ПРОЦЕДУРНАЯ ГЕНЕРАЦИЯ ТЕКСТУР НА ОСНОВЕ АЛГОРИТМА DIAMOND-SQUARE

Институт информационных технологий БГУИР,
г. Минск, Республика Беларусь

Мультан Р.И. Мазур А.Д.

Бакунова О.М.- ст. преподаватель каф. ИСиТ, м.т.н.
Бакунов А.М.-ст. преподаватель каф. ИСиТ, м.т.н.
Образцова О.Н. - доцент каф. ИСиТ, к.т.н,, доцент
Калитеня И.Л. - ассистент каф. ИСиТ, м.т.н..

В данной статье рассмотрен алгоритм diamond-square(далее DS), его практические реализации. Также будет рассмотрено использование этого алгоритма для того, чтобы процедурно сгенерировать текстуру.

Алгоритм DS представляет собой расширенную версию алгоритма midpoint displacement. Расширенная реализация заключается в использовании двухмерной плоскости и наличием 2 этапов (рисунок 1). Первый этап— “square” — на данном этапе каждый элемент массива — ромб, для него определяется центральная точка, для которой считается среднее значение из крайних точек и добавляется рандомное смещение. Второй этап — “diamond”— каждому квадрату в массиве определяется срединная точка, которой устанавливается среднее значение угловых точек и добавляется рандомное смещение. На каждом этапе и при каждой итерации случайное отклонение, которое прибавляется к срединным точкам, уменьшается. На выходе у него получается карта высот, точки в ней расположены по сетке. Таким образом, выходит, что вся плоскость покрыта квадратами.

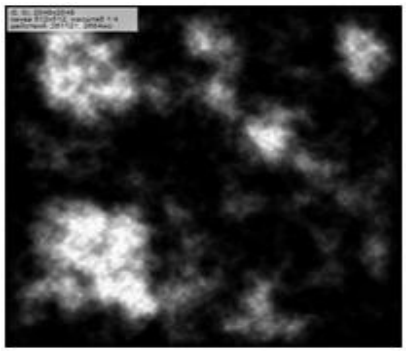

Рисунок 1 – Реализация в двухмерной плоскости алгоритма DS

С помощью алгоритма DS генерируют весьма реалистичные ландшафты, которые называют фрактальными. Но при максимальном увеличении можно заметить небольшое количество мелких островков вместо равнин и морей. Разрешить такого рода проблему можно комбинируя фрактальные ландшафты различной размерности. Значения таких ландшафтов можно складывать, перемножать, используя различные коэффициенты. Чтобы сделать равнины более пологими, а склоны гор более крутыми, можно нормализованные значения возводить в квадрат. Можно экспериментировать не стесняясь.

Для того, чтобы создать процедурно сгенерированную текстуру существует несколько шагов. В основе генерации лежит вышеописанный алгоритм DS, который поможет нам получить карту высот. Сначала необходим двумерный массив размерностью 2^n+1. Далее необходимо реализовать два основных шага DS – square и diamond (рисунок 2).

```
public static int ysize = 1025, xsize = ysize * 2 - 1;
public static float[,] heighmap = new float[xsize, ysize];
public static float roughness = 2f;

public static void Square(int lx, int ly, int rx, int ry)
{
    int l = (rx - lx) / 2;

    float a = heighmap[lx, ly];             //  B--------C
    float b = heighmap[lx, ry];             //  |        |
    float c = heighmap[rx, ry];             //  |   ce   |
    float d = heighmap[rx, ly];             //  |        |
    int cex = lx + l;                       //  A--------D
    int cey = ly + l;

    heighmap[cex, cey] = (a + b + c + d) / 4 + Random.Range(-l * 2 * roughness / ysize, l * 2 * roughness / ysize);
}
```

Рисунок 2 - Создание процедурно сгенерированной текстуры

Square будет принимать две противоположные точки квадрата — левую нижнюю и правую верхнюю, и потом сохранять значение в центр квадрата. Следующий метод — diamond — будет принимать значение точки, у которой нужно вычислять центральную точку сторон квадрата. Далее вычисляет точку на базисе значений перекрестных углов квадрата, центра и центральной точки соседнего квадрата. Ниже показана реализация одного из шагов DS:

Далее необходимо проверить координаты всех точек, не выходят ли они за пределы массива. Когда это все же случается, то точке необходимо присвоить обратное ей значение (рисунок 3). После этого если попробовать нанести текстуру на шар, то получается бесшовная текстура. Для следующего шага нам необходимо написать небольшой метод объединяющий их в квадрат.

```
public static void DiamondSquare(int lx, int ly, int rx, int ry)
{
    int l = (rx - lx) / 2;

    Square(lx, ly, rx, ry);

    Diamond(lx, ly + l, l);
    Diamond(rx, ry - l, l);
    Diamond(rx - l, ry, l);
    Diamond(lx + l, ly, l);
}
```

Рисунок 3 – Проверка координаты всех точек и объединение в квадрат

При использовании этого метода не используется рекурсия. Потому что при использовании метода DiamondSquare(){} только первый квадрат будет рассчитываться правильно, так как точки находятся за размерностью массива, поэтому иногда вместо этого используется центр квадрата (рисунок 4).

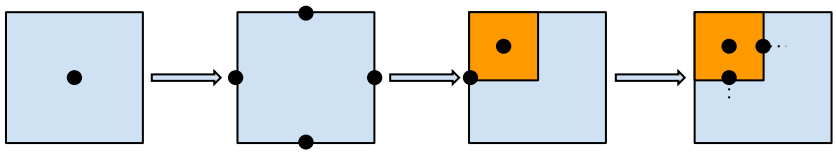

Рисунок 4 – Варианты объединения точек в квадраты

Последующие квадраты, созданные внутри, верно считаться уже не будут, потому что центры окружающих квадратов еще не подсчитаны. И получается, что использовать рекурсию не имеет смысла. Использовать надо обсчет по слоям. Он заключается в том, что надо просмотреть все длины сторон квадратов, и в каждом необходимо рассмотреть левые нижние углы в текущем слое. Выходит, что мы всегда будем знать центральные точки окружающих квадратов.

В следующем шаге необходимо перейти к цвету. Генерировать цвет мы будем при помощи midpoint displacement. В текущем этапе он будет использоваться напрямую. Т.е. он будет содержать список координат по оси y, которые сопоставлены координате по оси x. Ось x является границей поясов. Для придания поясу замкнутости необходимо делать края равными. Для пущего реализма необходимо делать различные значения roughness, таким образом, чтобы пояса могли двигаться по неожиданным траекториям. Но тут важно не зайти слишком далеко и не увлечься. Ведь выходная сгенерированная текстура должна быть приближена к реальной.

Необходимо будет инициализировать массив цветов (для этого в Unity существует метод, позволяющий читать массив, а также записывать значения элементов в текстуру) (рисунок 5). Для того чтобы сделать текстуру более реалистичной, можно возвести ее значения в степень, что придаст ей более гладкий и пологий вид. Ну и после всех этапов остается только чуть сгладить картинку и использовать нашу текстуру.

```
private Texture2D tex;
public static Color[] colors = new Color[Heighmap_class.xsize * Heighmap_class.ysize];

float waterLvl = 0.2f;
```

Рисунок 5 – Использование текстуры

Подводя итог этой статьи, необходимо понимать, что без алгоритма DS получить процедурно сгенерированную реалистичную текстуру никак не получится. Этот алгоритм может быть использован для генерации реалистичных ландшафтов, генерации миров, а также и для генерации бесконечных уровней.