Министерство образования Российской Федерации

ТАМБОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Кафедра САПР

**КУРСОВАЯ РАБОТА**

по дисциплине «Компьютерная графика»

Тема: «Моделирование растений. Фрактальная геометрия»

Выполнил студент гр. -31

Д. Ю.

Проверил: С.А.

Тамбов,

**Задание:**

Генерация фрактального дерева с помощью рекурсивной функции.

**Описание:**

Используя фрактал, называемый Деревом Пифагора (который, не изобретен Пифагором и никак не связан с теоремой Пифагора) и Броуновского движения (которое хаотично), попытаемся сделать имитацию реального дерева. Упорядочение листьев и веток на дереве довольно сложно и случайно и, вероятно не является чем-то достаточно простым [1].

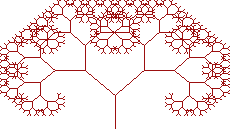
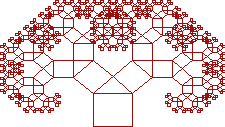
Для начала нужно сгенерировать Дерево Пифагора. Результат (Рисунок 1) напоминает те старые детсадовские рисунки. Так что давайте сделаем ствол толще. На этой стадии Броуновское движение не используется. Вместо этого, каждый отрезок линии теперь стал линией симметрии прямоугольника, который становится стволом, и веток снаружи. Результат (Рисунок 2) все еще выглядит слишком формальным и упорядоченным. Дерево еще не смотрится как живое.

Рисунок 2 Замена линий на симметрию прямоугольника

Рисунок Дерево Пифагора

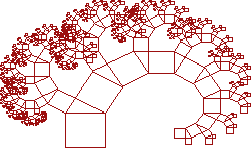
Теперь можно использовать Броуновское движение для создания некоторой случайной беспорядочности, которая изменяет числа, округляя их до двух разрядов. В оригинале были использованы 39 разрядные десятичные числа. Результат (Рисунок 3) не выглядит как дерево. Вместо этого, он выглядит как хитроумный рыболовный крючок.

Рисунок 3 Применяем алгоритм хаотичного движения

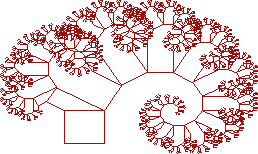
Снова применяем Броуновское движение, округленное на этот раз до 7 разрядов вместо 2. Результат по-прежнему выглядит как рыболовный крючок, но на этот раз в форме логарифмической спирали (Рисунок 4).

Рисунок 4 Результат округление до 7

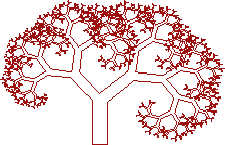
Так как левая сторона (содержащая все нечетные числа) не производит эффект крючка, случайные беспорядочности, произведенные Броуновским движением применяются дважды ко всем числам с левой стороны и только один раз к числам справа. Может быть этого будет достаточно чтобы исключить или уменьшить эффект логарифмической спирали. Итак, числа округляются до 24 разрядов. На этот раз, результат — приятно выглядящая компьютеризированная хаотическая эмуляция реального дерева (Рисунок 5).

Рисунок 5 Результат округления до 24 разрядов

**Реализация алгоритма:**

Генерация происходит при построении случайным образом (Random) листьев и веток дерева. Для этого создаётся процедура

**private** void Pictures**(**double x**,** double y**,** double a**,** int l**)**

На первом шаге эта процедура получает координаты начала дерева (x, y), угол наклона ствола (a) и длину основной ветви (l).

Во время работы процедуры проверяется длина вновь сгенерированной линии, при этом, если она слишком коротка, то будем считать её листом, а не веткой.

Координаты конца ветки вычисляются учитывая переданный угол "a":

double x1 **=** Math**.**Round**(**x **+** l**\***Math**.**Cos**(**a**));**

double y1 **=** Math**.**Round**(**y **+** l**\***Math**.**Sin**(**a**));**

Далее на основании длинны ветки, вводим дополнительную переменную (р) и ее обработку для сохранения некоторых веток длинными. Те ветви, что будут признаны короткими следующим условием if(p < 40) будут дополнительно обрисованы листвой.

При рисовании листвы случайно выбираем цвет из двух близких для листвы и циклом for (int i = 0; i < 3; i++) , где 3 - это ширина основания листа, отрисовываем лист на основании ранее вычисленных координат (x1, y1):

g**.**DrawLine**(**pen**,** **new** Point**((**int**)** x **+** i**,** **(**int**)** y**),** **new** Point**((**int**)** x1**,** **(**int**)** y1**));**

Если же длина ветви оказалась большой, рисуем ветку вместо листа в цикле for (int i = 0; i < (p / 6); i++). Здесь толщина ветки зависит от её длины.

Далее в цикле for (int i = 0; i < (9 - random.Next(6)); i++) создаётся случайное число веток или листьев из одного узла (от 0 до 9), вычисляется коэффициент длины будущих веток:

int s **=** random**.**Next**(**l **-** l **/** 6**)** **+** **(**l **/** 6**);**

и угол наклона будущих побегов:

double a1 **=** a **+** 1.6 **\*** **(**0.5 **-** random**.**NextDouble**());**

На основании s и a1 вычисляются длины будущих побегов:

x1 **=** Math**.**Round**(**x **+** s **\*** Math**.**Cos**(**a**));**

y1 **=** Math**.**Round**(**y **+** s **\*** Math**.**Sin**(**a**));**

Далее следует рекурсивный вызов процедуры Pictures() для прорисовки новых листьев и веток согласно полученному случайному количеству. При передаче данных в процедуру Pictures() изменяется случайным образом длина веток. Чем меньше будет число отнимаемое от длины ветки, тем больше будет ветвление:

Pictures**(**x1**,** y1**,** a1**,** p **-** 5 **-** random**.**Next**(**30**));**

**Исходный код программы:**

**using** System**;**

**using** System**.**Windows**.**Forms**;**

**namespace** WindowsFormsApplication1

**{**

**public** partial class Form1 **:** Form

**{**

**public** Form1**()**

**{**

InitializeComponent**();**

**}**

Random random **=** **new** Random**();**

/// <summary>

/// Рисуем фигуру

/// </summary>

/// <param name="x">начальная точка по X дерева</param>

/// <param name="y">начальная точка по Y дерева</param>

/// <param name="a">угол наклона ствола</param>

/// <param name="l">длина отрезка ветви</param>

**private** void Pictures**(**double x**,** double y**,** double a**,** int l**)**

**{**

Graphics g **=** Graphics**.**FromImage**(**pictureBox1**.**Image**);**

int p**;**

**if** **(**l **<** 8**)**

**return;**

//Координаты конца ветки вычисляются учитывая переданный угол:

double x1 **=** Math**.**Round**(**x **+** l**\***Math**.**Cos**(**a**));**

double y1 **=** Math**.**Round**(**y **+** l**\***Math**.**Sin**(**a**));**

**if** **(**l **>** 100**)**

p **=** 100**;**

**else**

p **=** l**;**

Pen pen **=** **new** Pen**(**Color**.**Black**);**

**if** **(**p **<** 40**)**

**{**//то ветки короткие и они становятся листьями

//Генерация листьев

**if** **(**random**.**Next**()** **>** 0.5**)**

pen**.**Color **=** Color**.**Green**;**

**else**

pen**.**Color **=** Color**.**FromArgb**(**90**,** 139**,** 5**);**

**for** **(**int i **=** 0**;** i **<** 3**;** i**++)**//3 - ширина основания листа

g**.**DrawLine**(**pen**,** **new** Point**((**int**)** x **+** i**,** **(**int**)** y**),** **new** Point**((**int**)** x1**,** **(**int**)** y1**));**

**}**

**else**

**{**//то ветка с длинной более 40

//Генерация веток

pen**.**Color **=** Color**.**FromArgb**(**100**,** 3**,** 4**);**

**for** **(**int i **=** 0**;** i **<** **(**p **/** 6**);** i**++)**

g**.**DrawLine**(**pen**,** **new** Point**((**int**)** x **+** i **-** **(**p**/**12**),** **(**int**)** y**),**

**new** Point**((**int**)** x1**,** **(**int**)** y1**));**

**}**

//Следующие ветки

**for** **(**int i **=** 0**;** i **<** **(**9 **-** random**.**Next**(**6**));** i**++)**

**{**

int s **=** random**.**Next**(**l **-** l **/** 6**)** **+** **(**l **/** 6**);**

double a1 **=** a **+** 1.6 **\*** **(**0.5 **-** random**.**NextDouble**());** //Угол наклона веток

x1 **=** Math**.**Round**(**x **+** s **\*** Math**.**Cos**(**a**));**

y1 **=** Math**.**Round**(**y **+** s **\*** Math**.**Sin**(**a**));**

//рекурсив:

Pictures**(**x1**,** y1**,** a1**,** p **-** 5 **-** random**.**Next**(**30**));** //Чем меньше вычитаем, тем пышнее дерево

**}**

**}**

**private** void pictureBox1\_Click**(object** sender**,** EventArgs e**)**

**{**

pictureBox1**.**Image **=** **(**Image**)new** Bitmap**(**1000**,** 1000**);**

Pictures**(**320**,** 580**,** 3 **\*** Math**.**PI **/** 2**,** 200**);**

**}**

**}**

**}**

**Пример работы приложения:**

На рисунке 6 представлено дерево, нарисованное рассмотренным выше алгоритмом.

Рисунок 6 Фрактал подобия дерева

**Список использованных источников:**

1. Фракталы и теория хаоса, URL: http://chaostarantula.narod.ru/ToC/1.htm (дата обращения 23.12.14);
2. Фрактальное дерево, основы программирования, URL: http://www.opita.net/node/603 (дата обращения 24.12.14);
3. Компьютерная графика, дерево Пифагора, URL: http://grafika.me/node/88 (дата обращения 23.12.14);