9　フラクタル図形

再帰手続きを使うと，簡単なプログラムで，複雑で面白い図形を描くことができます。

前回の講義の数列（むしろ文字列） $C_9 = 121312141213121512131214\cdots$ には同じ部分列（たとえば，1213121）が何回も繰り返して現れます。

これを文字ではなく図形にするわけです。すると，自分自身に相似な図形が中に何重にも入り組んで現れる面白いフラクタル図形が描かれます。

9.1　プログラムの準備

9.1.1　新規プログラムを保存

フォルダー	Fractal
ユニット	FractalU.pas
プロジェクト	FractalP.dpr

9.1.2　フォーム

Name	FormFractal
Caption	フラクタル図形（学生証番号，氏名）
WindowState	wsMaximized

9.1.3　メインパネルと終了ボタン

いつものように，PanelMain と ButtonClose を配置。

9.1.4　描画基本命令

Delphi に元々用意されている MoveTo(x,y)，LineTo(x,y) は行き先の座標 (x, y) を指定します。フラクタル図形を描くときは，進行方向 θ と距離 d を指定すると便利なので，そのためのメソッドを定義します。なお，今回は Image を使わないで Form の Canvas に描画します。その理由は，描いている途中の様子を見たいからです。Image の Canvas に描くと図が完成するまで表示されませんが，Form の Canvas に描くと途中も表示されます。

TFormFractal の宣言部に追加

```
private
  XNow,YNow : Real;                    // 現在の位置
public
  procedure StandBy(X,Y : Integer);    // 出発点を指定する。
  procedure Move(Muki,Kyori : Real);   // Muki 方向に Kyori だけ進む
  procedure Line(Muki,Kyori : Real);   // Muki 方向に Kyori だけ線を引く
```

実現部を書く

```
implementation
{£R *.DFM}
(******************** 一般のプロシージャ ********************)
(******************** フォームのメソッド ********************)
procedure TFormFractal.StandBy(X,Y : Integer);
          (* 出発点を指定する。 *)
begin
  Canvas.MoveTo(X,Y);
  XNow := X;
  YNow := Y;
end; (* StandBy *)

procedure TFormFractal.Move(Muki,Kyori : Real);
          (* Muki 方向に Kyori だけ進む *)
begin
  XNow := XNow + Kyori*Cos(Muki);
  YNow := YNow + Kyori*Sin(Muki);
  Canvas.MoveTo(Round(XNow),Round(YNow));
end; (* Move *)

procedure TFormFractal.Line(Muki,Kyori : Real);
          (* Muki 方向に Kyori だけ線を引く *)
begin
  XNow := XNow + Kyori*Cos(Muki);
  YNow := YNow + Kyori*Sin(Muki);
  Canvas.LineTo(Round(XNow),Round(YNow));
end; (* Line *)
```

9.1.5　消去ボタンと描画ボタン

Buttone を 2 つメインパネルに置く。

Name	ButtonClear	ButtonDraw
Caption	消去 (&C)	描画 (&D)

ButtonClear の OnClick ハンドラー

```
procedure TFormFractal.ButtonClearClick(Sender: TObject);
begin
  with Canvas do
    begin
      Brush.Color := clWhite;
      FillRect(ClipRect);
    end;
end; (* ButtonClearClick *)
```

ButtonDraw の OnClick ハンドラー

フラクタル図形を描く前に，基本描画命令を理解するテストとして正多角形を描いてみましょう。この基本描画命令を使うと正多角形を簡単に描くことができます。

```
procedure TFormFractal.ButtonDrawClick(Sender: TObject);
var
  Muki : Real;
  N,K  : Byte;
begin
  N := 9;                          // 頂点の数を指定する
  StandBy(200,100);                // 始点
  Muki := 0;                       // 最初の向き
  for K := 1 to N do
    begin
      Line(Muki,100);              // 長さ 100 の線を引く
      Muki := Muki+2*Pi/N;         // 向きを 360° /N 変える
    end;
end; (* ButtonDrawClick *)
```

問題

これを次のように変更するとどんな図を描くでしょう。実行する前に，どんな図になるか考えてください。

```
procedure TFormFractal.ButtonDrawClick(Sender: TObject);
var
  Muki : Real;
  N,K  : Byte;
begin
  N := 9;                          // 頂点の個数を指定する
  StandBy(200,100);                // 始点
  Muki := 0;                       // 最初の向き
  for K := 1 to 100 do             (*_*) // N を 100 に変更
    begin
      Line(Muki,100-K);            (*_*) // 100 を 100-K に変更
      Muki := Muki+2*Pi/N;         // 向きを 360° /N 変える
    end;
end; (* ButtonDrawClick *)
```

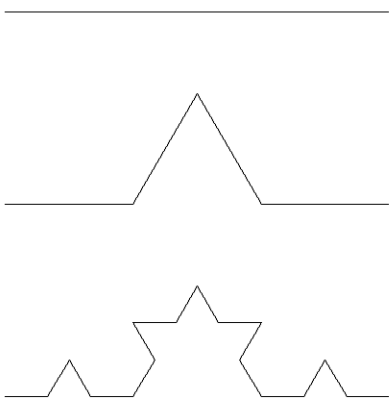
9.2 コッホ曲線 (Koch Curve)

9.2.1 作り方

(1) 1 本の線分を，長さが $\frac{1}{3}$ の線分 4 本からなる折れ線で置き換える。

(2) その 4 本の線分を，また同様の折れ線で置き換える。

(3) 以下同様。



9.2.2 再帰手続き Koch

TFormFractal のメソッドとして追加。

```
end; (* Line *)

procedure TFormFractal.Koch(Level : Byte; Muki,Nagasa : Real);
          (* Muki 方向に長さ Nagasa のレベル Level コッホ曲線を描く（再帰）*)
begin
  if Level <= 1
    then Line(Muki,Nagasa)                    // 実際に線を引く
    else begin                                // 長さ 1/3, Level-1 コッホ曲線を描く
           Koch(Level-1,Muki,Nagasa/3);       // 指定方向に描く
           Koch(Level-1,Muki-Pi/3,Nagasa/3); // 左に 60° 方向に描く
           Koch(Level-1,Muki+Pi/3,Nagasa/3); // 右に 60° 方向に描く
           Koch(Level-1,Muki,Nagasa/3);       // 指定方向に描く
         end;
end; (* Koch *)
```

9.2.3 コッホ曲線を描くメソッド

```
end; (* Koch *)

procedure TFormFractal.DrawKoch(Level : Byte);
          (* レベル Level-コッホ曲線を描く *)
begin
  StandBy(300,200);                           // 始点
  Koch(Level,Pi/6,400);                       // コッホ曲線
end; (* DrawKoch *)
```

9.2.4　描画ボタンの OnClick ハンドラー変更

```
procedure TFormFractal.ButtonDrawClick(Sender: TObject);
begin
  DrawKoch(2);
end; (* ButtonDrawClick *)
```

9.2.5　実行

始点，向き，長さ，レベルを書き換えて実行してみましょう。

9.2.6　レベル指定

実行時にレベルを変えられるようにします。

Panel をメインパネルの中に置く。

Align	alTop	
Alignment	alLeftJustify	
Name	PanelLevel	
Caption	□レベル	(□は全角空白)

Edit を PanelLevel の中央に置く。

Name	EditLevel
Text	1

Win32/UpDown を EditLevel の右に置く。

Name	UpDownLevel
Min	1
Max	10
Associate	EditLevel

Associate を EditLevel にすると，UpDownLevel の Position の値と EditLevel の Text が連動します。一方を変更すると他方に反映されます。

9.2.7　描画ボタンの OnClick ハンドラー変更

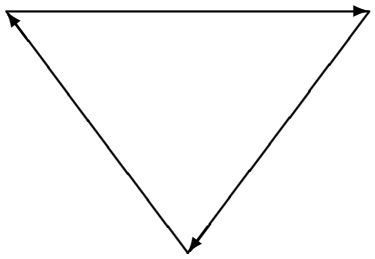
```
procedure TFormFractal.ButtonDrawClick(Sender: TObject);
var
  Level : Byte;
begin
  Level := UpDownLevel.Position;
  DrawKoch(Level);
  UpDownLevel.Position := Level+1;          // 自動的にレベルをアップする
end; (* ButtonDrawClick *)
```

9.2.8　実行

EditLevel に直接数を入れて変更することも，UpDownLevel で数を変更することもできることを確認しなさい。EditLevel に 1〜10 以外の数を入れるとどうなりますか。

9.3　問題

コッホ曲線を 3 つ正三角形状に描くと面白い図ができます。向きが影響するので，右図の矢印の順に進むように，3 つ描きます。レベル 1 のときはこの正三角形が描かれます。レベルが大きいとき，どんな図形になるか，実行前に考えてみましょう。



9.3.1　メソッド KochThree を追加

```
end; (* DrawKoch *)

procedure TFormFractal.KochThree(Level : Byte);
          (* 正三角形状にコッホ曲線を３つ描く *)
begin

  (* 正三角形状に描く *)
  (* 内部を水色で塗りつぶす *)

end;  (* KochThree *)
```

9.3.2　描画ボタンの OnClick ハンドラーを変更

```
procedure TFormFractal.ButtonDrawClick(Sender: TObject);
var
  Level : Byte;
begin
  Level := UpDownLevel.Position;
  // DrawKoch(Level);
  KochThree(Level);
  UpDownLevel.Position := Level+1;          // 自動的にレベルをアップする
end; (* ButtonDrawClick *)
```

9.3.3　実行

どんな図形になりましたか。
三角形を逆向きに進んだ図も描いてみましょう。