(/wiki/Rosetta_Code)

ROSETTACODE.ORG

Create account (/mw/index.php?title=Special:UserLogin&returnto=Barnsley+fern&type=signup)
Log in (/mw/index.php?title=Special:UserLogin&returnto=Barnsley+fern)

Search

Page (/wiki/Barnsley_fern) Discussion (/mw/index.php?title=Talk:Barnsley_fern&action=edit&redlink=1)
Edit (/mw/index.php?title=Barnsley_fern&action=edit)
History (/mw/index.php?title=Barnsley_fern&action=history)

I'm working on modernizing Rosetta Code's infrastructure. Starting with communications. Please accept this time-limited open invite to RC's Slack. (https://join.slack.com/t/rosettacode/shared_invite/zt-glwmugtu-xpMPcqHs0u6MsK5zCmJF~Q). --Michael Mol (/wiki/User:Short_Circuit) (talk (/wiki/User_talk:Short_Circuit)) 20:59, 30 May 2020 (UTC)

Barnsley fern

A Barnsley fern (https://en.wikipedia.org/wiki/Barnsley_fern) is a fractal named after British mathematician Michael Barnsley and can be created using an iterated function system (IFS).

(/wiki/Category:Solutions_by_Programming_Task)

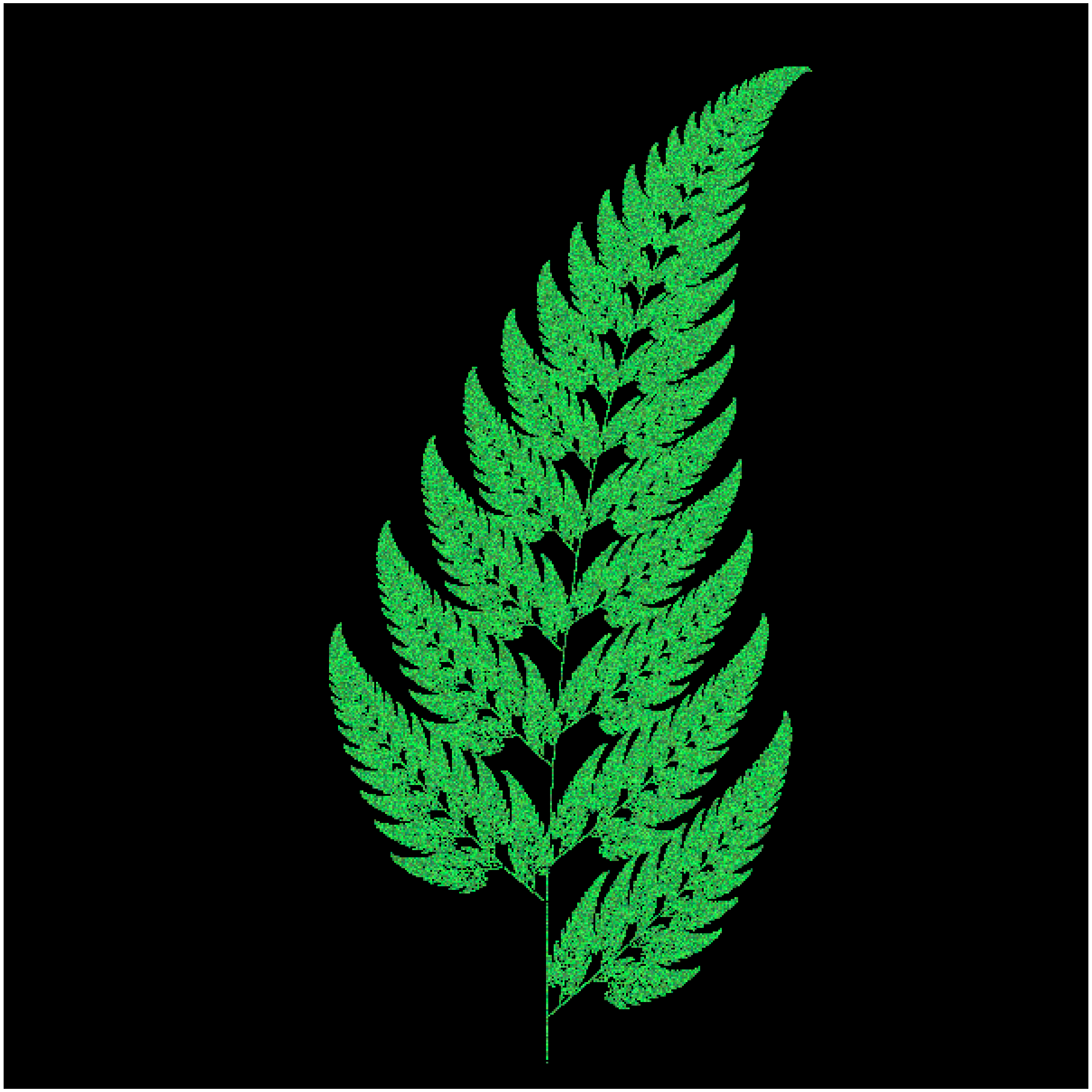
Barnsley fern
You are encouraged to solve this task (/wiki/Rosetta_Code:Solve_a_Task) according to the task

Task

Create this fractal fern, using the following transformations:

- $f1$ (chosen 1% of the time)

description, using any language you may know.



(/wiki
/File:BFCpp.png)

```
        xn + 1 = 0
        yn + 1 = 0.16 yn
```

- $f2$ (chosen 85% of the time)

```
        xn + 1 = 0.85 xn + 0.04 yn
        yn + 1 = -0.04 xn + 0.85 yn + 1.6
```

- $f3$ (chosen 7% of the time)

```
        xn + 1 = 0.2 xn - 0.26 yn
        yn + 1 = 0.23 xn + 0.22 yn + 1.6
```

- $f4$ (chosen 7% of the time)

```
        xn + 1 = -0.15 xn + 0.28 yn
        yn + 1 = 0.26 xn + 0.24 yn + 0.44.
```

Starting position: x = 0, y = 0

Contents

Ada (/wiki/Category:Ada)

Library: SDLAda (/mw/index.php?title=Category:SDLAda&action=edit&redlink=1)

```
with Ada.Numerics.Discrete_Random;

with SDL.Video.Windows.Makers;
with SDL.Video.Renderers.Makers;
with SDL.Events.Events;

procedure Barnsley_Fern is

   Iterations : constant := 1_000_000;
   Width      : constant := 500;
   Height     : constant := 750;
   Scale      : constant := 70.0;

   type Percentage is range 1 .. 100;
   package Random_Percentages is
      new Ada.Numerics.Discrete_Random (Percentage);

   Gen      : Random_Percentages.Generator;
   Window   : SDL.Video.Windows.Window;
   Renderer : SDL.Video.Renderers.Renderer;
   Event    : SDL.Events.Events.Events;

   procedure Draw_Barnsley_Fern is
      use type SDL.C.int;
      subtype F1_Range is Percentage range Percentage'First  .. Percentage'First;
      subtype F2_Range is Percentage range F1_Range'Last + 1 .. F1_Range'Last + 85;
      subtype F3_Range is Percentage range F2_Range'Last + 1 .. F2_Range'Last + 7;
      subtype F4_Range is Percentage range F3_Range'Last + 1 .. F3_Range'Last + 7;

      X0, Y0 : Float := 0.00;
      X1, Y1 : Float;
   begin
      for I in 1 .. Iterations loop
         case Random_Percentages.Random (Gen) is

            when F1_Range =>
               X1 := 0.00;
               Y1 := 0.16 * Y0;
            when F2_Range =>
               X1 :=  0.85 * X0 + 0.04 * Y0;
               Y1 := -0.04 * X0 + 0.85 * Y0 + 1.60;

            when F3_Range =>
               X1 := 0.20 * X0 - 0.26 * Y0;
               Y1 := 0.23 * X0 + 0.22 * Y0 + 1.60;

            when F4_Range =>
               X1 := -0.15 * X0 + 0.28 * Y0;
               Y1 :=  0.26 * X0 + 0.24 * Y0 + 0.44;

         end case;
         Renderer.Draw (Point => (X => Width / 2 + SDL.C.int (Scale * X1),
                                  Y => Height    - SDL.C.int (Scale * Y1)));
         X0 := X1; Y0 := Y1;

      end loop;
   end Draw_Barnsley_Fern;

   procedure Wait is
      use type SDL.Events.Event_Types;
   begin
      loop
         while SDL.Events.Events.Poll (Event) loop
            if Event.Common.Event_Type = SDL.Events.Quit then
               return;
            end if;
         end loop;
      end loop;
   end Wait;

begin
   if not SDL.Initialise (Flags => SDL.Enable_Screen) then
      return;
   end if;
   SDL.Video.Windows.Makers.Create (Win      => Window,
                                    Title    => "Barnsley Fern",
                                    Position => SDL.Natural_Coordinates'(X => 10, Y
                                    Size     => SDL.Positive_Sizes'(Width, Height),
                                    Flags    => 0);
   SDL.Video.Renderers.Makers.Create  (Renderer, Window.Get_Surface);
   Renderer.Set_Draw_Colour ((0, 0, 0, 255));
   Renderer.Fill (Rectangle => (0, 0, Width, Height));
   Renderer.Set_Draw_Colour ((0, 220, 0, 255));

   Random_Percentages.Reset (Gen);
   Draw_Barnsley_Fern;
   Window.Update_Surface;

   Wait;
   Window.Finalize;
   SDL.Finalise;
end Barnsley_Fern;
```

ALGOL 68 (/wiki/Category:ALGOL_68)

Works with: ALGOL 68G (/wiki/ALGOL_68G) version any with non-standard *establish* routine
This program generates a PBM file (https://en.wikipedia.org/wiki/Netpbm_format).

```
BEGIN
  INT iterations = 300000;
  LONG REAL scale x = 40, scale y = 40;
  [0:400,-200:200]CHAR canvas;

  LONG REAL x := 0, y := 0;

  FOR i FROM 1 LWB canvas TO 1 UPB canvas DO
    FOR j FROM 2 LWB canvas TO 2 UPB canvas DO
      canvas[i,j] := "0"
  OD OD;

  canvas[0, 0] := "1";
  TO iterations DO
    REAL choice := random;
    LONG REAL xn = x, yn = y;

    IF choice < 0.01 THEN
      x := 0;
      y := 0.16 * yn
    ELIF (choice -:= 0.01) < 0.85 THEN
      x :=  0.85 * xn + 0.04 * yn;
      y := -0.04 * xn + 0.85 * yn + 1.6
    ELIF (choice -:= 0.85) < 0.07 THEN
      x := 0.2  * xn - 0.26 * yn;
      y := 0.23 * xn + 0.22 * yn + 1.6
    ELSE
      x := -0.15 * xn + 0.28 * yn;
      y :=  0.26 * xn + 0.24 * yn + 0.44
    FI;

    INT px = SHORTEN ROUND (x * scale x),
        py = SHORTEN ROUND (y * scale y);
    IF px < 2 LWB canvas OR px > 2 UPB canvas OR
       py < 1 LWB canvas OR py > 1 UPB canvas
    THEN
      print(("resize canvas. px=", px, ", py=", py, new line));
      leave
    FI;

    canvas[py, px] := "1"
  OD;

  FILE f;
  IF establish(f, "fern.pbm", stand out channel) /= 0 THEN
    print("error creating file!"); leave
  FI;
  put(f, "P1"); new line(f);
  put(f, (whole((2 UPB canvas) - (2 LWB canvas) + 1, 0), " ",
          whole((1 UPB canvas) - (1 LWB canvas) + 1, 0), new line));
  FOR i FROM 1 UPB canvas BY -1 TO 1 LWB canvas DO
    put(f, canvas[i,]); new line(f)
  OD;
  close(f);
  leave: SKIP
END
```

Applesoft BASIC (/wiki/Category:Applesoft_BASIC)

```
100  LET YY(1) = .16
110  XX(2) =    .85:XY(2) = .04
120  YX(2) =  - .04:YY(2) = .85
130  LET Y(2) = 1.6
140  XX(3) = .20:XY(3) =  - .26
150  YX(3) = .23:YY(3) =    .22
160  LET Y(3) = 1.6
170  XX(4) =  - .15:XY(4) = .28
180  YX(4) =    .26:YY(4) = .24
190  LET Y(4) = .44
200  HGR :I =  PEEK (49234)
210  HCOLOR= 1
220  LET X = 0:Y = 0
230  FOR I = 1 TO 100000
240      R =  INT ( RND (1) * 100)
250      F = (R < 7) + (R < 14) + 2
260      F = F - (R = 99)
270      X = XX(F) * X + XY(F) * Y
280      Y = YX(F) * X + YY(F) * Y
290      Y = Y + Y(F)
300      X% = 62 + X * 27.9
320      Y% = 192 - Y * 19.1
330      HPLOT X% * 2 + 1,Y%
340  NEXT
```

BBC BASIC (/wiki/Category:BBC_BASIC)

Works with: BBC BASIC for Windows (/wiki/BBC_BASIC_for_Windows)

```
GCOL 2 : REM Green Graphics Color
X=0 : Y=0
FOR I%=1 TO 100000
  R%=RND(100)
  CASE TRUE OF
    WHEN R% == 1 NewX= 0                 : NewY= .16 * Y
    WHEN R% <  9 NewX= .20 * X - .26 * Y : NewY= .23 * X + .22 * Y + 1.6
    WHEN R% < 16 NewX=-.15 * X + .28 * Y : NewY= .26 * X + .24 * Y + .44
    OTHERWISE    NewX= .85 * X + .04 * Y : NewY=-.04 * X + .85 * Y + 1.6
  ENDCASE
  X=NewX : Y=NewY
  PLOT 1000 + X * 130 , Y * 130
NEXT
END
```

C (/wiki/Category:C)

This implementation requires the WinBGIm (http://www.cs.colorado.edu/~main/bgi/cs1300/) library. Iteration starts from (0,0) as required by the task however before plotting the point is translated and scaled as negative co-ordinates are not supported by the graphics window, scaling is necessary as otherwise the fern is tiny even for large iterations (> 1000000).

```
#include<graphics.h>
#include<stdlib.h>
#include<stdio.h>
#include<time.h>

void barnsleyFern(int windowWidth, unsigned long iter){

        double x0=0,y0=0,x1,y1;
        int diceThrow;
        time_t t;
        srand (https://www.opengroup.org/onlinepubs/009695399/functions/srand.html)

        while(iter>0){
                diceThrow = rand (https://www.opengroup.org/onlinepubs/009695399/fu

                if(diceThrow==0){
                        x1 = 0;
                        y1 = 0.16*y0;
                }

                else if(diceThrow>=1 && diceThrow<=7){
                        x1 = -0.15*x0 + 0.28*y0;
                        y1 = 0.26*x0 + 0.24*y0 + 0.44;
                }

                else if(diceThrow>=8 && diceThrow<=15){
                        x1 = 0.2*x0 - 0.26*y0;
                        y1 = 0.23*x0 + 0.22*y0 + 1.6;
                }

                else{
                        x1 = 0.85*x0 + 0.04*y0;
                        y1 = -0.04*x0 + 0.85*y0 + 1.6;
                }

                putpixel(30*x1 + windowWidth/2.0,30*y1,GREEN);
                x0 = x1;
                y0 = y1;

                iter--;
        }

}

int main()
{
        unsigned long num;

        printf (https://www.opengroup.org/onlinepubs/009695399/functions/printf.html
        scanf (https://www.opengroup.org/onlinepubs/009695399/functions/scanf.html)

        initwindow(500,500,"Barnsley Fern");

        barnsleyFern(500,num);

        getch (https://www.opengroup.org/onlinepubs/009695399/functions/getch.html)

        closegraph();

        return 0;
}
```

C# (/wiki/Category:C_sharp)

```
using System;
using System.Diagnostics;
using System.Drawing;

namespace RosettaBarnsleyFern
{
    class Program
    {
        static void Main(string[] args)
        {
            const int w = 600;
            const int h = 600;
            var bm = new (https://www.google.com/search?q=new+msdn.microsoft.com) B
            var r = new (https://www.google.com/search?q=new+msdn.microsoft.com) Ra
            double x = 0;
            double y = 0;
            for (int count = 0; count < 100000; count++)
            {
                bm.SetPixel((int)(300 + 58 * x), (int)(58 * y), Color.ForestGreen);
                int roll = r.Next(100);
                double xp = x;
                if (roll < 1)
                {
                    x = 0;
                    y = 0.16 * y;
                } else if (roll < 86)
                {
                    x = 0.85 * x + 0.04 * y;
                    y = -0.04 * xp + 0.85 * y + 1.6;
                } else if (roll < 93)
                {
                    x = 0.2 * x - 0.26 * y;
                    y = 0.23 * xp + 0.22 * y + 1.6;
                } else
                {
                    x = -0.15 * x + 0.28 * y;
                    y = 0.26 * xp + 0.24 * y + 0.44;
                }
            }
            const string filename = "Fern.png";
            bm.Save(filename);
            Process.Start(filename);
        }
    }
}
```

C++ (/wiki/Category:C%2B%2B)

(/wiki/File:BFCpp.png)

```
#include <windows.h>
#include <ctime>
#include <string>

const int BMP_SIZE = 600, ITERATIONS = static_cast<int>( 15e5 );

class myBitmap {
public:
    myBitmap() : pen( NULL ), brush( NULL ), clr( 0 ), wid( 1 ) {}
    ~myBitmap() {
        DeleteObject( pen ); DeleteObject( brush );
        DeleteDC( hdc ); DeleteObject( bmp );
    }
    bool create( int w, int h ) {
        BITMAPINFO bi;
        ZeroMemory( &bi, sizeof( bi ) );
        bi.bmiHeader.biSize        = sizeof( bi.bmiHeader );
        bi.bmiHeader.biBitCount    = sizeof( DWORD ) * 8;
        bi.bmiHeader.biCompression = BI_RGB;
        bi.bmiHeader.biPlanes      = 1;
        bi.bmiHeader.biWidth       =  w;
        bi.bmiHeader.biHeight      = -h;
        HDC dc = GetDC( GetConsoleWindow() );
        bmp = CreateDIBSection( dc, &bi, DIB_RGB_COLORS, &pBits, NULL, 0 );
        if( !bmp ) return false;
        hdc = CreateCompatibleDC( dc );
        SelectObject( hdc, bmp );
        ReleaseDC( GetConsoleWindow(), dc );
        width = w; height = h;
        return true;
    }
    void clear( BYTE clr = 0 ) {
        memset( pBits, clr, width * height * sizeof( DWORD ) );
    }
    void setBrushColor( DWORD bClr ) {
        if( brush ) DeleteObject( brush );
        brush = CreateSolidBrush( bClr );
        SelectObject( hdc, brush );
    }
    void setPenColor( DWORD c ) {
        clr = c; createPen();
    }
    void setPenWidth( int w ) {
        wid = w; createPen();
    }
    void saveBitmap( std::string path ) {
        BITMAPFILEHEADER fileheader;
        BITMAPINFO       infoheader;
        BITMAP           bitmap;
        DWORD            wb;
        GetObject( bmp, sizeof( bitmap ), &bitmap );
        DWORD* dwpBits = new DWORD[bitmap.bmWidth * bitmap.bmHeight];
        ZeroMemory( dwpBits, bitmap.bmWidth * bitmap.bmHeight * sizeof( DWORD ) );
        ZeroMemory( &infoheader, sizeof( BITMAPINFO ) );
        ZeroMemory( &fileheader, sizeof( BITMAPFILEHEADER ) );
        infoheader.bmiHeader.biBitCount = sizeof( DWORD ) * 8;
        infoheader.bmiHeader.biCompression = BI_RGB;
        infoheader.bmiHeader.biPlanes = 1;
        infoheader.bmiHeader.biSize = sizeof( infoheader.bmiHeader );
        infoheader.bmiHeader.biHeight = bitmap.bmHeight;
        infoheader.bmiHeader.biWidth = bitmap.bmWidth;
        infoheader.bmiHeader.biSizeImage = bitmap.bmWidth * bitmap.bmHeight * sizeo
        fileheader.bfType    = 0x4D42;
        fileheader.bfOffBits = sizeof( infoheader.bmiHeader ) + sizeof( BITMAPFILEH
        fileheader.bfSize    = fileheader.bfOffBits + infoheader.bmiHeader.biSizeIma
        GetDIBits( hdc, bmp, 0, height, ( LPVOID )dwpBits, &infoheader, DIB_RGB_COL
        HANDLE file = CreateFile( path.c_str(), GENERIC_WRITE, 0, NULL, CREATE_ALWAY
            FILE_ATTRIBUTE_NORMAL, NULL );
        WriteFile( file, &fileheader, sizeof( BITMAPFILEHEADER ), &wb, NULL );
        WriteFile( file, &infoheader.bmiHeader, sizeof( infoheader.bmiHeader ), &wb
        WriteFile( file, dwpBits, bitmap.bmWidth * bitmap.bmHeight * 4, &wb, NULL )
        CloseHandle( file );
        delete [] dwpBits;
    }
    HDC getDC() const     { return hdc; }
    int getWidth() const  { return width; }
    int getHeight() const { return height; }
private:
    void createPen() {
        if( pen ) DeleteObject( pen );
        pen = CreatePen( PS_SOLID, wid, clr );
        SelectObject( hdc, pen );
    }
    HBITMAP bmp; HDC    hdc;
    HPEN    pen; HBRUSH brush;
    void    *pBits; int    width, height, wid;
    DWORD    clr;
};
class fern {
public:
    void draw() {
        bmp.create( BMP_SIZE, BMP_SIZE );
        float x = 0, y = 0; HDC dc = bmp.getDC();
        int hs = BMP_SIZE >> 1;
        for( int f = 0; f < ITERATIONS; f++ ) {
            SetPixel( dc, hs + static_cast<int>( x * 55.f ),
                      BMP_SIZE - 15 - static_cast<int>( y * 55.f ),
                      RGB( static_cast<int>( rnd() * 80.f ) + 20,
                           static_cast<int>( rnd() * 128.f ) + 128,
                           static_cast<int>( rnd() * 80.f ) + 30 ) );
            getXY( x, y );
        }
        bmp.saveBitmap( "./bf.bmp" );
    }
private:
    void getXY( float& x, float& y ) {
        float g, xl, yl;
        g = rnd();
        if( g < .01f ) { xl = 0; yl = .16f * y; }
        else if( g < .07f ) {
            xl = .2f * x - .26f * y;
            yl = .23f * x + .22f * y + 1.6f;
        } else if( g < .14f ) {
```

```
      xl = -.15f * x + .28f * y;
      yl = .26f * x + .24f * y + .44f;
    } else {
      xl = .85f * x + .04f * y;
      yl = -.04f * x + .85f * y + 1.6f;
    }
    x = xl; y = yl;
  }
  float rnd() {
    return static_cast<float>( rand() ) / static_cast<float>( RAND_MAX );
  }
  myBitmap bmp;
};
int main( int argc, char* argv[]) {
  srand( static_cast<unsigned>( time( 0 ) ) );
  fern f; f.draw(); return 0;
}
```

Cross-Platform Alternative

Library: Qt (/wiki/Category:Qt)

This version uses the QImage class from the Qt toolkit as an easy way to save an image in PNG format. It also uses the C++ 11 random number library. Built and tested on macOS 10.15.4 with Qt 5.12.5.

```
#include <iostream>
#include <random>
#include <vector>

#include <QImage>

bool barnsleyFern(const char* fileName, int width, int height) {
    constexpr int iterations = 1000000;
    int bytesPerLine = 4 * ((width + 3)/4);
    std::vector<uchar> imageData(bytesPerLine * height);

    std::random_device dev;
    std::mt19937 engine(dev());
    std::uniform_int_distribution<int> distribution(1, 100);

    double x = 0, y = 0;
    for (int i = 0; i < iterations; ++i) {
        int r = distribution(engine);
        double x1, y1;
        if (r == 1) {
            x1 = 0;
            y1 = 0.16 * y;
        } else if (r <= 86) {
            x1 = 0.85 * x + 0.04 * y;
            y1 = -0.04 * x + 0.85 * y + 1.6;
        } else if (r <= 93) {
            x1 = 0.2 * x - 0.26 * y;
            y1 = 0.23 * x + 0.22 * y + 1.6;
        } else {
            x1 = -0.15 * x + 0.28 * y;
            y1 = 0.26 * x + 0.24 * y + 0.44;
        }
        x = x1;
        y = y1;
        int row = height * (1 - y/11);
        int column = width * (0.5 + x/11);
        imageData[row * bytesPerLine + column] = 1;
    }
    QImage image(&imageData[0], width, height, bytesPerLine, QImage::Format_Indexed8
    QVector<QRgb> colours(2);
    colours[0] = qRgb(255, 255, 255);
    colours[1] = qRgb(0, 160, 0);
    image.setColorTable(colours);
    return image.save(fileName);
}

int main(int argc, char *argv[]) {
    if (argc != 2) {
        std::cerr << "usage: " << argv[0] << " filename\n";
        return EXIT_FAILURE;
    }
    if (!barnsleyFern(argv[1], 600, 600)) {
        std::cerr << "image generation failed\n";
        return EXIT_FAILURE;
    }
    return EXIT_SUCCESS;
}
```

Output:

See: barnsley_fern.png (https://slack-files.com/T0CNUL56D-F017EKXBC0Y-ca68fe222b) (offsite PNG image)

Common Lisp (/wiki/Category:Common_Lisp)

This code uses the `opticl` package for generating an image and saving it as a PNG file.

```
(defpackage #:barnsley-fern
  (:use #:cl
        #:opticl))

(in-package #:barnsley-fern)

(defparameter *width* 800)
(defparameter *height* 800)
(defparameter *factor* (/ *height* 13))
(defparameter *x-offset* (/ *width* 2))
(defparameter *y-offset* (/ *height* 10))

(defun f1 (x y)
  (declare (ignore x))
  (values 0 (* 0.16 y)))

(defun f2 (x y)
  (values (+ (*  0.85 x) (* 0.04 y))
          (+ (* -0.04 x) (* 0.85 y) 1.6)))

(defun f3 (x y)
  (values (+ (* 0.2  x) (* -0.26 y))
          (+ (* 0.23 x) (*  0.22 y) 1.6)))

(defun f4 (x y)
  (values (+ (* -0.15 x) (* 0.28 y))
          (+ (*  0.26 x) (* 0.24 y) 0.44)))

(defun choose-transform ()
  (let ((r (random 1.0)))
    (cond ((< r 0.01) #'f1)
          ((< r 0.86) #'f2)
          ((< r 0.93) #'f3)
          (t          #'f4))))

(defun set-pixel (image x y)
  (let ((%x (round (+ (* *factor* x) *x-offset*)))
        (%y (round (- *height* (* *factor* y) *y-offset*))))
    (setf (pixel image %y %x) (values 0 255 0))))

(defun fern (filespec &optional (iterations 10000000))
  (let ((image (make-8-bit-rgb-image *height* *width* :initial-element 0))
        (x 0)
        (y 0))
    (dotimes (i iterations)
      (set-pixel image x y)
      (multiple-value-setq (x y) (funcall (choose-transform) x y)))
    (write-png-file filespec image)))
```

Delphi (/wiki/Category:Delphi)

Translation of: Java

Hint: After putting a TPaintBox on the main form align it to alClient. Client width / height of the main form should be no less than 640 x 480.

```
unit Unit1;

interface

uses
  Windows, SysUtils, Graphics, Forms, Controls, Classes, ExtCtrls;

type
  TForm1 = class(TForm)
    PaintBox1: TPaintBox;
    procedure FormPaint(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;

implementation

{$R *.dfm}

procedure CreateFern(const w, h: integer);
var r, x, y: double;
    tmpx, tmpy: double;
    i: integer;
begin
    x := 0;
    y := 0;
    randomize();

    for i := 0 to 200000 do begin
        r := random(100000000) / 99999989;
        if r <= 0.01 then begin
            tmpx := 0;
            tmpy := 0.16 * y;
        end
        else if r <= 0.08 then begin
            tmpx := 0.2 * x - 0.26 * y;
            tmpy := 0.23 * x + 0.22 * y + 1.6;
        end
        else if r <= 0.15 then begin
            tmpx := -0.15 * x + 0.28 * y;
            tmpy := 0.26 * x + 0.24 * y + 0.44;
        end
        else begin
            tmpx := 0.85 * x + 0.04 * y;
            tmpy := -0.04 * x + 0.85 * y + 1.6;
        end;
        x := tmpx;
        y := tmpy;

        Form1.PaintBox1.Canvas.Pixels[round(w / 2 + x * w / 11), round(h - y * h / 
    end;
end;

procedure TForm1.FormPaint(Sender: TObject);
begin
    CreateFern(Form1.ClientWidth, Form1.ClientHeight);
end;

end.
```

EasyLang (/wiki/Category:EasyLang)

Run it (https://easylang.online/apps/barnsley-fern.html)

```
set_color 060
for i range 200000
  r = randomf
  if r < 0.01
    nx = 0
    ny = 0.16 * y
  elif r < 0.08
    nx = 0.2 * x - 0.26 * y
    ny = 0.23 * x + 0.22 * y + 1.6
  elif r < 0.15
    nx = -0.15 * x + 0.28 * y
    ny = 0.26 * x + 0.24 * y + 0.44
  else
    nx = 0.85 * x + 0.04 * y
    ny = -0.04 * x + 0.85 * y + 1.6
  .
  x = nx
  y = ny
  move_pen 50 + x * 15 100 - y * 10
  draw_rect 0.3 0.3
.
```

Forth (/wiki/Category:Forth)

Works with: gforth (/wiki/Gforth) version 0.7.3
Library: SDL2 (/wiki/Category:SDL2)

Fixed Point and Matrix solution

Traditionaly, Forth use Fixed-Point Arithmetic (here with a 1000 scale). For transformation function choice, a formula is used to pick coefficients in a matrix.

```
s" SDL2" add-lib
\c #include <SDL2/SDL.h>
c-function sdl-init             SDL_Init                n -- n
c-function sdl-quit             SDL_Quit                -- void
c-function sdl-createwindow     SDL_CreateWindow        a n n n n n -- a
c-function sdl-createrenderer   SDL_CreateRenderer      a n n -- a
c-function sdl-setdrawcolor     SDL_SetRenderDrawColor  a n n n n -- n
c-function sdl-drawpoint        SDL_RenderDrawPoint     a n n -- n
c-function sdl-renderpresent    SDL_RenderPresent       a -- void
c-function sdl-delay            SDL_Delay               n -- void

require random.fs

0 value window
0 value renderer
variable x
variable y

: initFern ( -- )
  $20 sdl-init drop
  s\" Rosetta Task : Barnsley fern\x0" drop 0 0 1000 1000 $0 sdl-createwindow to wi
  window -1 $2 sdl-createrenderer to renderer
  renderer 0 255 0 255 sdl-setdrawcolor drop
;

create coefficients
             0 ,    0 ,   0 , 160 ,    0 ,    \  1% of the time - f1
           200 , -260 , 230 , 220 , 1600 ,    \  7% of the time - f3
          -150 ,  280 , 260 , 240 ,  440 ,    \  7% of the time - f4
           850 ,   40 , -40 , 850 , 1600 ,    \ 85% of the time - f2

: nextcoeff ( n -- n+1 coeff ) coefficients over cells + @ swap 1+ swap ;
: transformation ( n -- )
  nextcoeff x @ *   swap nextcoeff y @ *    rot + 1000 /   swap
  nextcoeff x @ *   swap nextcoeff y @ *    rot + 1000 /   swap nextcoeff rot +
  x !  \ x shall be modified after y calculation
;
```

```
: randomchoice ( -- index )
  100 random
  dup 0 > swap
  dup 7 > swap
  dup 14 > swap drop
  + + negate 5 *
;

: fern
initFern
20000 0 do
  randomchoice transformation
  renderer x @ 10 / 500 + y @ 10 / sdl-drawpoint drop
loop
renderer sdl-renderpresent
5000 sdl-delay
sdl-quit
;

fern
```

Floating Point and Multiple Functions solution

Forth may use a dedicated Floating Point Stack. For transformation, a pointer to one of the 4 functions is used to be be called at the end of the loop.

```
s" SDL2" add-lib
\c #include <SDL2/SDL.h>
c-function sdl-init             SDL_Init                n -- n
c-function sdl-quit             SDL_Quit                -- void
c-function sdl-createwindow     SDL_CreateWindow        a n n n n n -- a
c-function sdl-createrenderer   SDL_CreateRenderer      a n n -- a
c-function sdl-setdrawcolor     SDL_SetRenderDrawColor  a n n n n -- n
c-function sdl-drawpoint        SDL_RenderDrawPoint     a n n -- n
c-function sdl-renderpresent    SDL_RenderPresent       a -- void
c-function sdl-delay            SDL_Delay               n -- void

require random.fs

0 value window
0 value renderer
0 value diceThrow
fvariable x
fvariable y
variable transformation

: initFern ( -- )
  $20 sdl-init drop
  s\" Rosetta Task : Barnsley fern\x0" drop 0 0 1000 1000 $0 sdl-createwindow to wi
  window -1 $2 sdl-createrenderer to renderer
  renderer 0 255 0 255 sdl-setdrawcolor drop
;
: closeFern sdl-quit ;

: f1
  0e0 x f!
  y f@ 0.16e0 f* y f!
;
: f2
  x f@  0.85e0 f* y f@ 0.040e0 f* f+
  x f@ -0.04e0 f* y f@ 0.850e0 f* f+ 1.600e0 f+   y f!
  x f!
;
```

```
: f3
  x f@ 0.200e0 f* y f@ -0.260e0 f* f+
  x f@ 0.230e0 f* y f@  0.220e0 f* f+ 1.600e0 f+   y f!
  x f!
;
: f4
  x f@ -0.150e0 f* y f@ 0.280e0 f* f+
  x f@  0.260e0 f* y f@ 0.240e0 f* f+  0.440e0 f+    y f!
  x f!
;

: fern
initFern
0e0 x f!
0e0 y f!
20000 0 do
  renderer x f@ 50e0 f* f>s 500 + y f@ 50e0 f* f>s sdl-drawpoint drop
  100 random to diceThrow
  ['] f2 transformation !
  diceThrow 15 < if ['] f4 transformation ! then
  diceThrow 8 < if ['] f3 transformation ! then
  diceThrow 1 < if ['] f1 transformation ! then
  transformation @ execute
loop
  renderer sdl-renderpresent
  5000 sdl-delay
closeFern
;

fern
```

Fortran (/wiki/Category:Fortran)

```
!Generates an output file "plot.dat" that contains the x and y coordinates
!for a scatter plot that can be visualized with say, GNUPlot
program BarnsleyFern
implicit none

double precision :: p(4), a(4), b(4), c(4), d(4), e(4), f(4), trx, try, prob
integer :: itermax, i

!The probabilites and coefficients can be modified to generate other
!fractal ferns, e.g. http://www.home.aone.net.au/~byzantium/ferns/fractal.html
!probabilities
p(1) = 0.01; p(2) = 0.85; p(3) = 0.07; p(4) = 0.07

!coefficients
a(1) =  0.00; a(2) =  0.85; a(3) =  0.20; a(4) = -0.15
b(1) =  0.00; b(2) =  0.04; b(3) = -0.26; b(4) =  0.28
c(1) =  0.00; c(2) = -0.04; c(3) =  0.23; c(4) =  0.26
d(1) =  0.16; d(2) =  0.85; d(3) =  0.22; d(4) =  0.24
e(1) =  0.00; e(2) =  0.00; e(3) =  0.00; e(4) =  0.00
f(1) =  0.00; f(2) =  1.60; f(3) =  1.60; f(4) =  0.44

itermax = 100000

trx = 0.0D0
try = 0.0D0

open(1, file="plot.dat")
write(1,*) "#X            #Y"
write(1,'(2F10.5)') trx, try

do i = 1, itermax
  call random_number(prob)
  if (prob < p(1)) then
    trx = a(1) * trx + b(1) * try + e(1)
    try = c(1) * trx + d(1) * try + f(1)
  else if(prob < (p(1) + p(2))) then
    trx = a(2) * trx + b(2) * try + e(2)
    try = c(2) * trx + d(2) * try + f(2)
  else if ( prob < (p(1) + p(2) + p(3))) then
    trx = a(3) * trx + b(3) * try + e(3)
    try = c(3) * trx + d(3) * try + f(3)
  else
    trx = a(4) * trx + b(4) * try + e(4)
    try = c(4) * trx + d(4) * try + f(4)
  end if
  write(1,'(2F10.5)') trx, try
end do
close(1)
end program BarnsleyFern
```

FreeBASIC (/wiki/Category:FreeBASIC)

```
' version 10-10-2016
' compile with: fbc -s console

Sub barnsley(height As UInteger)

  Dim As Double x, y, xn, yn
  Dim As Double f = height / 10.6
  Dim As UInteger offset_x = height \ 4 - height \ 40
  Dim As UInteger n, r

  ScreenRes height \ 2, height, 32

  For n = 1 To height * 50

    r = Int(Rnd * 100)      ' f from 0 to 99

    Select Case As Const r
      Case 0 To 84
        xn  =  0.85 * x + 0.04 * y
        yn  = -0.04 * x + 0.85 * y + 1.6
      Case 85 To 91
        xn  = 0.2  * x - 0.26 * y
        yn  = 0.23 * x + 0.22 * y + 1.6
      Case 92 To 98
        xn  = -0.15 * x + 0.28 * y
        yn  =  0.26 * x + 0.24 * y + 0.44
      Case Else
        xn = 0
        yn = 0.16 * y
    End Select

    x = xn : y = yn
    PSet( offset_x + x * f, height - y * f), RGB(0, 255, 0)

  Next
' remove comment (') in next line to save window as .bmp file
' BSave "barnsley_fern_" + Str(height) + ".bmp", 0
```

```
End Sub



' ------=< MAIN >=------

' adjustable window height
' call the subroutine with the height you want
' it's possible to have a window that's large than your display
 barnsley(800)



' empty keyboard buffer
While Inkey <> "" : Wend
Windowtitle "hit any key to end program"
Sleep
End
```

Frink (/wiki/Category:Frink)

```
g = new graphics
g.backgroundColor[0,0,0] // black
g.color[0,0.5,0] // green

x = 0
y = 0

for i = 1 to 100000
{
   g.fillEllipseCenter[x*10,y*-10,0.25,0.25]
   z = random[1, 100]
   if z == 1
   {
      xn = 0
      yn = 0.16 * y
   }
   if z >= 2 and z <= 86
   {
      xn = 0.85 * x + 0.04 * y
      yn = -0.04 * x + 0.85 * y + 1.6
   }
   if z >= 87 and z <= 93
   {
      xn = 0.2 * x - 0.26 * y
      yn = 0.23 * x + 0.22 * y + 1.6
   }
   if z >= 94 and z <= 100
   {
      xn = -0.15 * x + 0.28 * y
      yn = 0.26 * x + 0.24 * y + 0.44
   }
   x = xn
   y = yn
}

g.show[]
```

Fōrmulæ (/wiki/Category:F%C5%8Drmul%C3%A6)

In this (https://wiki.formulae.org/Barnsley_fern) page you can see the solution of this task.

Fōrmulæ programs are not textual, visualization/edition of programs is done showing/manipulating structures but not text (more info (http://wiki.formulae.org/Editing_F%C5%8Drmul%C3%A6_expressions)). Moreover, there can be multiple visual representations of the same program. Even though it is possible to have textual representation —i.e. XML, JSON— they are intended for transportation effects more than visualization and edition.

The option to show Fōrmulæ programs and their results is showing images. Unfortunately images cannot be uploaded in Rosetta Code.

G'MIC (/mw/index.php?title=Category:G%27MIC&action=edit&redlink=1)

```
# Put this into a new file 'fern.gmic' and invoke it from the command line, like th
# $ gmic fern.gmic -barnsley_fern

barnsley_fern :
  1024,2048
  -skip {"
      f1 = [ 0,0,0,0.16 ];           g1 = [ 0,0 ];
      f2 = [ 0.2,-0.26,0.23,0.22 ];  g2 = [ 0,1.6 ];
      f3 = [ -0.15,0.28,0.26,0.24 ]; g3 = [ 0,0.44 ];
      f4 = [ 0.85,0.04,-0.04,0.85 ]; g4 = [ 0,1.6 ];
      xy = [ 0,0 ];
      for (n = 0, n<2e6, ++n,
        r = u(100);
        xy = r<=1?((f1**xy)+=g1):
             r<=8?((f2**xy)+=g2):
             r<=15?((f3**xy)+=g3):
                   ((f4**xy)+=g4);
        uv = xy*200 + [ 480,0 ];
        uv[1] = h - uv[1];
        I(uv) = 0.7*I(uv) + 0.3*255;
      )"}
  -r 40%,40%,1,1,2
```

gnuplot (/wiki/Category:Gnuplot)

Translation of: PARI/GP
Works with: gnuplot (/wiki/Gnuplot) version 5.0 (patchlevel 3) and above

File:BarnsleyFernGnu.png (/mw/index.php?title=Special
wpDestFile=BarnsleyFernGr

Output BarnsleyFernGnu.png

```
## Barnsley fern fractal 2/17/17 aev
reset
fn="BarnsleyFernGnu"; clr='"green"';
ttl="Barnsley fern fractal"
dfn=fn.".dat"; ofn=fn.".png";
set terminal (https://www.google.com/search?q=%22set+terminal%22+site%3Ahttp%3A%2F%
set print dfn append
set output (https://www.google.com/search?q=%22set+output%22+site%3Ahttp%3A%2F%2Fww
unset border (https://www.google.com/search?q=%22set+border%22+site%3Ahttp%3A%2F%2F
set size (https://www.google.com/search?q=%22set+size%22+site%3Ahttp%3A%2F%2Fwww.gn
set title (https://www.google.com/search?q=%22set+title%22+site%3Ahttp%3A%2F%2Fwww.
n=100000; max=100; x=y=xw=yw=p=0;
randgp(top) = floor(rand(0)*top)
do for [i=1:n] {
  p=randgp(max);
  if (p==1) {xw=0;yw=0.16*y;}
  if (1<p&&p<=8) {xw=0.2*x-0.26*y;yw=0.23*x+0.22*y+1.6;}
  if (8<p&&p<=15) {xw=-0.15*x+0.28*y;yw=0.26*x+0.24*y+0.44;}
  if (p>15) {xw=0.85*x+0.04*y;yw=-0.04*x+0.85*y+1.6;}
  x=xw;y=yw; print x," ",y;
}
plot dfn using 1:2 with points  pt 7 ps 0.5 lc @clr
set output (https://www.google.com/search?q=%22set+output%22+site%3Ahttp%3A%2F%2Fww
unset print
```

Output:

```
File: BarnsleyFernGnu.png
  (also BarnsleyFernGnu.dat)
```

Go (/wiki/Category:Go)

```
package main

import (
    "image"
    "image/color"
    "image/draw"
    "image/png"
    "log"
    "math/rand"
    "os"
)

// values from WP
const (
    xMin = -2.1820
    xMax = 2.6558
    yMin = 0.
    yMax = 9.9983
)

// parameters
var (
    width = 200
    n     = int(1e6)
    c     = color.RGBA{34, 139, 34, 255} // forest green
)

func main() {
    dx := xMax - xMin
    dy := yMax - yMin
    fw := float64(width)
    fh := fw * dy / dx
    height := int(fh)
    r := image.Rect(0, 0, width, height)
    img := image.NewRGBA(r)
    draw.Draw(img, r, &image.Uniform{color.White}, image.ZP, draw.Src)
    var x, y float64
    plot := func() {
        // transform computed float x, y to integer image coordinates
        ix := int(fw * (x - xMin) / dx)
        iy := int(fh * (yMax - y) / dy)
        img.SetRGBA(ix, iy, c)
    }
    plot()
    for i := 0; i < n; i++ {
        switch s := rand.Intn(100); {
        case s < 85:
            x, y =
                .85*x+.04*y,
                -.04*x+.85*y+1.6
        case s < 85+7:
            x, y =
                .2*x-.26*y,
                .23*x+.22*y+1.6
        case s < 85+7+7:
            x, y =
                -.15*x+.28*y,
                .26*x+.24*y+.44
        default:
            x, y = 0, .16*y
        }
        plot()
    }
    // write img to png file
    f, err := os.Create("bf.png")
    if err != nil {
        log.Fatal(err)
    }
    if err := png.Encode(f, img); err != nil {
        log.Fatal(err)
    }
}
```

Haskell (/wiki/Category:Haskell)

```
import Data.List (scanl (https://haskell.org/ghc/docs/latest/html/libraries/base/Pre
import Diagrams.Backend.Rasterific.CmdLine
import Diagrams.Prelude
import System.Random

type Pt  = (Double, Double)

-- Four affine transformations used to produce a Barnsley fern.
f1, f2, f3, f4 :: Pt -> Pt
f1 (x, y) = (                       0,             0.16 * y)
f2 (x, y) = ( 0.85 * x + 0.04 * y    , -0.04 * x + 0.85 * y + 1.60)
f3 (x, y) = ( 0.20 * x - 0.26 * y    ,  0.23 * x + 0.22 * y + 1.60)
f4 (x, y) = (-0.15 * x + 0.28 * y    ,  0.26 * x + 0.24 * y + 0.44)

-- Given a random number in [0, 1) transform an initial point by a randomly
-- chosen function.
func :: Pt -> Double -> Pt
func p r | r < 0.01  = f1 p
         | r < 0.86  = f2 p
         | r < 0.93  = f3 p
         | otherwise = f4 p

-- Using a sequence of uniformly distributed random numbers in [0, 1) return
-- the same number of points in the fern.
fern :: [Double] -> [Pt]
fern = scanl' func (0, 0)

-- Given a supply of random values and a count, generate a diagram of a fern
-- composed of that number of points.
drawFern :: [Double (https://haskell.org/ghc/docs/latest/html/libraries/base/Prelude
drawFern rs n = frame 0.5 . diagramFrom . take (https://haskell.org/ghc/docs/latest/
  where diagramFrom = flip (https://haskell.org/ghc/docs/latest/html/libraries/base/
        dot = circle 0.005 # lc green

-- To generate a PNG image of a fern, call this program like:
--
--   fern -o fern.png -w 640 -h 640 50000
--
```

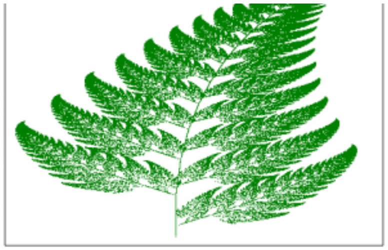
```
-- where the arguments specify the width, height and number of points in the
-- image.
main :: IO (https://haskell.org/ghc/docs/latest/html/libraries/base/Prelude.html#t:
main = do
  rand <- getStdGen
  mainWith $ drawFern (randomRs (0, 1) rand)
```

IS-BASIC (/wiki/Category:IS-BASIC)

```
100 PROGRAM "Fern.bas"
110 RANDOMIZE
120 SET VIDEO MODE 1:SET VIDEO COLOR 0:SET VIDEO X 40:SET VIDEO Y 27
130 OPEN #101:"video:"
140 DISPLAY #101:AT 1 FROM 1 TO 27
150 SET PALETTE BLACK,GREEN
160 LET MX=16000:LET X,Y=0
170 FOR N=1 TO MX
180   LET P=RND(100)
190   SELECT CASE P
200   CASE IS<=1
210     LET NX=0:LET NY=.16*Y
220   CASE IS<=8
230     LET NX=.2*X-.26*Y:LET NY=.23*X+.22*Y+1.6
240   CASE IS<=15
250     LET NX=-.15*X+.28*Y:LET NY=.26*X+.24*Y+.44
260   CASE ELSE
270     LET NX=.85*X+.04*Y:LET NY=-.04*X+.85*Y+1.6
280   END SELECT
290   LET X=NX:LET Y=NY
300   PLOT X*96+600,Y*96
310 NEXT
```

J (/wiki/Category:J)

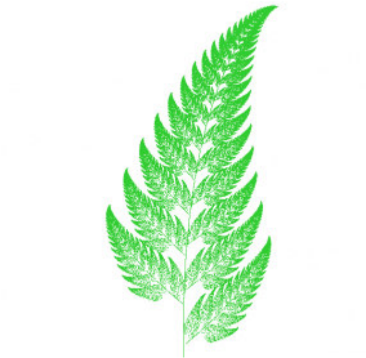
```
require 'plot'

f=: |: 0 ". ];._2 noun define
  0     0     0    0.16   0 0      0.01
  0.85 -0.04  0.04 0.85   0 1.60   0.85
  0.20  0.23 -0.26 0.22   0 1.60   0.07
 -0.15  0.26  0.28 0.24   0 0.44   0.07
)

fm=: {&(|: 2 2 $ f)
fa=: {&(|: 4 5 { f)
prob=: (+/\ 6 { f) I. ?@0:

ifs=: (fa@] + fm@] +/ .* [) prob
getPoints=: ifs^:(<200000)
plotFern=: 'dot;grids 0 0;tics 0 0;labels 0 0;color gre

   plotFern getPoints 0 0
```



(/wiki/File:Jfern.png)

Java (/wiki/Category:Java)

Works with: Java (/wiki/Java) version 8



(/wiki
/File:Barnsley_fern_java.png)

```
import java.awt.*;
import java.awt.image.BufferedImage;
import javax.swing.*;

public class BarnsleyFern extends JPanel (https://www.google.com/search?hl=en&q=alli

    BufferedImage (https://www.google.com/search?hl=en&q=allinurl%3Abufferedimage+ja

    public BarnsleyFern() {
        final int dim = 640;
        setPreferredSize(new Dimension (https://www.google.com/search?hl=en&q=allinu
        setBackground(Color (https://www.google.com/search?hl=en&q=allinurl%3Acolor
        img = new BufferedImage (https://www.google.com/search?hl=en&q=allinurl%3Abu
        createFern(dim, dim);
    }

    void createFern(int w, int h) {
        double x = 0;
        double y = 0;

        for (int i = 0; i < 200_000; i++) {
            double tmpx, tmpy;
            double r = Math (https://www.google.com/search?hl=en&q=allinurl%3Amath+j

            if (r <= 0.01) {
                tmpx = 0;
                tmpy = 0.16 * y;
            } else if (r <= 0.08) {
                tmpx = 0.2 * x - 0.26 * y;
                tmpy = 0.23 * x + 0.22 * y + 1.6;
            } else if (r <= 0.15) {
                tmpx = -0.15 * x + 0.28 * y;
                tmpy = 0.26 * x + 0.24 * y + 0.44;
            } else {
                tmpx = 0.85 * x + 0.04 * y;
                tmpy = -0.04 * x + 0.85 * y + 1.6;
            }
            x = tmpx;
            y = tmpy;

            img.setRGB((int) Math (https://www.google.com/search?hl=en&q=allinurl%3A
                    (int) Math (https://www.google.com/search?hl=en&q=allinurl%3Ama
        }
    }

    @Override
    public void paintComponent(Graphics (https://www.google.com/search?hl=en&q=alli
        super.paintComponent(gg);
        Graphics2D (https://www.google.com/search?hl=en&q=allinurl%3Agraphics2d+java
        g.setRenderingHint(RenderingHints (https://www.google.com/search?hl=en&q=al
                RenderingHints (https://www.google.com/search?hl=en&q=allinurl%3Are

        g.drawImage(img, 0, 0, null);
    }

    public static void main(String (https://www.google.com/search?hl=en&q=allinurl%
        SwingUtilities (https://www.google.com/search?hl=en&q=allinurl%3Aswingutili
            JFrame (https://www.google.com/search?hl=en&q=allinurl%3Ajframe+java.su
            f.setDefaultCloseOperation(JFrame (https://www.google.com/search?hl=en&
            f.setTitle("Barnsley Fern");
            f.setResizable(false);
            f.add(new BarnsleyFern(), BorderLayout (https://www.google.com/search?hl
            f.pack();
            f.setLocationRelativeTo(null);
            f.setVisible(true);
        });
    }
}
```

JavaScript (/wiki/Category:JavaScript)

Translation of: PARI/GP

File:BarnsleyFernjs.png
(/mw/index.php?title=Special

```
// Barnsley fern fractal
//6/17/16 aev
function pBarnsleyFern(canvasId, lim) {
    // DCLs
    var canvas = document.getElementById(canvasId);
    var ctx = canvas.getContext("2d");
    var w = canvas.width;
    var h = canvas.height;
    var x = 0.,
        y = 0.,
        xw = 0.,
        yw = 0.,
        r;
    // Like in PARI/GP: return random number 0..max-1
    function randgp(max) {
        return Math.floor(Math.random() * max)
    }
    // Clean canvas
    ctx.fillStyle = "white";
    ctx.fillRect(0, 0, w, h);
    // MAIN LOOP
    for (var i = 0; i < lim; i++) {
        r = randgp(100);
        if (r <= 1) {
            xw = 0;
            yw = 0.16 * y;
        } else if (r <= 8) {
            xw = 0.2 * x - 0.26 * y;
            yw = 0.23 * x + 0.22 * y + 1.6;
        } else if (r <= 15) {
            xw = -0.15 * x + 0.28 * y;
            yw = 0.26 * x + 0.24 * y + 0.44;
        } else {
            xw = 0.85 * x + 0.04 * y;
            yw = -0.04 * x + 0.85 * y + 1.6;
        }
        x = xw;
        y = yw;
```

wpDestFile=BarnsleyFernjs.

Output BarnsleyFernjs.png

```
        ctx.fillStyle = "green";
        ctx.fillRect(x * 50 + 260, -y * 50 + 540, 1, 1);
    } //fend i
}
```

Executing:

```
<html>
 <head><script src="BarnsleyFern.js"></script></head>
 <body onload="pBarnsleyFern('canvas', 100000)">
   <br /> <h3>Barnsley fern fractal</h3>
   <canvas id="canvas" width="540" height="540" style="border: 2px inset;"></canvas
 </body>
</html>
```

Output:

```
Page with BarnsleyFernjs.png
```

Julia (/wiki/Category:Julia)

Works with: Julia (/wiki/Julia) version 0.6

```
function barnsleyfern(n::Integer)
    funs = (
        (x, y) -> (0, 0.16y),
        (x, y) -> (0.85x + 0.04y, -0.04x + 0.85y + 1.6),
        (x, y) -> (0.2x - 0.26y, 0.23x + 0.22y + 1.6),
        (x, y) -> (-0.15x + 0,28y, 0.26x + 0.24y + 0.44))
    rst = Matrix{Float64}(n, 2)
    rst[1, :] = 0.0
    for row in 2:n
        r = rand(0:99)
        if r < 1;      f = 1;
        elseif r < 86; f = 2;
        elseif r < 93; f = 3;
        else           f = 4; end
        rst[row, 1], rst[row, 2] = funs[f](rst[row-1, 1], rst[row-1, 2])
    end
    return rst
end
```

Kotlin (/wiki/Category:Kotlin)

Translation of: Java

```
// version 1.1.0

import (https://scala-lang.org) java.awt.*
import (https://scala-lang.org) java.awt.image.BufferedImage
import (https://scala-lang.org) javax.swing.*

class (https://scala-lang.org) BarnsleyFern(private (https://scala-lang.org) val (h
    private (https://scala-lang.org) val (https://scala-lang.org) img: BufferedImag

    init {
        preferredSize = Dimension(dim, dim)
        background = Color.black
        img = BufferedImage(dim, dim, BufferedImage.TYPE_INT_ARGB)
        createFern(dim, dim)
    }

    private (https://scala-lang.org) fun createFern(w: Int, h: Int) {
        var (https://scala-lang.org) x = 0.0
        var (https://scala-lang.org) y = 0.0
        for (https://scala-lang.org) (i in 0 until 200_000) {
            var (https://scala-lang.org) tmpx: Double
            var (https://scala-lang.org) tmpy: Double
            val (https://scala-lang.org) r = Math.random()
            if (https://scala-lang.org) (r <= 0.01) {
                tmpx = 0.0
                tmpy = 0.16 * y
            }
            else (https://scala-lang.org) if (https://scala-lang.org) (r <= 0.86) {
                tmpx =  0.85 * x + 0.04 * y
                tmpy = -0.04 * x + 0.85 * y + 1.6
            }
            else (https://scala-lang.org) if (https://scala-lang.org) (r <= 0.93) {
                tmpx = 0.2  * x - 0.26 * y
                tmpy = 0.23 * x + 0.22 * y + 1.6
            }
            else (https://scala-lang.org) {
                tmpx = -0.15 * x + 0.28 * y
                tmpy =  0.26 * x + 0.24 * y + 0.44
            }
            x = tmpx
            y = tmpy
            img.setRGB(Math.round(w / 2.0 + x * w / 11.0).toInt(),
                       Math.round(h - y * h / 11.0).toInt(), 0xFF32CD32.toInt())
        }
    }

    override (https://scala-lang.org) protected (https://scala-lang.org) fun paintC
        super (https://scala-lang.org).paintComponent(gg)
        val (https://scala-lang.org) g = gg as Graphics2D
        g.setRenderingHint(RenderingHints.KEY_ANTIALIASING, RenderingHints.VALUE_AN
        g.drawImage(img, 0, 0, null (https://scala-lang.org))
    }
}

fun main(args: Array<String>) {
    SwingUtilities.invokeLater {
        val (https://scala-lang.org) f = JFrame()
        f.defaultCloseOperation = JFrame.EXIT_ON_CLOSE
        f.title = "Barnsley Fern"
        f.setResizable(false (https://scala-lang.org))
        f.add(BarnsleyFern(640), BorderLayout.CENTER)
        f.pack()
        f.setLocationRelativeTo(null (https://scala-lang.org))
        f.setVisible(true (https://scala-lang.org))
    }
}
```

Lambdatalk (/wiki/Category:Lambdatalk)

```
{def fern
 {lambda {:size :sign}
  {if {> :size 2}
   then M:size
        T{* 70 :sign}
          {fern {* :size 0.5} {- :sign}}
        T{* {- 70} :sign}
        M:size
        T{* {- 70} :sign}
          {fern {* :size 0.5} :sign}
        T{* 70 :sign}
        T{* 7 :sign}
          {fern {- :size 1} :sign}
        T{* {- 7} :sign}
        M{* -:size 2}
   else }}}

{def F {fern 25 1}}
```

The output can be seen in http://lambdaway.free.fr/lambdawalks/?view=fern (http://lambdaway.free.fr/lambdawalks/?view=fern)

Liberty BASIC (/wiki/Category:Liberty_BASIC)

```
nomainwin
WindowWidth=800
WindowHeight=600
open "Barnsley Fern" for graphics_nf_nsb as #1
#1 "trapclose [q];down;fill black;flush;color green"

for n = 1 To WindowHeight * 50
    r = int(rnd(1)*100)
    Select Case
      Case (r>=0) and (r<=84)
        xn=0.85*x+0.04*y
        yn=-0.04*x+0.85*y+1.6
      Case (r>84) and (r<=91)
        xn=0.2*x-0.26*y
        yn=0.23*x+0.22*y+1.6
      Case (r>91) and (r<=98)
        xn=-0.15*x+0.28*y
        yn=0.26*x+0.24*y+0.44
      Case Else
        xn=0
        yn=0.16*y
    End Select
    x=xn
    y = yn
    #1 "set ";x*80+300;" ";WindowHeight/1.1-y*50
  next n
  #1 "flush"
  wait
[q]
close #1
```

Locomotive Basic (/wiki/Category:Locomotive_Basic)

Translation of: ZX Spectrum Basic

```
10 mode 2:ink 0,0:ink 1,18:randomize time
20 scale=38
30 maxpoints=20000: x=0: y=0
40 for z=1 to maxpoints
50 p=rnd*100
60 if p<=1 then nx=0: ny=0.16*y: goto 100
70 if p<=8 then nx=0.2*x-0.26*y: ny=0.23*x+0.22*y+1.6: goto 100
80 if p<=15 then nx=-0.15*x+0.28*y: ny=0.26*x+0.24*y+0.44: goto 100
90 nx=0.85*x+0.04*y: ny=-0.04*x+0.85*y+1.6
100 x=nx: y=ny
110 plot scale*x+320,scale*y
120 next
```

Lua (/wiki/Category:Lua)

Needs LÖVE 2D Engine

```
g = love.graphics
wid, hei = g.getWidth(), g.getHeight()

function choose( i, j )
  local r = math.random()
  if r < .01 then return 0, .16 * j
    elseif r < .07 then return .2 * i - .26 * j, .23 * i + .22 * j + 1.6
    elseif r < .14 then return -.15 * i + .28 * j, .26 * i + .24 * j + .44
    else return .85 * i + .04 * j, -.04 * i + .85 * j + 1.6
  end
end
function createFern( iterations )
  local hw, x, y, scale = wid / 2, 0, 0, 45
  local pts = {}
  for k = 1, iterations do
    pts[1] = { hw + x * scale, hei - 15 - y * scale,
               20 + math.random( 80 ),
               128 + math.random( 128 ),
               20 + math.random( 80 ), 150 }
    g.points( pts )
    x, y = choose( x, y )
  end
end
function love.load()
  math.randomseed( os.time() )
  canvas = g.newCanvas( wid, hei )
  g.setCanvas( canvas )
  createFern( 15e4 )
  g.setCanvas()
end
function love.draw()
  g.draw( canvas )
end
```

Mathematica (/wiki/Category:Mathematica) / Wolfram Language (/wiki /Category:Wolfram_Language)

```
BarnsleyFern[{x_, y_}] := Module[{},
   i = RandomInteger[{1, 100}];
   If[i <= 1, {xt = 0, yt = 0.16*y},
    If[i <= 8, {xt = 0.2*x - 0.26*y, yt = 0.23*x + 0.22*y + 1.6},
     If[i <= 15, {xt = -0.15*x + 0.28*y, yt = 0.26*x + 0.24*y + 0.44},
      {xt = 0.85*x + 0.04*y, yt = -0.04*x + 0.85*y + 1.6}]]];
   {xt, yt}];
points = NestList[BarnsleyFern, {0,0}, 100000];
Show[Graphics[{Hue[.35, 1, .7], PointSize[.001], Point[#] & /@ points}]]
```

MiniScript (/wiki/Category:MiniScript)

Translation of: C#
Works with: Mini Micro (/wiki/Mini_Micro)

```
clear
x = 0
y = 0
for i in range(100000)
        gfx.setPixel 300 + 58 * x, 58 * y, color.green
        roll = rnd * 100
        xp = x
        if roll < 1 then
                x = 0
                y = 0.16 * y
        else if roll < 86 then
                x = 0.85 * x + 0.04 * y
                y = -0.04 * xp + 0.85 * y + 1.6
        else if roll < 93 then
                x = 0.2 * x - 0.26 * y
                y = 0.23 * xp + 0.22 * y + 1.6
        else
                x = -0.15 * x + 0.28 * y
                y = 0.26 * xp + 0.24 * y + 0.44
        end if
end for
```

Nim (/wiki/Category:Nim)

```
import nimPNG, random

randomize()

const
  width = 640
  height = 640
  minX = -2.1815
  maxX = 2.6556
  minY = 0.0
  maxY = 9.9982
  iterations = 1_000_000

var img: array[width * height * 3, char]

proc floatToPixel(x,y:float): tuple[a:int,b:int] =
  var px = abs(x - minX) / abs(maxX - minX)
  var py = abs(y - minY) / abs(maxY - minY)

  var a:int = (int)(width * px)
  var b:int = (int)(height * py)

  a = a.clamp(0, width-1)
  b = b.clamp(0, height-1)
  # flip the y axis
  (a:a,b:height-b-1)

proc pixelToOffset(a,b: int): int =
  b * width * 3 + a * 3

proc toString(a: openArray[char]): string =
  result = newStringOfCap(a.len)

  for ch in items(a):
    result.add(ch)

proc drawPixel(x,y:float) =
  var (a,b) = floatToPixel(x,y)
  var offset = pixelToOffset(a,b)

  #img[offset] = 0 # red channel
  img[offset+1] = char(250) # green channel
  #img[offset+2] = 0 # blue channel

# main
var x, y: float = 0.0

for i in 1..iterations:
  var r = random(101)
  var nx, ny: float
  if r <= 85:
    nx = 0.85 * x + 0.04 * y
    ny = -0.04 * x + 0.85 * y + 1.6
  elif r <= 85 + 7:
    nx = 0.2 * x - 0.26 * y
    ny = 0.23 * x + 0.22 * y + 1.6
  elif r <= 85 + 7 + 7:
    nx = -0.15 * x + 0.28 * y
    ny = 0.26 * x + 0.24 * y + 0.44
  else:
    nx = 0
    ny = 0.16 * y

  x = nx
  y = ny

  drawPixel(x,y)

discard savePNG24("fern.png",img.toString, width, height)
```

Oberon-2 (/wiki/Category:Oberon-2)

```
MODULE BarnsleyFern;
(**
        Oxford Oberon-2
**)

        IMPORT Random, XYplane;

        VAR
                a1, b1, c1, d1, e1, f1, p1: REAL;
                a2, b2, c2, d2, e2, f2, p2: REAL;
                a3, b3, c3, d3, e3, f3, p3: REAL;
                a4, b4, c4, d4, e4, f4, p4: REAL;
                X, Y: REAL;
                x0, y0, e: INTEGER;

        PROCEDURE Draw;
                VAR x, y: REAL; xi, eta: INTEGER; rn: REAL
        BEGIN
                REPEAT
                        rn := Random.Uniform();
                        IF rn < p1 THEN
                                x := a1 * X + b1 * Y + e1;
                        ELSIF rn < (p1 + p2) THEN
                                x := a2 *X + b2 * Y + e2;
                        ELSIF rn < (p1 + p2 + p3) THEN
                                x := a3 * X + b3 * Y + e3;
                        ELSE
                                x := a4 * X + b4 * Y + e4;
                        END;
                        X := x; xi := x0 + SHORT(ENTIER(X
                        Y := y; eta := y0 + SHORT(ENTIER(Y
                        XYplane.Dot(xi, eta, XYplane.draw)
                UNTIL "s" = XYplane.Key()
        END Draw;

        PROCEDURE Init;
        BEGIN
```

File:Barnsleyfern-oberon2.png (/mw/index.php?title=Special wpDestFile=Barnsleyfern-oberon2.png)

250px

```
                X := 0; Y := 0;
                x0 := 120; y0 := 0; e := 25;

                a1 := 0.00; a2 :=  0.85; a3 :=  0.20; a4 := -0.15;
                b1 := 0.00; b2 :=  0.04; b3 := -0.26; b4 :=  0.28;
                c1 := 0.00; c2 := -0.04; c3 :=  0.23; c4 :=  0.26;
                d1 := 0.16; d2 :=  0.85; d3 :=  0.22; d4 :=      0.24;
                e1 := 0.00; e2 :=  0.00; e3 :=  0.00; e4 :=  0.00;
                f1 := 0.00; f2 :=  1.60; f3 :=  1.60; f4 :=  0.44;
                p1 := 0.01; p2 :=  0.85; p3 :=  0.07; p4 :=      0.07;
                XYplane.Open;
        END Init;

BEGIN
        Init;Draw
END BarnsleyFern.
```

PARI/GP (/wiki/Category:PARI/GP)

Translation of: zkl

Works with: PARI/GP (/wiki/PARI/GP) version 2.7.4 and above

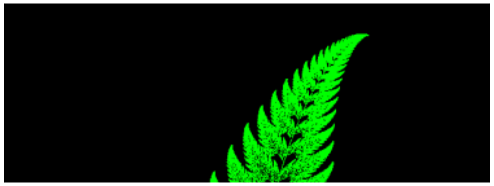
File:BarnsleyFern.png (/mw/index.php?title=Special wpDestFile=BarnsleyFern.pr
Output BarnsleyFern.png

```
\\ Barnsley fern fractal
\\ 6/17/16 aev
pBarnsleyFern(size,lim)={
my(X=List(),Y=X,x=y=xw=yw=0.0,r);
print(" *** Barnsley Fern, size=",size," lim=",lim);
plotinit(0); plotcolor(0,6); \\green
plotscale(0, -3,3, 0,10); plotmove(0, 0,0);
for(i=1, lim,
  r=random(100);
  if(r<=1, xw=0;yw=0.16*y,
    if(r<=8, xw=0.2*x-0.26*y;yw=0.23*x+0.22*y+1.6,
      if(r<=15, xw=-0.15*x+0.28*y;yw=0.26*x+0.24*y+0.44,
        xw=0.85*x+0.04*y;yw=-0.04*x+0.85*y+1.6)));
  x=xw;y=yw; listput(X,x); listput(Y,y);
);\\fend i
plotpoints(0,Vec(X),Vec(Y));
plotdraw([0,-3,-0]);
}
{\\ Executing:
pBarnsleyFern(530,100000);  \\ BarnsleyFern.png
}
```

Output:

```
> pBarnsleyFern(530,100000);  \\ BarnsleyFern.png
 *** Barnsley Fern, size=530 lim=100000
```

Perl (/wiki/Category:Perl)



```
use Imager;

my $w = 640;
my $h = 640;

my $img = Imager->new(xsize => $w, ysize => $h, c
my $green = Imager::Color->new('#00FF00');

my ($x, $y) = (0, 0);

foreach (1 .. 2e5) {
  my $r = rand (https://perldoc.perl.org/functior
  ($x, $y) = do {
    if    ($r <=  1) { ( 0.00 * $x - 0.00 * $y,
    elsif ($r <=  8) { ( 0.20 * $x - 0.26 * $y,
    elsif ($r <= 15) { (-0.15 * $x + 0.28 * $y,
    else             { ( 0.85 * $x + 0.04 * $y, -
  };
  $img->setpixel(x => $w / 2 + $x * 60, y (https:
}

$img->flip(dir => 'v');
$img->write (https://perldoc.perl.org/functions/v
```

(/wiki/File:BarnsleyFernPerl.png)

Phix (/wiki/Category:Phix)

Library: Phix/pGUI (/wiki/Category:Phix/pGUI)
output: on imgur (https://imgur.com/a/04ZZZt9)

```
-- demo\rosetta\BarnsleyFern.exw
include pGUI.e

Ihandle dlg, canvas
cdCanvas cddbuffer, cdcanvas

function redraw_cb(Ihandle /*ih*/, integer /*posx*/, integer /*posy*/)
atom {x,y,r} @= 0
integer {width, height} = IupGetIntInt(canvas, "DRAWSIZE")
    cdCanvasActivate(cddbuffer)
    for i=1 to 20000 do
        r = rand(100)
        {x, y} = iff(r<=1? {             0,        0.16*y     } :
                 iff(r<=8? { 0.20*x-0.26*y, 0.23*x+0.22*y+1.60} :
                 iff(r<=15?{-0.15*x+0.28*y, 0.26*x+0.24*y+0.44} :
                           { 0.85*x+0.04*y,-0.04*x+0.85*y+1.60})))
        cdCanvasPixel(cddbuffer, width/2+x*60, y*60, #00FF00)
    end for
    cdCanvasFlush(cddbuffer)
    return IUP_DEFAULT
end function

function map_cb(Ihandle ih)
    cdcanvas = cdCreateCanvas(CD_IUP, ih)
    cddbuffer = cdCreateCanvas(CD_DBUFFER, cdcanvas)
    cdCanvasSetBackground(cddbuffer, CD_WHITE)
    cdCanvasSetForeground(cddbuffer, CD_RED)
    return IUP_DEFAULT
end function

function esc_close(Ihandle /*ih*/, atom c)
    if c=K_ESC then return IUP_CLOSE end if
    return IUP_CONTINUE
end function

procedure main()
    IupOpen()
    canvas = IupCanvas(NULL)
    IupSetAttribute(canvas, "RASTERSIZE", "340x620") -- initial size
    IupSetCallback(canvas, "MAP_CB", Icallback("map_cb"))

    dlg = IupDialog(canvas)
    IupSetAttribute(dlg, "TITLE", "Barnsley Fern")
    IupSetCallback(dlg, "K_ANY",     Icallback("esc_close"))
    IupSetCallback(canvas, "ACTION", Icallback("redraw_cb"))

    IupMap(dlg)
    IupSetAttribute(canvas, "RASTERSIZE", NULL) -- release the minimum limitation
    IupShowXY(dlg,IUP_CENTER,IUP_CENTER)
    IupMainLoop()
    IupClose()
end procedure

main()
```

PicoLisp (/wiki/Category:PicoLisp)

```
`(== 64 64)
(seed (in "/dev/urandom" (rd 8)))
(scl 20)
(de gridX (X)
   (*/ (+ 320.0 (*/ X 58.18 1.0)) 1.0) )
(de gridY (Y)
   (*/ (- 640.0 (*/ Y 58.18 1.0)) 1.0) )
(de calc (R X Y)
   (cond
      ((< R 1) (list 0 (*/ Y 0.16 1.0)))
      ((< R 86)
         (list
            (+ (*/ 0.85 X 1.0) (*/ 0.04 Y 1.0))
            (+ (*/ -0.04 X 1.0) (*/ 0.85 Y 1.0) 1.6) ) )
      ((< R 93)
         (list
            (- (*/ 0.2 X 1.0) (*/ 0.26 Y 1.0))
            (+ (*/ 0.23 X 1.0) (*/ 0.22 Y 1.0) 1.6) ) )
      (T
         (list
            (+ (*/ -0.15 X 1.0) (*/ 0.28 Y 1.0))
            (+ (*/ 0.26 X 1.0) (*/ 0.24 Y 1.0) 0.44) ) ) ) )
(let
   (X 0
      Y 0
      G (make (do 640 (link (need 640 0)))) )
   (do 100000
      (let ((A B) (calc (rand 0 99) X Y))
         (setq X A  Y B)
         (set (nth G (gridY Y) (gridX X)) 1) ) )
   (out "fern.pbm"
      (prinl "P1")
      (prinl 640 " " 640)
      (mapc prinl G) ) )
```

Processing (/wiki/Category:Processing)

```
void setup() {
  size(640, 640);
  background(0, 0, 0);
}

float x = 0;
float y = 0;

void draw() {
  for (int i = 0; i < 100000; i++) {

    float xt = 0;
    float yt = 0;

    float r = random(100);

    if (r <= 1) {
      xt = 0;
      yt = 0.16*y;
    } else if (r <= 8) {
      xt = 0.20*x - 0.26*y;
      yt = 0.23*x + 0.22*y + 1.60;
    } else if (r <= 15) {
      xt = -0.15*x + 0.28*y;
      yt =  0.26*x + 0.24*y + 0.44;
    } else {
      xt =  0.85*x + 0.04*y;
      yt = -0.04*x + 0.85*y + 1.60;
    }

    x = xt;
    y = yt;

    int m = round(width/2 + 60*x);
    int n = height-round(60*y);

    set(m, n, #00ff00);
  }
  noLoop();
}
```

Processing Python mode (/wiki/Category:Processing_Python_mode)

```
size(640, 640)
background(0)

x = 0
y = 0

for _ in range(100000):
    xt = 0
    yt = 0
    r = random(100)

    if r <= 1:
        xt = 0
        yt = 0.16 * y
    elif r <= 8:
        xt = 0.20 * x - 0.26 * y
        yt = 0.23 * x + 0.22 * y + 1.60
    elif r <= 15:
        xt = -0.15 * x + 0.28 * y
        yt = +0.26 * x + 0.24 * y + 0.44
    else:
        xt = +0.85 * x + 0.04 * y
        yt = -0.04 * x + 0.85 * y + 1.60
size(640, 640)
background(0)

x = 0
y = 0

for _ in range(100000):
    xt = 0
    yt = 0
    r = random(100)

    if r <= 1:
        xt = 0
        yt = 0.16*y
    elif r <= 8:
        xt = 0.20*x - 0.26*y
        yt = 0.23*x + 0.22*y + 1.60
    elif r <= 15:
        xt = -0.15*x + 0.28*y
        yt =    0.26*x + 0.24*y + 0.44
    else:
        xt =    0.85*x + 0.04*y
        yt = -0.04*x + 0.85*y + 1.60

    x = xt
    y = yt

    m = round(width/2 + 60*x)
    n = height-round(60*y)

    set(m, n, "#00ff00")
    x = xt
    y = yt

    m = round(width / 2 + 60 * x)
    n = height - round(60 * y)

    set(m, n, "#00ff00")
```

PureBasic (/wiki/Category:PureBasic)

```
EnableExplicit
DisableDebugger

DataSection
  R84:  : Data.d 0.85,0.04,-0.04,0.85,1.6
  R91:  : Data.d 0.2,-0.26,0.23,0.22,1.6
  R98:  : Data.d -0.15,0.28,0.26,0.24,0.44
  R100: : Data.d 0.0,0.0,0.0,0.16,0.0
EndDataSection

Procedure Barnsley(height.i)
  Define x.d, y.d, xn.d, yn.d, v1.d, v2.d, v3.d, v4.d, v5.d,
         f.d=height/10.6,
         offset.i=Int(height/4-height/40),
         n.i, r.i
  For n=1 To height*50
    r=Random(99,0)
    Select r
      Case 0 To 84  : Restore R84
      Case 85 To 91 : Restore R91
      Case 92 To 98 : Restore R98
      Default       : Restore R100
    EndSelect
    Read.d v1 : Read.d v2 : Read.d v3 : Read.d v4 : Read.d v5
    xn=v1*x+v2*y : yn=v3*x+v4*y+v5
    x=xn : y=yn
    Plot(offset+x*f,height-y*f,RGB(0,255,0))
  Next
EndProcedure

Define w1.i=400,
       h1.i=800

If OpenWindow(0,#PB_Ignore,#PB_Ignore,w1,h1,"Barnsley fern")
  If CreateImage(0,w1,h1,24,0) And StartDrawing(ImageOutput(0))
    Barnsley(h1)
    StopDrawing()
  EndIf
  ImageGadget(0,0,0,0,0,ImageID(0))
  Repeat : Until WaitWindowEvent(50)=#PB_Event_CloseWindow
EndIf
End
```

Python (/wiki/Category:Python)

```
import random
from PIL import Image


class BarnsleyFern(object):
    def __init__(self, img_width, img_height, paint_color=(0, 150, 0),
                 bg_color=(255, 255, 255)):
        self.img_width, self.img_height = img_width, img_height
        self.paint_color = paint_color
        self.x, self.y = 0, 0
        self.age = 0

        self.fern = Image.new('RGB', (img_width, img_height), bg_color)
        self.pix = self.fern.load()
        self.pix[self.scale(0, 0)] = paint_color

    def scale(self, x, y):
        h = (x + 2.182)*(self.img_width - 1)/4.8378
        k = (9.9983 - y)*(self.img_height - 1)/9.9983
        return h, k

    def transform(self, x, y):
        rand = random.uniform(0, 100)
        if rand < 1:
            return 0, 0.16*y
        elif 1 <= rand < 86:
            return 0.85*x + 0.04*y, -0.04*x + 0.85*y + 1.6
        elif 86 <= rand < 93:
            return 0.2*x - 0.26*y, 0.23*x + 0.22*y + 1.6
        else:
            return -0.15*x + 0.28*y, 0.26*x + 0.24*y + 0.44

    def iterate(self, iterations):
        for _ in range(iterations):
            self.x, self.y = self.transform(self.x, self.y)
            self.pix[self.scale(self.x, self.y)] = self.paint_color
        self.age += iterations

fern = BarnsleyFern(500, 500)
fern.iterate(1000000)
fern.fern.show()
```

R (/wiki/Category:R)

Matrix solution

Translation of: PARI/GP

File:BarnsleyFernR.png
(/mw/index.php?title=Special
wpDestFile=BarnsleyFernR.p

Output BarnsleyFernR.png

```
## pBarnsleyFern(fn, n, clr, ttl, psz=600): Plot Barnsley fern fractal.
## Where: fn - file name; n - number of dots; clr - color; ttl - plot title;
## psz - picture size.
## 7/27/16 aev
pBarnsleyFern <- function(fn, n, clr, ttl, psz=600) {
  cat(" *** START:", date(), "n=", n, "clr=", clr, "psz=", psz, "\n");
  cat(" *** File name -", fn, "\n");
  pf = paste0(fn,".png"); # pf - plot file name
  A1 <- matrix(c(0,0,0,0.16,0.85,-0.04,0.04,0.85,0.2,0.23,-0.26,0.22,-0.15,0.26,0.28
  A2 <- matrix(c(0,0,0,1.6,0,1.6,0,0.44), ncol=2, nrow=4, byrow=TRUE);
  P <- c(.01,.85,.07,.07);
  # Creating matrices M1 and M2.
  M1=vector("list", 4); M2 = vector("list", 4);
  for (i in 1:4) {
    M1[[i]] <- matrix(c(A1[i,1:4]), nrow=2);
    M2[[i]] <- matrix(c(A2[i, 1:2]), nrow=2);
  }
  x <- numeric(n); y <- numeric(n);
  x[1] <- y[1] <- 0;
  for (i in 1:(n-1)) {
    k <- sample(1:4, prob=P, size=1);
    M <- as.matrix(M1[[k]]);
    z <- M%*%c(x[i],y[i]) + M2[[k]];
    x[i+1] <- z[1]; y[i+1] <- z[2];
  }
  plot(x, y, main=ttl, axes=FALSE, xlab="", ylab="", col=clr, cex=0.1);
  # Writing png-file
  dev.copy(png, filename=pf,width=psz,height=psz);
  # Cleaning
  dev.off(); graphics.off();
  cat(" *** END:",date(),"\n");
}
## Executing:
pBarnsleyFern("BarnsleyFernR", 100000, "dark green", "Barnsley Fern Fractal", psz=6
```

Output:

```
> pBarnsleyFern("BarnsleyFernR", 100000, "dark green", "Barnsley Fern Fractal", psz=
 *** START: Wed Jul 27 13:50:49 2016 n= 1e+05 clr= dark green psz= 600
 *** File name - BarnsleyFernR
 *** END: Wed Jul 27 13:50:56 2016
+ BarnsleyFernR.png file
```

'Obvious' solution

The matrix solution above is a clever approach, but the following solution is more readable if you're unfamiliar with linear algebra. This is very much a blind "just do what the task says" solution. It's so simple that it probably runs unadapted in S. I suspect that there is room for an interesting use of R's ifelse function somewhere, but I couldn't find a clean way.

```
fernOfNPoints<-function(n)
{
  currentX<-currentY<-newX<-newY<-0
  plot(0,0,xlim=c(-2,3),ylim=c(0,10),xlab="",ylab="",pch=20,col="darkgreen",cex=0.1
  f1<-function()#ran 1% of the time
  {
    newX<<-0
    newY<<-0.16*currentY
  }
  f2<-function()#ran 85% of the time
  {
    newX<<-0.85*newX+0.04*newY
    newY<<--0.04*newX+0.85*newY+1.6#<<-- is not an error, R's assignment is just tha
  }
  f3<-function()#ran 7% of the time
  {
    newX<<-0.2*newX-0.26*newY
    newY<<-0.23*newX+0.22*newY+1.6
  }
  f4<-function()#ran 7% of the time
  {
    newX<<--0.15*newX+0.28*newY
    newY<<-0.26*newX+0.24*newY+0.44
  }
 for(i in 2:n)#We've already plotted (0,0), so we can skip one run.
 {
   case<-runif(1)
   if(case<=0.01)f1()
   else if(case<=0.86)f2()
   else if(case<=0.93)f3()
   else f4()
   points(newX,newY,pch=20,col="darkgreen",cex=0.1)
 }
  return(invisible())
}
#To plot the fern, use:
fernOfNPoints(500000)
#It will look better if you use a bigger input, but the plot might take a while.
```

```
#I find that there's a large delay between RStudio saying that my code is finished
#If your input is truly big, you may want to reduce the two cex parameters (to make
```

Racket (/wiki/Category:Racket)

File:Racket-barnsley-fern.png (/mw/index.php?title=Special:Upload&wpDestFile=Racket-barnsley-fern.png) : file uploading broken :-(

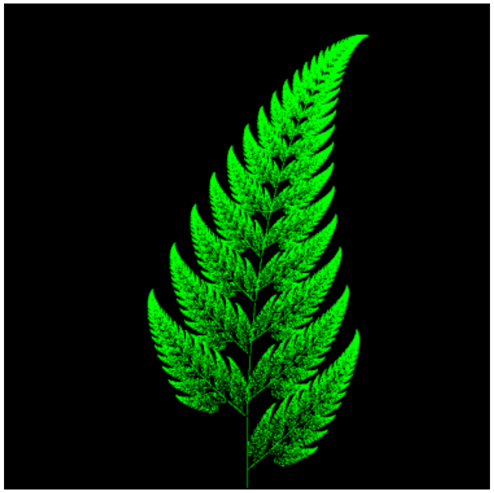
```
#lang racket

(require racket/draw)

(define fern-green (make-color #x32 #xCD #x32 0.66))

(define (fern dc n-iterations w h)
  (for/fold ((x #i0) (y #i0))
            ((i n-iterations))
    (define-values (x′ y′)
      (let ((r (random)))
        (cond
          [(<= r 0.01) (values 0
                               (* y 16/100))]
          [(<= r 0.08) (values (+ (* x 20/100) (* y -26/100))
                               (+ (* x 23/100) (* y 22/100) 16/10))]
          [(<= r 0.15) (values (+ (* x -15/100) (* y 28/100))
                               (+ (* x 26/100) (* y 24/100) 44/100))]
          [else (values (+ (* x 85/100) (* y 4/100))
                        (+ (* x -4/100) (* y 85/100) 16/10))])))

    (define px (+ (/ w 2) (* x w 1/11)))
    (define py (- h (* y h 1/11)))
    (send dc set-pixel (exact-round px) (exact-round py) fern-green)
    (values x′ y′)))


(define bmp (make-object bitmap% 640 640 #f #t 2))

(fern (new bitmap-dc% [bitmap bmp]) 200000 640 640)

bmp
(send bmp save-file "images/racket-barnsley-fern.png" 'png)
```

Raku (/wiki/Category:Raku)

(formerly Perl 6)

Works with: Rakudo (/wiki/Rakudo) version 2016.03
Translation of: Perl

```
use Image::PNG::Portable;

my ($w, $h) = (640, 640);

my $png = Image::PNG::Portable.new: :width($w), :

my ($x, $y) = (0, 0);

for ^2e5 {
    my $r = 100.rand;
    ($x, $y) = do given $r {
        when  $r <=  1 { (                     0,
        when  $r <=  8 { ( 0.20 * $x - 0.26 * $y,
        when  $r <= 15 { (-0.15 * $x + 0.28 * $y,
        default        { ( 0.85 * $x + 0.04 * $y,
    };
    $png.set(($w / 2 + $x * 60).Int, $h - ($y * 6
}

$png.write: 'Barnsley-fern-perl6.png';
```



(/wiki/File:Barnsley-fern-perl6.png)

REXX (/wiki/Category:REXX)

This REXX version is modeled after the **Fortran** entry; it generates an output file ("BARNSLEY.DAT") that
contains the **X** and **Y** coördinates for a scatter plot that can be visualized with a plotting program.

```
/*REXX pgm gens X & Y coördinates for a scatter plot to be used to show a Barnsley
parse arg N FID seed .                           /*obtain optional arguments from t
if   N=='' |   N==","  then   N= 100000          /*Not specified?   Then use the de
if FID=='' | FID==","  then FID= 'BARNSLEY.DAT'  /* "      "          "   "   "
if datatype(seed,'W')  then call random ,,seed   /*if specified, then use random se
call lineout FID, , 1                            /*just set the file ptr to the 1st
x=0                                              /*set the initial value for  X  co
y=0                                              /* "   "     "      "    "   Y
    do #=1  for N                                /*generate   N   number of plot po
    ?=random(, 100)                              /*generate a random number: 0 ≤ ?
       select
       when ?==0  then do;   xx=   0           ;    yy=            .16*y         ;
       when ?< 8  then do;   xx=  .2 *x - .26*y;    yy=  .23*x  +  .22*y  +  1.6 ;
       when ?<15  then do;   xx= -.15*x + .28*y;    yy=  .26*x  +  .24*y  +   .44;
       otherwise             xx=  .85*x + .04*y;    yy= -.04*x  +  .85*y  +  1.6
       end   /*select*/
                             x=xx;                     y=yy
    if #==1  then  do;    minx= x;  maxx= x;        miny= y;  maxy= y
                   end
                          minx= min(minx, x);       miny= min(miny, y)
                          maxx= max(maxx, x);       maxy= max(maxy, y)
    call lineout FID, x","y
    end      /*#*/                               /* [↓]  close the file (safe pract
call lineout FID                                 /*stick a fork in it,  we're all d
```

output is generated to an output file: BARNSLEY.DAT which contains the **X** and **Y** coördinates of a scatter plot.

Ring (/wiki/Category:Ring)

```
Load "guilib.ring"

/*
 +---------------------------------------------------------------------------
 +        Program Name : Draw Barnsley Fern
 +        Purpose      : Draw Fern using Quadratic Equation and Random Number
 +---------------------------------------------------------------------------
*/

###-------------------------------
### DRAW CHART  size 400 x 500
###-------------------------------


New qapp {
        win1 = new qwidget() {
                        ### Position and Size on Screen
                        setwindowtitle("Drawing using QPainter")
                        setgeometry( 10, 25, 400, 500)

                        ### Draw within this Win Box
                        label1 = new qlabel(win1) {
                                        ### Label Position and Size
                                        setgeometry(10, 10, 400, 500)
                                        settext(" ")
                        }

                        buttonFern = new qpushbutton(win1) {
                                        ### Button DrawFern
                                        setgeometry(10, 10, 80, 20)
                                        settext("Draw Fern")
                                        setclickevent("DrawFern()")     ### Call DR
                        }

                        show()
        }
        exec()
}

###------------------------
### FUNCTIONS
###------------------------

Func DrawFern
                p1 = new qpicture()

                colorGreen = new qcolor() { setrgb(0,255,0,255) }
                penGreen   = new qpen()   { setcolor(colorGreen)    setwidth(1) }

                new qpainter() {
                        begin(p1)
                        setpen(penGreen)

                                ###-------------------------------------
                                ### Quadratic equation matrix of arrays

                                a = [ 0,    0.85,  0.2,  -0.15 ]
                                b = [ 0,    0.04, -0.26,  0.28 ]
                                c = [ 0,   -0.04,  0.23,  0.26 ]
                                d = [ 0.16, 0.85,  0.22,  0.24 ]
                                e = [ 0,    0,     0,     0    ]
                                f = [ 0,    1.6,   1.6,   0.44 ]

                                ### Initialize x, y points

                                xf = 0.0
                                yf = 0.0

                                ### Size of output screen

                                MaxX = 400
                                MaxY = 500
                                MaxIterations = MaxY * 200
                                Count = 0
                ###------------------------------------------------

                while ( Count <= MaxIterations )

                        ### NOTE *** RING *** starts at Index 1,
                        ### Do NOT use Random K=0 result

                        k = random() % 100
                        k = k +1

                        ### if  (k = 0)                  k = 1  ok

                                if ((k > 0)  and (k <= 85))  k = 2
                                if ((k > 85) and (k <= 92))  k = 3
                                if  (k > 92)                 k = 4

                        TempX = ( a[k] * xf ) + ( b[k] * yf ) + e[k
                        TempY = ( c[k] * xf ) + ( d[k] * yf ) + f[k

                        xf = TempX
                        yf = TempY

                        if( (Count >= MaxIterations) or (Count != 0
                                xPoint = (floor(xf *  MaxY / 11) + 
                                yPoint = (floor(yf * -MaxY / 11) + 
                                drawpoint( xPoint , yPoint  )
                        ok

                        Count++
                end

                ###------------------------------------------------

        endpaint()
}

label1 { setpicture(p1) show() }
```

```
return
```

Ruby (/wiki/Category:Ruby)

Library: RubyGems (/wiki/Category:RubyGems)
Library: JRubyArt (/wiki/Category:JRubyArt)

```
MAX_ITERATIONS = 200_000

def setup
  sketch_title 'Barnsley Fern'
  no_loop
  puts 'Be patient. This takes about 10 seconds to render.'
end

def draw
  background 0
  load_pixels
  x0 = 0.0
  y0 = 0.0
  x = 0.0
  y = 0.0
  MAX_ITERATIONS.times do
    r = rand(100)
    if r < 85
      x = 0.85 * x0 + 0.04 * y0
      y = -0.04 * x0 + 0.85 * y0 + 1.6
    elsif r < 92
      x = 0.2 * x0 - 0.26 * y0
      y = 0.23 * x0 + 0.22 * y0 + 1.6
    elsif r < 99
      x = -0.15 * x0 + 0.28 * y0
      y = 0.26 * x0 + 0.24 * y0 + 0.44
    else
      x = 0
      y = 0.16 * y
    end
    i = height - (y * 48).to_i
    j = width / 2 + (x * 48).to_i
    pixels[i * height + j] += 2_560
    x0 = x
    y0 = y
  end
  update_pixels
```

```
end

def settings
  size 500, 500
end
```

Run BASIC (/wiki/Category:Run_BASIC)

```
'Barnsley Fern - Run BASIC
  'http://rosettacode.org/wiki/Barnsley_fern#Run_BASIC
  'copy code and run it at http://www.runbasic.com
  '
  ' -----------------------------------
  ' Barnsley Fern
  ' -----------------------------------maxpoints        = 20000
graphic #g, 200, 200
#g fill("blue")
FOR n   = 1 TO maxpoints
p       = RND(0)*100
IF p <= 1 THEN
        nx      = 0
        ny      = 0.16 * y
else if p <= 8 THEN
        nx      = 0.2 * x - 0.26 * y
        ny      = 0.23 * x + 0.22 * y + 1.6
else if p <= 15 THEN
        nx      = -0.15 * x + 0.28 * y
        ny      = 0.26 * x + 0.24 * y + 0.44
else
        nx      = 0.85 * x +0.04 * y
        ny      = -0.04 * x +0.85 * y + 1.6
end if
x       = nx
y       = ny
#g "color green ; set "; x * 17 + 100; " "; y * 17

NEXT n
render #g
#g "flush"
```

Rust (/wiki/Category:Rust)

Translation of: Java

Library: rand (/wiki/Category:Rand)

```
extern crate rand;
extern crate raster;

use rand::Rng;

fn main() {
    let max_iterations = 200_000u32;
    let height = 640i32;
    let width = 640i32;

    let mut rng = rand::thread_rng();
    let mut image = raster::Image::blank(width, height);
    raster::editor::fill(&mut image, raster::Color::white()).unwrap();

    let mut x = 0.;
    let mut y = 0.;
    for _ in 0..max_iterations {
        let r = rng.gen::<f32>();
        let cx: f64;
        let cy: f64;

        if r <= 0.01 {
            cx = 0f64;
            cy = 0.16 * y as f64;
        } else if r <= 0.08 {
            cx = 0.2 * x as f64 - 0.26 * y as f64;
            cy = 0.23 * x as f64 + 0.22 * y as f64 + 1.6;
        } else if r <= 0.15 {
            cx = -0.15 * x as f64 + 0.28 * y as f64;
            cy = 0.26 * x as f64 + 0.26 * y as f64 + 0.44;
        } else {
            cx = 0.85 * x as f64 + 0.04 * y as f64;
            cy = -0.04 * x as f64 + 0.85 * y as f64 + 1.6;
        }
        x = cx;
        y = cy;

        let _ = image.set_pixel(
            ((width as f64) / 2. + x * (width as f64) / 11.).round() as i32,
            ((height as f64) - y * (height as f64) / 11.).round() as i32,
            raster::Color::rgb(50, 205, 50));
    }

    raster::save(&image, "fractal.png").unwrap();
}
```

Scala (/wiki/Category:Scala)

Java Swing Interoperability

```
import (https://scala-lang.org) java.awt._
import (https://scala-lang.org) java.awt.image.BufferedImage

import (https://scala-lang.org) javax.swing._

object (https://scala-lang.org) BarnsleyFern extends (https://scala-lang.org) App {

  SwingUtilities.invokeLater(() =>
    new (https://scala-lang.org) JFrame("Barnsley Fern") {

      private (https://scala-lang.org) class (https://scala-lang.org) BarnsleyFern
        val (https://scala-lang.org) dim = 640
        val (https://scala-lang.org) img = new (https://scala-lang.org) BufferedIma

        private (https://scala-lang.org) def (https://scala-lang.org) createFern(w:
          var (https://scala-lang.org) x, y = 0.0
          for (https://scala-lang.org) (i <- 0 until 200000) {
            var (https://scala-lang.org) tmpx, tmpy = .0
            val (https://scala-lang.org) r = math.random
            if (https://scala-lang.org) (r <= 0.01) {
              tmpx = 0
              tmpy = 0.16 * y
            }
            else (https://scala-lang.org) if (https://scala-lang.org) (r <= 0.08) {
              tmpx = 0.2 * x - 0.26 * y
              tmpy = 0.23 * x + 0.22 * y + 1.6
            }
            else (https://scala-lang.org) if (https://scala-lang.org) (r <= 0.15) {
              tmpx = -0.15 * x + 0.28 * y
              tmpy = 0.26 * x + 0.24 * y + 0.44
            }
            else (https://scala-lang.org) {
              tmpx = 0.85 * x + 0.04 * y
              tmpy = -0.04 * x + 0.85 * y + 1.6
            }
            x = tmpx
            y = tmpy
            img.setRGB((w / 2 + tmpx * w / 11).round.toInt,
       (h - tmpy * h / 11).round.toInt, 0xFF32CD32)
     }
    }

    override (https://scala-lang.org) def (https://scala-lang.org) paintComponen
     super (https://scala-lang.org).paintComponent(gg)
     val (https://scala-lang.org) g = gg.asInstanceOf[Graphics2D]
     g.setRenderingHint(RenderingHints.KEY_ANTIALIASING, RenderingHints.VALUE_
     g.drawImage(img, 0, 0, null (https://scala-lang.org))
    }

    setBackground(Color.white)
    setPreferredSize(new (https://scala-lang.org) Dimension(dim, dim))
    createFern(dim, dim)
   }

   add(new (https://scala-lang.org) BarnsleyFern, BorderLayout.CENTER)
   pack()
   setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE)
   setLocationRelativeTo(null (https://scala-lang.org))
   setResizable(false (https://scala-lang.org))
   setVisible(true (https://scala-lang.org))
  })

}
```

Scheme (/wiki/Category:Scheme)

This version creates a list of points, defining the fern, which are then rescaled and output to an eps file.

```
(import (scheme base)
        (scheme cxr)
        (scheme file)
        (scheme inexact)
        (scheme write)
        (srfi 27))     ; for random numbers

(define (create-fern x y num-points)
  (define (new-point xn yn)
    (let ((r (* 100 (random-real))))
      (cond ((< r 1) ; f1
             (list 0 (* 0.16 yn)))
            ((< r 86) ; f2
             (list (+ (* 0.85 xn) (* 0.04 yn))
                   (+ (* -0.04 xn) (* 0.85 yn) 1.6)))
            ((< r 93) ; f3
             (list (- (* 0.2 xn) (* 0.26 yn))
                   (+ (* 0.23 xn) (* 0.22 yn) 1.6)))
            (else ; f4
              (list (+ (* -0.15 xn) (* 0.28 yn))
                    (+ (* 0.26 xn) (* 0.24 yn) 0.44))))))
  ;
  (random-source-randomize! default-random-source)
  (do ((i 0 (+ i 1))
       (pts (list (list x y)) (cons (new-point (caar pts) (cadar pts)) pts)))
    ((= i num-points) pts)))

;; output the fern to an eps file
(define (output-fern-as-eps filename fern)
  (when (file-exists? filename) (delete-file filename))
  (with-output-to-file
    filename
    (lambda ()
      (let* ((width 600)
             (height 800)
             (min-x (apply min (map car fern)))
             (max-x (apply max (map car fern)))
             (min-y (apply min (map cadr fern)))
             (max-y (apply max (map cadr fern)))
             (scale-x (/ (- width 50) (- max-x min-x)))
             (scale-y (/ (- height 50) (- max-y min-y)))
             (scale-points (lambda (point)
                             (list (truncate (+ 20 (* scale-x (- (car point) min-x)
                                   (truncate (+ 20 (* scale-y (- (cadr point) min-y

        (display
          (string-append "%!PS-Adobe-3.0 EPSF-3.0\n%%BoundingBox: 0 0 "
                         (number->string width) " " (number->string height) "\n"))

        ;; add each point in fern as an arc - sets linewidth based on depth in tree
        (for-each (lambda (point)
                    (display
                      (string-append (number->string (list-ref point 0))
                                     " "
                                     (number->string (list-ref point 1))
                                     " 0.1 0 360 arc\nstroke\n")))
                  (map scale-points fern))
        (display "\n%%EOF")))))

(output-fern-as-eps "barnsley.eps" (create-fern 0 0 50000))
```

Scilab (/wiki/Category:Scilab)

Works with: Scilab (/wiki/Scilab) version 5.4.0 and above
This version creates a list of points, defining the fern, and shows them on a graphic window which can then be saved to a file via the GUI or the console by the user.

```
iteractions=1.0d6;

XY=zeros(2,iteractions+1);
x=0;
y=0;

i=2;
while i<iteractions+2
    random_numbers=rand();
    xp=x;
    if random_numbers(1) < 0.01 then
        x = 0;
        y = 0.16 * y;
    elseif random_numbers(1) >= 0.01 & random_numbers(1) < 0.01+0.85 then
        x = 0.85 * x + 0.04 * y;
        y = -0.04 * xp + 0.85 * y + 1.6;
    elseif random_numbers(1) >= 0.86 & random_numbers(1) < 0.86+0.07 then
        x = 0.2 * x - 0.26 * y;
        y = 0.23 * xp + 0.22 * y + 1.6;
    else
        x = -0.15 * x + 0.28 * y;
        y = 0.26 * xp + 0.24 * y + 0.44;
    end

    XY(1,i)=x;
    XY(2,i)=y;

    i=i+1;
end

scf(0);
clf();
xname('Barnsley fern');
plot2d(XY(1,:),XY(2,:),-0)
axes=gca();
axes.isoview="on";
axes.children.children.mark_foreground=13;
```

SequenceL (/wiki/Category:SequenceL)

Tail-Recursive SequenceL Code:

```
import <Utilities/Math.sl>;
import <Utilities/Random.sl>;

transform(p(1), rand) :=
    let
        x := p[1]; y := p[2];
    in
        [0.0, 0.16*y] when rand <= 0.01
    else
        [0.85*x + 0.04*y, -0.04*x + 0.85*y + 1.6] when rand <= 0.86
    else
        [0.2*x - 0.26*y, 0.23*x + 0.22*y + 1.6] when rand <= 0.93
    else
        [-0.15*x + 0.28*y, 0.26*x + 0.24*y + 0.44];

barnsleyFern(rand, count, result(2)) :=
    let
        nextRand := getRandom(rand);
        next := transform(result[size(result)], nextRand.value / 2147483647.0);
    in
        result when count <= 0
    else
        barnsleyFern(nextRand.generator, count - 1, result ++ [next]);

scale(p(1), width, height) := [round((p[1] + 2.182) * width / 4.8378),
                               round((9.9983 - p[2]) * height / 9.9983)];

entry(seed, count, width, height) :=
    let
        fern := barnsleyFern(seedRandom(seed), count, [[0.0,0.0]]);
    in
        scale(fern, width, height);
```

C++ Driver Code:

Library: CImg (/wiki/Category:CImg)

```
#include "SL_Generated.h"
#include "CImg.h"

using namespace cimg_library;

int main(int argc, char** argv)
{
    int threads = 0; if(argc > 1) threads = atoi (https://www.opengroup.org/onlinepu
    int width = 300; if(argc > 2) width = atoi (https://www.opengroup.org/onlinepubs
    int height = 600; if(argc > 3) height = atoi (https://www.opengroup.org/onlinepu
    int steps = 10000; if(argc > 4) steps = atoi (https://www.opengroup.org/onlinepu
    int seed = 314159; if(argc > 5) seed = atoi (https://www.opengroup.org/onlinepub

    CImg<unsigned char> visu(width, height, 1, 3, 0);
    Sequence< Sequence<int> > result;

    sl_init(threads);

    sl_entry(seed, steps, width-1, height-1, threads, result);

    visu.fill(0);
    for(int i = 1; i <= result.size(); i++)
        visu(result[i][1], result[i][2],1) = 255;

    CImgDisplay draw_disp(visu);
    draw_disp.set_title("Barnsley Fern in SequenceL");
    visu.display(draw_disp);

    while(!draw_disp.is_closed()) draw_disp.wait();

    sl_done();

    return 0;
}
```

Output:

Output Screenshot (https://i.imgur.com/zerRZo8.png)

Sidef (/wiki/Category:Sidef)

```
require('Imager')

var w = 640
var h = 640

var img   = %O<Imager>.new(xsize => w, ysize => h, channels => 3)
var green = %O<Imager::Color>.new('#00FF00')

var (x, y) = (0.float, 0.float)

1e5.times {
  var r = 100.rand
  (x, y) = (
    if    (r <=  1) { ( 0.00*x - 0.00*y,  0.00*x + 0.16*y + 0.00) }
    elsif (r <=  8) { ( 0.20*x - 0.26*y,  0.23*x + 0.22*y + 1.60) }
    elsif (r <= 15) { (-0.15*x + 0.28*y,  0.26*x + 0.24*y + 0.44) }
    else            { ( 0.85*x + 0.04*y, -0.04*x + 0.85*y + 1.60) }
  )
  img.setpixel(x => w/2 + 60*x, y => 60*y, color => green)
}

img.flip(dir => 'v')
img.write(file => 'barnsleyFern.png')
```

Output image: Barnsley fern (https://github.com/trizen/rc/blob/master/img/barnsley-fern-sidef.png)

SPL (/wiki/Category:SPL)

```
w,h = #.scrsize()
x,y = 0
>
  r = #.rnd(100)
  ? r<85, x,y = f2(x,y)
  ? r!<85 & r<92, x,y = f3(x,y)
  ? r!<92 & r<99, x,y = f4(x,y)
  ? r!<99, x,y = f1(y)
  #.drawpoint(x/10*w+w/2,h-y/10*h,0,0.5,0,0.1)
<
f1(y) <= 0, 0.16*y
f2(x,y) <= 0.85*x+0.04*y, -0.04*x+0.85*y+1.6
f3(x,y) <= 0.2*x-0.26*y, 0.23*x+0.22*y+1.6
f4(x,y) <= -0.15*x+0.28*y, 0.26*x+0.24*y+0.44
```

Standard ML (/wiki/Category:Standard_ML)

Works with PolyML. Random generator copy from the Random_numbers#Standard_ML (/wiki/Random_numbers#Standard_ML) task. Window slimmed down from Animation#Standard_ML (/wiki/Animation#Standard_ML).

```
open XWindows ;
open Motif ;

val uniformdeviate = fn seed =>
 let
  val in31m = (Real.fromInt o Int32.toInt ) (getOpt (Int32.maxInt,0) );
  val in31 = in31m +1.0;
  val (s1,s2,v) = (41160.0 , 950665216.0 , Real.realFloor seed);
  val (val1,val2) = (v*s1, v*s2);
  val next1 = Real.fromLargeInt (Real.toLargeInt IEEEReal.TO_NEGINF (val1/in31)) ;
  val next2 = Real.rem(Real.realFloor(val2/in31) , in31m );
  val valt = val1+val2 - (next1+next2)*in31m;
  val nextt = Real.realFloor(valt/in31m);
  val valt = valt - nextt*in31m;
 in
  (valt/in31m,valt)
end;



local
 val sizeup = 60.0 ;
 fun toI {x=x,y=y} = {x=Real.toInt  IEEEReal.TO_NEAREST (sizeup *x),y=Real.toInt  I
 val next  = [  (fn {x=x,y=y} =>  {x= 0.0,           y= 0.16*y             })
              , (fn {x=x,y=y} =>  {x= 0.85*x+0.04*y, y= ~0.04*x+0.85*y+1.6})
              , (fn {x=x,y=y} =>  {x= 0.2*x-0.26*y,  y= 0.23*x+0.22*y+1.6 })
              , (fn {x=x,y=y} =>  {x= ~0.15*x+0.28*y,y= 0.26*x+0.24*y+0.44}) ] ;
 val seed  = ref 100027.0
in

 fun putNext  1 win usegc coord =  XFlush (XtDisplay win)
 |   putNext  N win usegc coord =
  let
   val (i,ns) =  uniformdeviate ( !seed ) ;
   val _      =  seed := ns  ;
   val fi     =  List.nth (next, List.foldr (fn (a,b) => b + (if i>a then 1 else 0)
   val nwp    =  fi coord
  in
      (XDrawPoint (XtWindow win) usegc  ( AddPoint ((XPoint o toI) coord, XPoint {x
        putNext (N-1) win usegc nwp  )
  end

end;



val demoWindow = fn () =>
let
  val shell     =  XtAppInitialise      ""    "demo" "top" [] [ XmNwidth 600, XmNhe
  val main      =  XmCreateMainWindow   shell    "main"       [ XmNmappedWhenManage
  val canvas    =  XmCreateDrawingArea  main   "drawarea"     [ XmNwidth 600, XmNhe
  val usegc     =  DefaultGC (XtDisplay canvas) ;
  val _         =  XSetForeground usegc 0x4a632d ;
  val drawall   =  fn (w,c,t)=> ( XClearWindow (XtWindow canvas ); putNext 1000000 
in
  (
   XtSetCallbacks   canvas [ (XmNexposeCallback ,  drawall)  ] XmNarmCallback ;
   XtManageChild    canvas ;
   XtManageChild    main   ;
   XtRealizeWidget  shell
  )
end ;
```

call

```
demoWindow () ;
```

Swift (/wiki/Category:Swift)

Output is viewable in a playground.

```
import UIKit
import CoreImage
import PlaygroundSupport

let imageWH = 300
let context = CGContext(data: nil,
                        width: imageWH,
                        height: imageWH,
                        bitsPerComponent: 8,
                        bytesPerRow: 0,
                        space: CGColorSpace(name: CGColorSpace.sRGB)!,
                        bitmapInfo: CGImageAlphaInfo.premultipliedFirst.rawValue)!
var x0 = 0.0
var x1 = 0.0
var y0 = 0.0
var y1 = 0.0

context.setFillColor(#colorLiteral(red: 0, green: 0, blue: 0, alpha: 1))
context.fill(CGRect(x: 0, y: 0, width: imageWH, height: imageWH))
context.setFillColor(#colorLiteral(red: 0.539716677, green: 1, blue: 0.265400682, a

for _ in 0..<100_000 {
    switch Int(arc4random()) % 100 {
    case 0:
        x1 = 0
        y1 = 0.16 * y0
    case 1...7:
        x1 = -0.15 * x0 + 0.28 * y0
        y1 = 0.26 * x0 + 0.24 * y0 + 0.44
    case 8...15:
        x1 = 0.2 * x0 - 0.26 * y0
        y1 = 0.23 * x0 + 0.22 * y0 + 1.6
    default:
        x1 = 0.85 * x0 + 0.04 * y0
        y1 = -0.04 * x0 + 0.85 * y0 + 1.6
    }

    context.fill(CGRect(x: 30 * x1 + Double(imageWH) / 2.0, y: 30 * y1,
                        width: 1, height: 1))

    (x0, y0) = (x1, y1)
}

let uiImage = UIImage(cgImage: context.makeImage()!)
```

TI-83 BASIC (/wiki/Category:TI-83_BASIC)

```
ClrDraw
Input "ITERS:",M
[[0,0,1]]→[A]
[[0,0,0][0,.16,0][0,0,1]]→[B]
[[.85,-.04,0][.04,.85,0][0,1.6,1]]→[C]
[[.2,.23,0][-.26,.22,0][0,1.6,1]]→[D]
[[-.15,.26,0][.28,.24,0][0,.44,1]]→[E]
0→I
While I<M
randInt(1,100)→R

If R=1
Then
[A][B]→[A]
101→R
End

If R<86
Then
[A][C]→[A]
101→R
End

If R<93
Then
[A][D]→[A]
101→R
End

If R<101:Then
[A][E]→[A]
End

round([A](1,1)*8+31,0)→E
round([A](1,2)*8,0)→F
Pxl-On(E,F)
I+1→I
End
```

Unicon (/wiki/Category:Unicon)

Library: graphics (/wiki/Category:Graphics)

```
link graphics

global x, y

procedure main()
    &window := open("FERN", "g", "size=400,400", "bg=black")

    x := y := 0

    repeat {
        draw()
        delay(30)
        if *Pending() > 0 then {
            case Event() of {
                "q"|"\e": return
            }
        }
    }
end

procedure next_point()
    local nx, ny, r

    nx := 0.0
    ny := 0.0

    r := ?100

    if r < 1 then {
        nx := 0.0
        ny := 0.16 * y
    } else if r < 86 then {
        nx := 0.85 * x + 0.04 * y
        ny := -0.04 * x + 0.85 * y + 1.6
    } else if r < 93 then {
        nx := 0.2 * x - 0.26 * y
        ny := 0.23 * x + 0.22 * y + 1.6
    } else {
        nx := -0.15 * x + 0.28 * y
        ny := 0.26 * x + 0.24 * y + 0.44
    }

    x := nx
    y := ny
end

procedure map(v:real, a, b, c, d)
    return (v - a) / (b - a) * (d - c) + c;
end

procedure draw_point()
    local px, py

    px := map(x, -2.1820, 2.6558, 0.0, 400.0)
    py := map(y, 0.0, 9.9983, 400.0, 0.0)

    Fg("green")
    DrawPoint(px, py)
end

procedure draw()
    every i := 0 to 10000 do {
        draw_point()
        next_point()
    }
end
```

VBA (/wiki/Category:VBA)

```
Private Sub plot_coordinate_pairs(x As Variant, y As Variant)
    Dim chrt As Chart
    Set chrt = ActiveSheet.Shapes.AddChart.Chart
    With chrt
        .ChartType = xlXYScatter
        .HasLegend = False
        .SeriesCollection.NewSeries
        .SeriesCollection.Item(1).XValues = x
        .SeriesCollection.Item(1).Values = y
    End With
End Sub
Public Sub barnsley_fern()
    Const MAX = 50000
    Dim x(MAX) As Double
    Dim y(MAX) As Double
    x(0) = 0: y(0) = 0
    For i = 1 To MAX
        Select Case CInt(100 * Rnd)
            Case 0 To 1
                x(i) = 0
                y(i) = 0.16 * y(i - 1)
            Case 2 To 85
                x(i) = 0.85 * x(i - 1) + 0.04 * y(i - 1)
                y(i) = -0.04 * x(i - 1) + 0.85 * y(i - 1) + 1.6
            Case 86 To 92
                x(i) = 0.2 * x(i - 1) - 0.26 * y(i - 1)
                y(i) = 0.23 * x(i - 1) + 0.22 * y(i - 1) + 1.6
            Case 93 To 100
                x(i) = -0.15 * x(i - 1) + 0.28 * y(i - 1)
                y(i) = 0.26 * x(i - 1) + 0.24 * y(i - 1) + 0.44
        End Select
    Next i
    plot_coordinate_pairs x, y
End Sub
```

/* Visual Basic .NET (/wiki/Category:Visual_Basic_.NET) */ Section added

Visual Basic .NET (/wiki /Category:Visual_Basic_.NET)

Works with: Visual Basic .NET (/wiki/Visual_Basic_.NET) version 2011

```
' Barnsley Fern - 11/11/2019
Public Class BarnsleyFern

    Private Sub BarnsleyFern_Paint(sender As Object, e As PaintEventArgs) Handles M
        Const Height = 800
        Dim x, y, xn, yn As Double
        Dim f As Double = Height / 10.6
        Dim offset_x As UInteger = Height \ 4 - Height \ 40
        Dim n, r As UInteger
        Dim Bmp As New Drawing.Bitmap(Height \ 2, Height) 'x,y
        'In Form: xPictureBox As PictureBox(800,400)
        xPictureBox.Image = Bmp
        For n = 1 To Height * 50
            r = Int (https://www.google.com/search?q=INT+site:msdn.microsoft.com)(R
            Select Case r
                Case 0 To 84
                    xn = 0.85 * x + 0.04 * y
                    yn = -0.04 * x + 0.85 * y + 1.6
                Case 85 To 91
                    xn = 0.2 * x - 0.26 * y
                    yn = 0.23 * x + 0.22 * y + 1.6
                Case 92 To 98
                    xn = -0.15 * x + 0.28 * y
                    yn = 0.26 * x + 0.24 * y + 0.44
                Case Else
                    xn = 0
                    yn = 0.16 * y
            End Select
            x = xn : y = yn
            Bmp.SetPixel(offset_x + x * f, Height - y * f, Color.FromArgb(0, 255, 0
        Next n
    End Sub 'Paint

End Class 'BarnsleyFern
```

XPL0 (/wiki/Category:XPL0)

```
int  N, R;
real NX, NY, X, Y;
[SetVid($12);           \set 640x480x4 VGA graphics (on PC or RPi)
X:= 0.0;  Y:= 0.0;
for N:= 0 to 200_000 do
        [R:= Ran(100);  \0..99
        case of
        R < 1:  [NX:= 0.0;              NY:= 0.16*Y];
        R < 8:  [NX:= 0.20*X - 0.26*Y;  NY:= 0.23*X + 0.22*Y + 1.60];
        R < 15: [NX:=-0.15*X + 0.28*Y;  NY:= 0.26*X + 0.24*Y + 0.44]
        other   [NX:= 0.85*X + 0.04*Y;  NY:=-0.04*X + 0.85*Y + 1.60];
        X:= NX;  Y:= NY;
        Point(320+fix(X*40.0), 440-fix(Y*40.0), 2\green\);
        ]
]
```

Yabasic (/wiki/Category:Yabasic)

Translation of: ZX Spectrum Basic
Classic style

```
10 REM Fractal Fern
20 LET wid = 800 : LET hei = 600 : open window wid, hei : window origin "cb"
25 backcolor 0, 0, 0 : color 0, 255, 0 : clear window
30 LET maxpoints=wid*hei/2: LET x=0: LET y=0
40 FOR n=1 TO maxpoints
50 LET p=RAN(100)
60 IF p<=1 LET nx=0: LET ny=0.16*y: GOTO 100
70 IF p<=8 LET nx=0.2*x-0.26*y: LET ny=0.23*x+0.22*y+1.6: GOTO 100
80 IF p<=15 LET nx=-0.15*x+0.28*y: LET ny=0.26*x+0.24*y+0.44: GOTO 100
90 LET nx=0.85*x+0.04*y: LET ny=-0.04*x+0.85*y+1.6
100 LET x=nx: LET y=ny
110 DOT x*wid/12,y*hei/12
120 NEXT n
```

Modern style

```
REM Fractal Fern
wid = 800 : hei = 600 : open window wid, hei : window origin "cb"
backcolor 0, 0, 0 : color 0, 255, 0 : clear window
maxpoints = wid * hei / 2 : x = 0 : y = 0
for n = 1 to maxpoints
   p = ran(100)
   if p <= 1 then nx = 0 : ny = 0.16 * y
   elseif p <= 8 then nx = 0.2 * x - 0.26 * y : ny = 0.23 * x + 0.22 * y + 1.6
   elseif p <= 15 then nx = -0.15 * x + 0.28 * y : ny = 0.26 * x + 0.24 * y + 0.44
   else nx = 0.85 * x + 0.04 * y : ny = -0.04 * x + 0.85 * y + 1.6
   end if
   x = nx : y = ny
   dot x * wid / 12, y * hei / 12
next
```

zkl (/wiki/Category:Zkl)

Uses the PPM class from http://rosettacode.org/wiki/Bitmap/Bresenham%27s_line_algorithm#zkl (https://rosettacode.org/wiki/Bitmap/Bresenham%27s_line_algorithm#zkl)

Translation of: Java

(/wiki/File:BarnsleyFern.zkl.jpg)

```
fcn barnsleyFern(){
   w,h:=640,640;
   bitmap:=PPM(w+1,h+1,0xFF|FF|FF);  // White background

   x,y, nx,ny:=0.0, 0.0, 0.0, 0.0;
   do(0d100_000){
      r:=(0).random(100);  // [0..100)%
      if     (r<= 1) nx,ny= 0,                0.16*y;
      else if(r<= 8) nx,ny= 0.2*x  - 0.26*y,  0.23*x + 0.22*y + 1.6;
      else if(r<=15) nx,ny=-0.15*x + 0.28*y,  0.26*x + 0.24*y + 0.44;
      else           nx,ny= 0.85*x + 0.04*y, -0.04*x + 0.85*y + 1.6;
      x,y=nx,ny;
      bitmap[w/2 + x*60, y*60] = 0x00|FF|00;  // Green dot
   }
   bitmap.writeJPGFile("barnsleyFern.jpg");
}();
```

ZX Spectrum Basic (/wiki/Category:ZX_Spectrum_Basic)

Translation of: zkl

```
10 REM Fractal Fern
20 PAPER 7: BORDER 7: BRIGHT 1: INK 4: CLS
30 LET maxpoints=20000: LET x=0: LET y=0
40 FOR n=1 TO maxpoints
50 LET p=RND*100
60 IF p<=1 THEN LET nx=0: LET ny=0.16*y: GO TO 100
70 IF p<=8 THEN LET nx=0.2*x-0.26*y: LET ny=0.23*x+0.22*y+1.6: GO TO 100
80 IF p<=15 THEN LET nx=-0.15*x+0.28*y: LET ny=0.26*x+0.24*y+0.44: GO TO 100
90 LET nx=0.85*x+0.04*y: LET ny=-0.04*x+0.85*y+1.6
100 LET x=nx: LET y=ny
110 PLOT x*17+127,y*17
120 NEXT n
```

It is recommended to run on an emulator that supports running at full speed.

Categories (/wiki/Special:Categories): Programming Tasks (/wiki/Category:Programming_Tasks)
| Solutions by Programming Task (/wiki/Category:Solutions_by_Programming_Task)
| Ada (/wiki/Category:Ada) | SDLAda (/mw/index.php?title=Category:SDLAda&action=edit&redlink=1)
| ALGOL 68 (/wiki/Category:ALGOL_68) | Applesoft BASIC (/wiki/Category:Applesoft_BASIC)
| BBC BASIC (/wiki/Category:BBC_BASIC) | C (/wiki/Category:C) | C sharp (/wiki/Category:C_sharp)
| C++ (/wiki/Category:C%2B%2B) | Qt (/wiki/Category:Qt) | Common Lisp (/wiki/Category:Common_Lisp)
| Delphi (/wiki/Category:Delphi) | EasyLang (/wiki/Category:EasyLang) | Forth (/wiki/Category:Forth)
| SDL2 (/wiki/Category:SDL2) | Fortran (/wiki/Category:Fortran) | FreeBASIC (/wiki/Category:FreeBASIC)
| Frink (/wiki/Category:Frink) | Fōrmulæ (/wiki/Category:F%C5%8Drmul%C3%A6)
| G'MIC (/mw/index.php?title=Category:G%27MIC&action=edit&redlink=1)
| Gnuplot (/wiki/Category:Gnuplot)
| Pages with broken file links (/mw/index.php?title=Category:Pages_with_broken_file_links&action=edit&redlink=1)
| Go (/wiki/Category:Go) | Haskell (/wiki/Category:Haskell) | IS-BASIC (/wiki/Category:IS-BASIC)
| J (/wiki/Category:J) | Java (/wiki/Category:Java) | JavaScript (/wiki/Category:JavaScript)
| Julia (/wiki/Category:Julia) | Kotlin (/wiki/Category:Kotlin) | Lambdatalk (/wiki/Category:Lambdatalk)
| Liberty BASIC (/wiki/Category:Liberty_BASIC) | Locomotive Basic (/wiki/Category:Locomotive_Basic)
| Lua (/wiki/Category:Lua) | Mathematica (/wiki/Category:Mathematica)
| Wolfram Language (/wiki/Category:Wolfram_Language) | MiniScript (/wiki/Category:MiniScript)
| Nim (/wiki/Category:Nim) | Oberon-2 (/wiki/Category:Oberon-2) | PARI/GP (/wiki/Category:PARI/GP)
| Perl (/wiki/Category:Perl) | Phix (/wiki/Category:Phix) | Phix/pGUI (/wiki/Category:Phix/pGUI)
| PicoLisp (/wiki/Category:PicoLisp) | Processing (/wiki/Category:Processing)
| Processing Python mode (/wiki/Category:Processing_Python_mode)
| PureBasic (/wiki/Category:PureBasic) | Python (/wiki/Category:Python) | R (/wiki/Category:R)
| Racket (/wiki/Category:Racket) | Raku (/wiki/Category:Raku) | REXX (/wiki/Category:REXX)
| Ring (/wiki/Category:Ring) | Ruby (/wiki/Category:Ruby) | RubyGems (/wiki/Category:RubyGems)
| JRubyArt (/wiki/Category:JRubyArt) | Run BASIC (/wiki/Category:Run_BASIC)
| Rust (/wiki/Category:Rust) | Rand (/wiki/Category:Rand) | Scala (/wiki/Category:Scala)
| Scheme (/wiki/Category:Scheme) | Scilab (/wiki/Category:Scilab)
| SequenceL (/wiki/Category:SequenceL) | CImg (/wiki/Category:CImg) | Sidef (/wiki/Category:Sidef)
| SPL (/wiki/Category:SPL) | Standard ML (/wiki/Category:Standard_ML) | Swift (/wiki/Category:Swift)
| TI-83 BASIC (/wiki/Category:TI-83_BASIC) | Unicon (/wiki/Category:Unicon)

| Graphics (/wiki/Category:Graphics) | VBA (/wiki/Category:VBA)
| Visual Basic .NET (/wiki/Category:Visual_Basic_.NET) | XPL0 (/wiki/Category:XPL0)
| Yabasic (/wiki/Category:Yabasic) | Zkl (/wiki/Category:Zkl)
| ZX Spectrum Basic (/wiki/Category:ZX_Spectrum_Basic)

This page was last modified on 24 January 2021, at 20:26.