоне — триангуляция для заданного множества точек P на плоскости, при которой для любого треугольника все точки из P за исключением точек, являющихся его вершинами, лежат вне окружности, описанной вокруг треугольника.

Был реализован алгоритм алгоритм «Разделяй и властвуй». Проверка выполения условия Делоне осуществлялась с помощью модифицированной проверки суммы противолежащих углов.

Таким образом, была получена подходящая для нашей задачи карта высот.

Mesh (множество точек с атрибутами, объединенные в треугольники) состоял из точек с координатами в трёхмерном пространстве и координатами нормали. Точки дублировались для каждого треугольника, чтобы добиться эффекта “треугольности” рельефа.

Раскраска и модель освещения Фонга

Модель освещения Фонга[9] — модель освещения трёхмерных объектов. Расчёт освещения по Фонгу требует вычисления цветовой интенсивности трёх компонент освещения: фоновой (ambient), рассеянной (diffuse) и глянцевых бликов (specular).

Для отрисовки трёхмерной модели были написаны программы вершинных и фрагментов шейдеров. В вершинном шейдере в зависимости от удаленности, высоты и нормали, вычислялся цвет, в

который будет покрашена данная точка. В фрагментном шейдере производится покраска в соответствующий цвет с учетом модели Фонга и выбранного пользователем освещения.

Интегрирование в приложение

Были изучены архитектура классов в проекте, параметры для сохранения модели в базу данных, способ синхронизации сцен между пользователями, различные контроллеры для управления объектами. Решение было успешно интегрировано в приложение.

Заключение

В рамках данной работы был проведен анализ существующих различных подходов решения задачи генерирования 3D ландшафта. Исходя из особенностей и специфики поставленных требований и существующих алгоритмов, был выбран подход позволяющий сократить потребляемые ресурсы и дающий требуемый результат.

Для генерирования 3D ландшафта

- Разработана модель для генерирования 3D ландшафта,
- Решение успешно интегрировано в приложение CoachingSpaces. (примеры работы на Рис. 3 - 4)

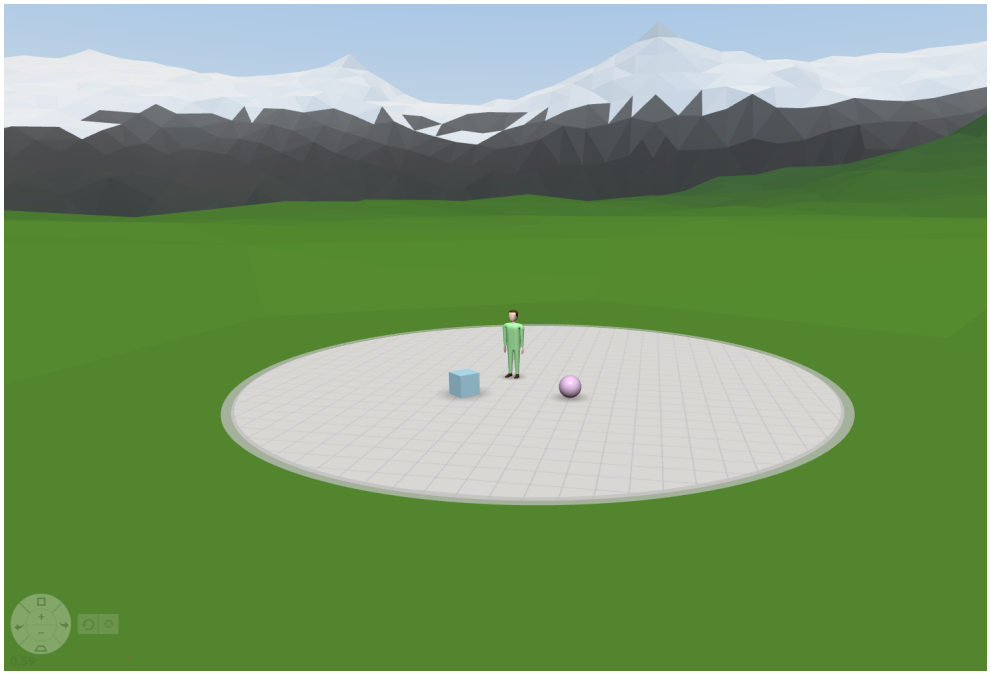
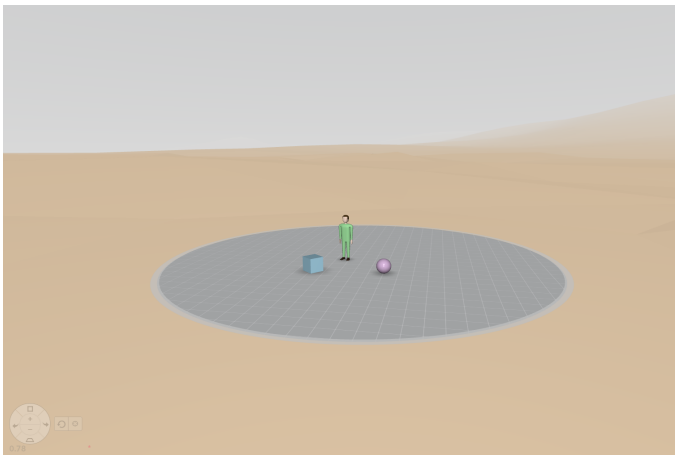
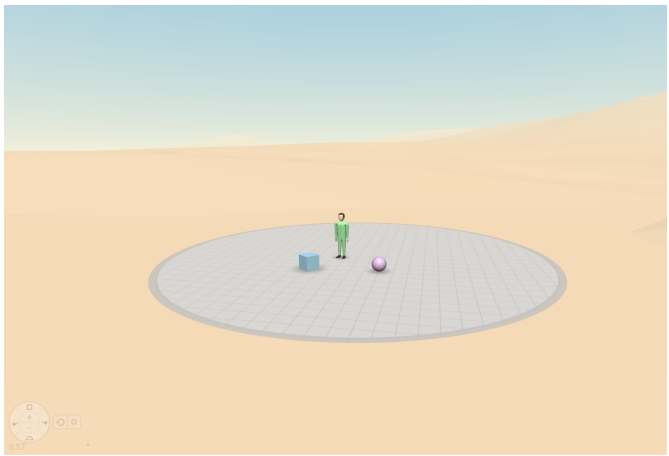


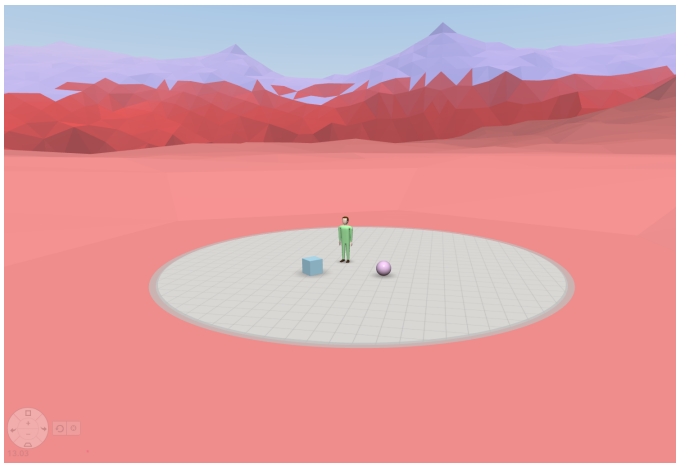
Рис. 3

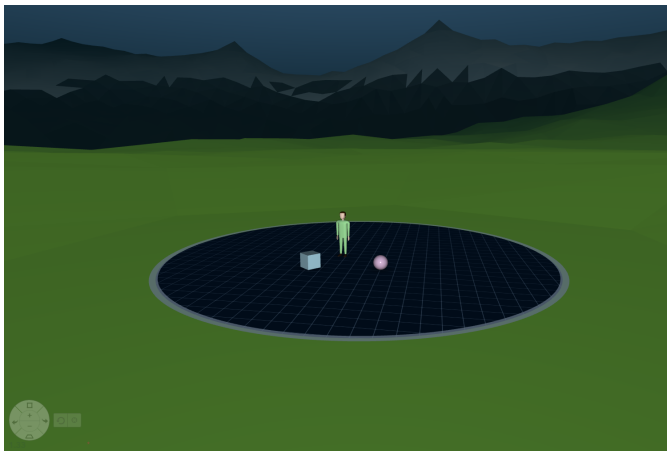


а)



б)





в) г)

Рис. 4

Список используемой литературы

[1] URL: http://www.coachingspaces.com/ дата обращения 28.05.2015.

[2] URL: https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D1%83%D1%87%D0%B8%D0%BD%D0%B3 дата обращения 28.05.2015.

[3] URL: https://www.opengl.org/ дата обращения 28.05.2015.

[4] URL: http://www.gwtproject.org/ дата обращения 28.05.2015.

[5] URL: https://www.java.com дата обращения 28.05.2015.

[6] URL: http://vterrain.org/ дата обращения 28.05.2015.

[7] URL: http://www.world-machine.com/ дата обращения 28.05.2015.

[8] Скворцов А. В. «Триангуляция Делоне и её применение», 2002.

[9] URL: https://ru.wikipedia.org/wiki/%D0%97%D0%B0%D1%82%D0%B5%D0%BD%D0%B5%D0%BD%D0%B8%D0%B5_%D0%BF%D0%BE_%D0%A4%D0%BE%D0%BD%D0%B3%D1%83 дата обращения 28.05.2015.

[10]P. Cignoniz , C. Montaniz , R. Scopigno «DeWall: A Fast Divide & Conquer Delaunay Triangulation Algorithm», 1997.

[11] S. Gustavson «Simplex Noise Demystified», 2005.

[12] William L. Raffe, Fabio Zambetta, and Xiaodong Li «A Survey of Procedural Terrain Generation Techniques using Evolutionary Algorithms», 2012.