Задание №1. Фрактальные изображения

Начало: 10 февраля 2004 года.
Конец: 24 февраля 2004 года (23:59)

Авторы задания:
Анастасия Куликова (sergevna@mail333.com)

Цель задания

Рассчитать (желательно с приемлемой скоростью) и отобразить фрактальное изображение.

Описание задания

Введение

Рассматривая некоторые последовательности комплексных чисел и выбирая закраску точки в зависмости от скорости сходимости последовательности, можно получить интересные изображения.

Обязательная часть

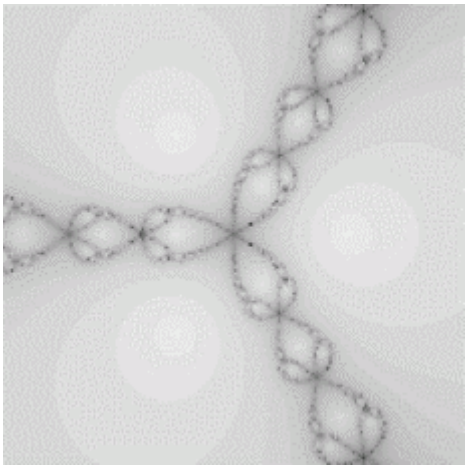
Здесь кратко рассмотрим наиболее простые методы генерации фракталов. Для выполнения задания достаточно реализовать какой-нибудь из них. Более подробную информацию можно посмотреть на www.fractorama.com

Метод Ньютона

Метод основан на решении уравнений методом Ньютона. Решение уравнения f(z) = 0 имеет вид $z_{k+1} = N(z_k)$, где N(z) = z - f(z) / f'(z). Рассмотрим уравнение pow (z, 3) - 1 = 0. Вот как будет выглядеть построение фракталов, используя метод Ньютона для решения уравнения:

```
maxIterations = 64;
count  = 0;
z      = current;//комплексные переменные
delta  = z;
z1     = z;
while ( count < maxIterations && abs( z ) < 1e6
&& abs ( delta ) > 1e-6 )
   {
   z = z - ( z * z * z - 1 )  /  ( 3 * z * z );
   delta = z1 - z;
   z1 = z;
   }
SetColor ( ColorPalette[count * 256 / maxIterations] );
```

Палитру можно сформировать по-разному, на данной картинке - градиентный переход цветов.



Более сложный пример задания последовательности комплексных чисел

Общая идея простая. Берется некая последовательность комлексных чисел $z_n = F(z_{n-1})$, где F может быть суперпозицией различных функций. Рассматривается сходимость (расходимость) этой последовательности. Обычно достаточно рассмотреть около 30 итераций. Далее в зависимости от скорости сходимости выбирается цвет в точке. Рассмотрим пример с розой - расчет цвета в одной точке. Диапазон изменения комлексного числа current (1.41243196908302890000, 0.43820235769674321000) до (1.41299091299888270000, 0.43876197245340087000) Разность действительных частей соответствует ширине изображения, разность мнимых - высоте.

```
tolerance = 1e-6;
z         = current;
zold = [0, 0];//комлексное число, Re( zold ) = Im( zold ) = 0
xmax = xmin = xtot = 0;//вещественные числа
ymin = ymax = ytot = 0;
mmin = mmax = mtot = 0;
stop = 0;
//собственно вычисления
z += log( z );//комплексные функции
while( stop == 0 && count < 30 )//30 итераций достаточно
   {
        if( abs( Re( z ) - Re( zold ) ) <= tolerance
                && abs( Im( z ) - Im( zold ) ) <= tolerance )
           //критерий остановки
                stop = 1;
        else
        {
                sc   = sec( z ) * csc( z );//где sec(z)=1./cos(z), csc(z)=1./sin(z)
                f_z  = log( sc );
                fp_z = 1/sc * ( sc * tan( z ) - sec( z ) * pow( csc( z ), 2 ) );
                zold = z;
                z    = z - fp_z / f_z;//так определяется последовательность zn
                value = csc(z);
                //все что далее нужно для определения цвета
                x = Re( value );
                y = Im( value );
                m = abs( value );//модуль комл. числа - sqrt( x*x + y*y )
                xtot += x;
                ytot += y;
                mtot += m;
```

```
                xmin = min( xmin, x );
                xmax = max( xmax, x );
                ymin = min( ymin, y );
                ymax = max( ymax, y );
                mmin = min( mmin, m );
                mmax = max( mmax, m );
                count++;
        }
  }
//теперь собственно определение цвета - немного изменив эту часть, получаем
совсем др вид
r = g = b = 0;
if( count > 0 )
  {
        r = ((xtot/count) - xmin)/(xmax - xmin);
        g = ((ytot/count) - ymin)/(ymax - ymin);
        b = ((mtot/count) - mmin)/(mmax - mmin);
  //параметры для расчета цвета, можно менять по-разному, но так
  // чтобы 0 < xStart < 255 и 0 < xStep < 2
        rStart = 255;
        gStart = 205;
        bStart = 255;
        rStep = 1.5;
        gStep = 1.5;
        bStep = 1.0;
  //собственно расчет цвета, здесь цвета как бы "идут по кругу"
        r = -cos( 2*PI*r );
        g = -cos( 2*PI*g );
        b = -cos( 2*PI*b );
        r = r*127 + 127;
        g = g*127 + 127;
        b = b*127 + 127;
        r = r*rStep + rStart;
        g = g*gStep + gStart;
        b = b*bStep + bStart;
        r /= 360.0;
        g /= 360.0;
        b /= 360.0;
        r = sin( PI*r );
        g = sin( PI*g );
        b = sin( PI*b );
        r = r*128 + 127;
        g = g*128 + 127;
        b = b*128 + 127;
  }
SetColor( b, g, r );
```

Метод Тейлора

Вот еще один достаточно простой метод. Рассмотрим разложение exp(z) в ряд Тейлора. Z - комлексное число. Фактически раскраска определяется тем, насколько быстро сумма многочленов, взятых с некоторым коэффициентом (в его выборе можно поэксперементировать), приближается к значению экспоненты. Вот пример, как это может выглядеть. Расчет цвета в данной точке:

```
maxIterations = 15;
tolerance     = 1e-6;
z = exp (current);//current - точка на комлексной плоскости,
//соответствующая данной точке получаемого изображения
xmax = xdev = xavg = 0;
ymax = ydev = yavg = 0;
mmax = mdev = mavg = 0;
zSum = 0;
double denominator = 1;//вещественная переменная
point = exp( current );//это и будет весовой коэффициент
//совсем необязательно брать здесь экспоненту,
// выглядет это может и так point[5, 5]; point = sin (point)
// и т.д.
//цикл, не более maxIterations, пока zSum не подойдет "достаточно близко" к z
while( count < maxIterations && norm( zSum - z ) > tolerance )
  {
        zSum += exp( point ) * ( ( current - point ) ^ count ) / denominator;
        if( count > 0 )
                denominator *= ( count + 1 );
        //теперь собственно определение цвета
        value = cot( zSum );
        x = abs( value^0.1 );
        y = abs( value^0.2 );
        m = abs( value^0.4 );
        xavg = (x + xavg*count) / (count + 1);
        yavg = (y + yavg*count) / (count + 1);
        mavg = (m + mavg*count) / (count + 1);
        xdev = sqrt( (x-xavg) * (x-xavg) );
           ydev = sqrt( (y-yavg) * (y-yavg) );
        mdev = sqrt( (m-mavg) * (m-mavg) );
        xmax = max( x, xdev );
        ymax = max( y, ydev );
        mmax = max( m, mdev );
        xtot += xdev;
        ytot += ydev;
        mtot += mdev;
   }
    r = g = b = 0;
  if (count > 0)
    {
        r = (xtot/count)/ xmax;
        g = (ytot/count)/ ymax;
        b = (mtot/count)/ mmax;
        r = InterpolateColor( 1, 255, r );
        g = InterpolateColor( 1, 255, g );
        b = InterpolateColor( 1, 255, b );
    }
  SetColor (r, g, b);
```

Метод регионов

Можно сгенерировать интересные изображения, задавая определенные условия цикла. Рассмотрим механизм регионов (т.е. в цикле определяем, принадлежит ли данная точка некоторому региону). В общем виде выглядит так

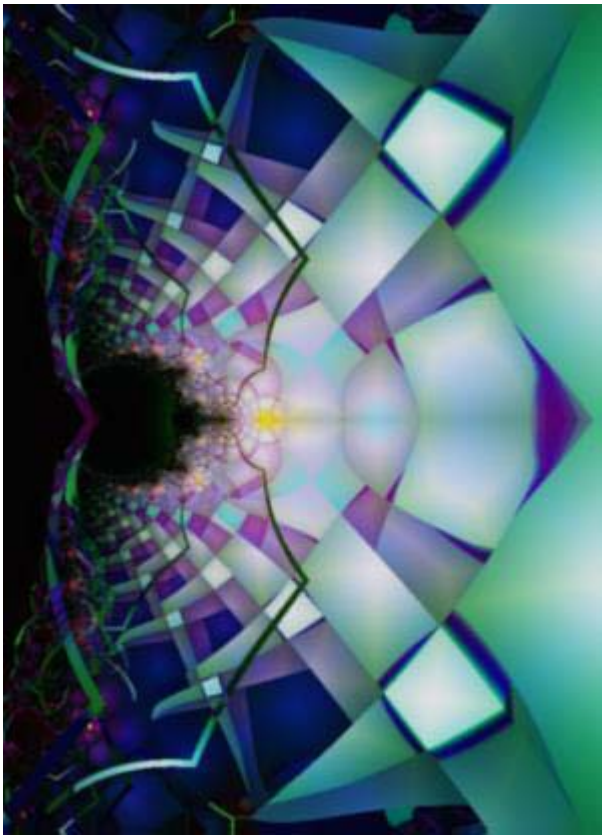
```
z = [0, 0];
while( count < 20 && squared_abs( z ) < 4 )
        {
           if( inside(z,  some region ) )
           {
              // точка z принадлежит региону, выходим из цикла
              break;
           }
           z = z ^ 2 + current;
        }
value = deg( z ) * ( 255 / 360.0 );
set_color( value, value, value );
```

В качестве региона можно взять круг, эллипс, многоугольник, но лучше их комбинации (объединение, пересечение, отсечение).

Вот пример

```
      z = [0, 0];
      xmin = 0;  ymin = 0;   mmin = 0;
      xmax = 0;  ymax = 0;   mmax = 0;


      xtot = 0;  ytot = 0;   mtot = 0;
      flag = 0;
      p   = [0, 0];
      r1  = 1;   r2 = 1.05;  r3 = 2;
      r4 = 2.05;
      while( count < 30 && squared_abs( z ) < 1e4 )
      {
         if( flag == 0 )
         {
            if( inside(  z,   region_and( region_not(
                       region_sidedpoly(p, 5, r1, 0)),
                       region_sidedpoly(p, 5, r2, 0) ) ) )
```

```
         //здесь sidedpoly( p, 5, r2, 0 ) - 5-угольник, вписанный в
         //окружность с центром в т.p и радиусом r2

            {
               flag = 1;
            }
         }
                 else
         {
            if( inside( z,   region_and( region_not(
                        region_sidedpoly(p, 5, r3, 0)),
                        region_sidedpoly(p, 5, r4, 0) ) ) )
            {
               flag = 0;
            }
         }
         if( flag == 0 )
               z = sec(z - current);
         else
               z = csc(z - current);
         z = log( z );
         value = z;
         x = real( value );
         y = imag( value );
         m =  mag( value );
         xmin = min( xmin, x );
         ymin = min( ymin, y );
         mmin = min( mmin, m );
         xmax = max( xmax, x );
         ymax = max( ymax, y );
         mmax = max( mmax, m );
         xtot += x;
         ytot += y;
         mtot += m;
      }
      r = 0; g = 0; b = 0;
      if(count > 0)
      {
         r = ( xtot/count - xmin ) / ( xmax - xmin );
         g = ( ytot/count - ymin ) / ( ymax - ymin );
         b = ( mtot/count - mmin ) / ( mmax - mmin );

         r = get_sin_color( r, 255, 1 );
         g = get_sin_color( g, 255, 1 );
         b = get_sin_color( b, 255, 1 );
      }
      set_color(r, g, b);
   }


}
```



Дополнительная часть

Желательно реализовать сохранение файла в форматы png, jpeg, tif и т.д.

Как один из несложных вариантов этого - использование классов и функций GDI+. Основные ваши действия при этом (потребуется подключить #include и добавить библиотеку gdiplus.lib, а также указать using namespace Gdiplus;)

Для инициализации GDI+ определить GdiplusStartupInput gdiplusStartupInput; ULONG_PTR gdiplusToken; и в начале программы вызвать GdiplusStartup(&gdiplusToken, &gdiplusStartupInput, NULL); При завершении работы программы вызвать GdiplusShutdown(gdiplusToken).

GDI+ по умолчанию позволяет работать со следующими форматами:

- image/bmp
- image/jpeg
- image/gif
- image/tiff
- image/png

Для сохранения изображения в желаемом формате нужно получить идентификатор требуемого кодировщик, например:

```
CLSID pngClsid;
GetEncoderClsid(L"image/png", &pngClsid);
```

```
 image.Save(L"Mosaic2.png", &pngClsid, NULL);

 //где image - переменная класса Image или его производного Bitmap, содержащая
 //сгенерированное изображение.
```

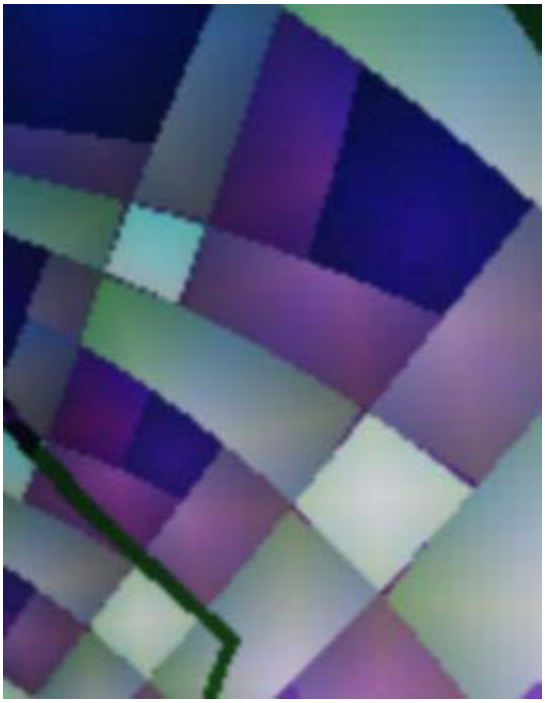
Определить идентификатор енкодера данного формата можно так.

```
int GetEncoderClsid(const WCHAR* format, CLSID* pClsid)
{
   UINT  num = 0;          // number of image encoders
   UINT  size = 0;         // size of the image encoder array in bytes
   ImageCodecInfo* pImageCodecInfo = NULL;
   GetImageEncodersSize(&num, &size);
   if(size == 0)
      return -1;  // Failure
   pImageCodecInfo = (ImageCodecInfo*)(malloc(size));
   if(pImageCodecInfo == NULL)
      return -1;  // Failure
   GetImageEncoders(num, size, pImageCodecInfo);
   for(UINT j = 0; j < num; ++j)
   {
      if( wcscmp(pImageCodecInfo[j].MimeType, format) == 0 )
      {
         *pClsid = pImageCodecInfo[j].Clsid;
         free(pImageCodecInfo);
         return j;  // Success
      }
   }
   free(pImageCodecInfo);
   return -1;  // Failure
}
```

Третий параметр в методе Image::Save() - указатель на EncoderParameters, используя его, для каждого формата можно задать определенные настройки - например, для jpeg - уровень качества. Подробнее см. документацию по GDI+.

Для повышения качества картинки рекомендуется избавиться от эффекта алиасинга(ступенчатости)

Фрактальное изображение с сильной ступенчатостью



Работая с GDI+ достаточно установить для объекта Graphics желаемое качество.

```
VOID Example_SetSetSmoothingMode(HDC hdc)
{
   Graphics graphics( hdc );
   graphics.SetSmoothingMode( SmoothingModeHighQuality );
}
```

Требования к программе

Обязательные требования

Программа должна генерировать фрактал каким-либо из предложенных способов (наиболее простым является метод Ньютона), можно сделать и своим способом, тогда в readme файле кратко опишите метод. Если расчет занимает много времени, выводите индикатор состояния.

Дополнительные требования

Будут поощряться дополнительным баллами возможность сохранения изображения в файл (особенно форматы png, jpeg - последний с регулированием уровня качества/размера файла), возможность масштабирования части изображения, реализация фрактала более оригинальным способом, а также быстрый расчет и удобный интерфейс. </p<>

Дополнительные источники информации

Дополнительную информацию по фракталам вы можете посмотреть на сайте Fractorama. Там же программа, позволяющая генерировать различные фракталы по заданным формульным данным. Есть много примеров с заданными расчетными формулами. Если есть интерес,

разобраться в этом несложно, к тому же есть достаточно подробная документация.

Оценка

Генерация фракталов методом Ньютона, или методом Тейлора, или методом регионов, или методом анализирования сходимости последовательности комплексных чисел	7 баллов
Возможность сохранения в файл (в простейшем случае bmp или targa)	+1 балл
Возможность сохранения в различные форматы файлов (jpeg с по крайней мере некоторыми настройками формата, png, tiff)	+2 балла
Возможность масштабирования изображения (при этом выполняется пересчет для той части комплексной плоскости, которая соответствует выделенной части изображения). Область для увеличения (пересчета) задавать мышкой.	+1 балл
Быстрый расчет сложных фракталов (оптимизировать скорость), повышенное качество картинки (антиалиасинг и др.), удачный интерфейс (GUI Windows), оригинальные картинки	+1, +2 балла

Оформление

См. Информацию о курсе и FAQ.

Не забудьте положить в архив файл readme.txt. В файле указать следующие подзадания:

```
генерация фракталов одним из обязательных методов [+]/[-]
сохранение в bmp/targa [+]/[-]
сохранение в различные форматы ( png, tiff, jpg.. ) [+]/[-]
масштабирование   [+]/[-]
оптимизации, улучшение, оригинальность [+]/[-], если [+],  пояснить,
что улучшено;)
```

Если вы реализовали какой-либо оригинальный метод, опишите его в комментариях.

Результаты работы

Результаты смотрите в интернете и/или стенде около лаборатории. Все вопросы присылать авторам и проверяющим.

Задание выполняется строго индивидуально. За совместную работу или обмен кусками кода ставится ноль баллов всем участникам, если факт командной работы не был указан в readme.txt заданий.

ЧаВо по заданию

У меня Windows XP и Visual Studio 6.0. Как я могу работать с GDI+?

Нужно скачать библиотечные lib и include. Их можно взять, например, здесь (zip, 113кб). Больше ничего устанавливать не нужно.

У меня установлена Platform SDK из Visual Studio 7.0. Как я могу работать с GDI+?

Ничего вообще устанавливать не нужно, вы можете работать с GDI+, даже если вся Visual Studio 7.0 у вас не установлена.

У меня Windows 2000 (Windows NT / Windows Me / Windows 98). Как я могу работать с GDI+?

Нужно скачать библиотечные lib и include. Их можно взять, например, здесь (zip, 113кб). Если у вас нет приложений, использующих gdiplus.dll (т.е. нет gdiplus.dll) - вы можете скачать ее отсюда.