2　グラフィックスの基礎

グラフィックスの基本である線を描くプログラムを作成します。

2.1　原型

2.1.1　新規アプリケーション

新規アプリケーションに，次の名前をつけて保存する。

フォルダー	Sen
ユニット	SenU
プロジェクト	SenP

フォームのプロパティを変更する。

Name	FormMain
Caption	プログラムの基礎－線を描く－
Position	poScreenCenter

ユニットの {$R *.DFM} の下にプロシージャを分類して書くためのコメントを記入する。

```
(******************** 一般のプロシージャ ********************)
(******************** フォームのメソッド ********************)
(******************** イベントハンドラー ********************)
```

2.1.2　メインパネル

Panel を フォームの中に置く。

Align	alRight	右の辺に接する
Name	PanelMain	
Caption		

2.1.3　終了ボタン

Button をメインパネルの中，下の方に置く。

Name	ButtonClose
Caption	終了 (&X)

OnClick ハンドラー

イベントハンドラーは，オブジェクトインスペクターのイベントの該当する欄をダブルクリックして，スケルトンその他を自動作成して実行部だけを記入することを，思い出してください。

下記の行を全部自分で書いてはいけません。

```
(******************** イベントハンドラー ********************)
procedure TFormMain.ButtonCloseClick(Sender: TObject);
begin
  Close;
end; {ButtonCloseClick}
```

2.1.4　メインイメージ

グラフィックスは Image コンポーネント（Additional グループにある）を使います。
Additional/Image をフォームの中，メインパネルの左に置く。

```
Align          alClient          フォームのメインパネル以外全体になる
Name           ImageMain
```

OnMouseDown，OnMouseMove，OnMouseUp の各ハンドラー

イメージメインの中で，マウスボタンを押したままマウスを動かすと，その軌跡を描くようにします。

それには，Boolean 変数 DownFlag にマウスボタンを押している状態かどうか（押しているとき True，放しているとき False）を記憶して，マウスを動かしたときにボタンを押している状態ならば,（前のマウスの位置から）マウスの位置まで線を引くようにします。

```
procedure TFormMain.ImageMainMouseDown(Sender: TObject;
  Button: TMouseButton; Shift: TShiftState; X, Y: Integer);
begin
  ImageMain.Canvas.MoveTo(X,Y);                // ペンを (X,Y) に移動する
  DownFlag := True;                            // 描画開始
end; {ImageMouseDown}

procedure TFormMain.ImageMainMouseMove(Sender: TObject; Shift: TShiftState;
  X, Y: Integer);
begin
  if DownFlag                                  // 描画中ならば
    then ImageMain.Canvas.LineTo(X,Y);         // (X,Y) まで線を引く
end; {ImageMainMouseMove}

procedure TFormMain.ImageMainMouseUp(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
begin
  DownFlag := False;                           // 描画終了
end; {ImageMouseUp}
```

変数 DownFlag はフォームの private 部に宣言します。

```
  private
    { Private 宣言 }
    DownFlag : Boolean;          // マウスボタンを押した状態かどうか
```

2.1.5　実行

実行して，簡単な絵を描いて見ましょう。

2.2　画面最大化

実行したばかりのときは，メインイメージはフォームの色（clSilver）をしていますが，何かを描画すると白く変わります。このとき，イメージのキャンバスの作業領域がメモリに作らたのです。それまでは，サイズを変更できますが，いったん作られるとその後では描画領域のサイズは変更されません。そのことを調べてみましょう。フォームの右上にある _□× の □ をクリックするとフォームが最大（画面いっぱい）になります。メインイメージの Align プロパティを alClient にしてあるので，フォームのサイズを変えるとメインイメージもサイズが変わります。

(1) 実行してすぐにフォームを最大化してから描画する。
(2) 実行して描画してからフォームを最大化して，さらに描画する。

違いがわかりますね。(2) の場合でも，メインイメージは拡大されてメインパネル以外いっぱいになっています。ですから，描画領域の外でマウスダウンして，ボタンを押したまま描画領域にマウスを移動すると線を描きます（メインパネルの中でマウスダウンした場合は線を描きません）。なお，イメージのサイズと描画領域のサイズが食い違っている場合は，描画するたびに画面がちらついてしまいます。

2.2.1　終了ボタンの位置

フォームを最大化すると，終了ボタンがメインパネルの中ほどに位置するので気持ち悪いですね。ボタンの位置をパネルの上からの距離（ドット数）で定めているのでこうなります。パネルの下からの距離で定めるように指定することができます。

```
Anchors.akTop      False
Anchors.akBottom   True
```

実行してみましょう。最大化しても，終了ボタンが下の方にあります。

2.3　座標表示

マウスの位置を表示するようにします。

2.3.1　座標ラベル

Label をメインパネルの中に置く。

Align	alTop	
Name	LabelZahyou	
Caption	(0000,0000)	
AutoSize	False	キャプションに応じてサイズが変わるのを禁止する
Color	clWhite	
Height	40	
Alignment	taCenter	キャプションの水平位置を中央にする
Layout	tlCenter	キャプションの垂直位置を中央にする
Font.Name	MS ゴシック	P を消す
Font.Size	12	

2.3.2　メインイメージの OnMouseMove ハンドラーを修正

```
procedure TFormMain.ImageMainMouseMove(Sender: TObject; Shift: TShiftState;
  X, Y: Integer);
begin
  if DownFlag                                    // 描画中ならば
    then ImageMain.Canvas.LineTo(X,Y);           // (X,Y) まで線を引く
  LabelZahyou.Caption := Format('(%4d,%4d)',[X,Y]);
end; {ImageMainMouseMove}
```

Format(書式, データリスト) はデータを書式にしたがって文字列に変換します。上の例では, (　　,　　) がそのままの文字で, 間に X と Y の値が 10 進法 4 桁で入ります。コンソールアプリケーションで Write('(',X:4,',',Y:4,')') としたときの出力と同じです。%4d を %2x とすると 16 進法 2 桁で表されます。前回のプログラム Fukusosuu では, %6.3f を使いました。これは実数データの値を全体で 6 桁, 小数部を 3 桁で表すように指定するものです。

2.3.3　実行

メインイメージの中でマウスを動かすと座標が表示されます

2.4　直線

原型はマウスの軌跡を描きましたが, マウスをダウンした点とアップした点を結ぶ直線も描けるようにします。そして, 軌跡を描くか直線を描くか実行時に選べるようにします。

2.4.1　軌跡選択ラジオグループ

RadioGroup をメインパネルの中に置く。

Align	alTop	
Name	RadioGroupKiseki	
Caption	軌跡か直線か	
Items	軌跡	下記参照
	直線	
Columns	2	横に並べる
ItemIndex	0	デフォルトを軌跡にする
Height	48	

Items は, 右の欄をダブルクリックすると入力画面が現れるので, ‘‘軌跡’’ と ‘‘直線’’ の 2 行を書きます。

ItemIndex が選択している Item の番号 $(0, 1, 2, \cdots)$ を表しています。実行中に, ‘‘直線’’ を選ぶと ItemIndex が 1 に変わります。

2.4.2　ImageMain のイベントハンドラーを修正

直線を描くように変更するのは簡単です。マウスを移動したときに線を引かないで，マウスボタンを放したときに線を引けばよいのです。

```
procedure TFormMain.ImageMainMouseMove(Sender: TObject; Shift: TShiftState;
  X, Y: Integer);
begin
  if DownFlag                                    // 描画中ならば
     and (RadioGroupKiseki.ItemIndex = 0)        // 軌跡を描く場合
    then ImageMain.Canvas.LineTo(X,Y);           // (X,Y) まで線を引く
  LabelZahyou.Caption := Format('(%4d,%4d)',[X,Y]);
end; {ImageMainMouseMove}

procedure TFormMain.ImageMainMouseUp(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
begin
  ImageMain.Canvas.LineTo(X,Y);                  // (X,Y) まで線を引く
  DownFlag := False;                             // 描画終了
end; {ImageMouseUp}
```

2.4.3　実行

座標ラベルを参考にして，頂点が A(200,100),B(100,200),C(200,300),D(300,200) の（傾いた）正方形 ABCD を描きなさい。

2.5　線の色

スクリーンに表示する字や絵の色は**光の三原色** ―赤，緑，青―の組み合わせで決まります。三原色それぞれの色の明るさが，0〜255（16 進法で表すと$00〜$FF）の 256 段階あります。赤，緑，青の明るさを R, G, B とすると，それを組み合わせた色は

$$R + G * 256 + B * 256^2 = R + G * \$100 + B * \$10000$$

の数で表されます。

代表的な色には cl〜 という名前が付いています。

	暗い		明るい		
黒	clBlack	$000000	clSillver	$C0C0C0	
赤	clMaroon	$000080	clRed	$0000FF	赤
緑	clGreen	$008000	clLime	$00FF00	緑
黄	clOlive	$008080	clYellow	$00FFFF	緑＋赤
青	clNavy	$800000	clBlue	$FF0000	青
紫	clPurple	$800080	clFuchsia	$FF00FF	青＋　赤
水	clTeal	$808000	clAqua	$FFFF00	青＋緑
白	clGray	$808080	clWhite	$FFFFFF	青＋緑＋赤

2.5.1　色パネルとラベル

Panel をメインパネルの中に置く。

Align	alTop	
Alignment	taLeftJustify	
Name	PanelIro	
Caption	線の色　$000000	
Font.Name	MS ゴシック	

このパネルの中に Label を置く。

Align	alRight	
Name	LabelIro	
AutoSize	False	
Caption		
Color	0	clBlack になる

2.5.2　赤パネルとトラックバー

Panel をメインパネルの中に置く。

Align	alTop	
Name	PanelRed	
Caption		
Color	clRed	

このパネルの中に Win32/TrackBar を置く。

Align	alClient	
Name	TrackBarRed	

同様に，緑パネルとトラックバー，青パネルとトラックバーも作ってください。ただし，緑パネルの色は clGreen ではなく clLime にします。

トラックバーは Min 〜 Max の数直線上をサムを動かして，好きな値を選ぶためのものです。Position がサムの位置の値を示します。

3 つのトラックバーに共通の設定

赤トラックバーをクリックして選び，Shift きーを押しながら緑トラックバーをクリック，さらに青トラックバーをクリックすると，3 つのトラックバーがまとめて選択されます。3 つが選択されていることを確認してください。この状態で，プロパティを変えたりイベントハンドラーを書くと，3 つ同時に設定できます。

Max	255	
Frequency	17	17 の倍数の所だけ｜をつける
PageSize	17	端をクリックすると 17 ずつ移動する
Position	3	サムが移動したことを確認したら
Position	0	0 に戻す

OnChange イベントハンドラー

３つのトラックバーが選択されている状態で，イベントの OnChange 欄をダブルクリックしてスケルトンを作り，実行部を記入します。

```
procedure TFormMain.TrackBarRedChange(Sender: TObject);
var
  R,G,B : Byte;
  Iro : TColor;
begin
  R := TrackBarRed.Position;
  G := TrackBarGreen.Position;
  B := TrackBarBlue.Position;
  Iro := R+G*$100+B*$10000;
  LabelIro.Color := Iro;
  PanelIro.Caption := Format('  線の色＝$%6x',[Iro]);
  ImageMain.Canvas.Pen.Color := Iro;
end; {TrackBarRedChange}
```

名前は TrackBarRedChange ですが，３つに共通です。３つのトラックバーの OnChange 欄に TrackBarRedChange が書かれていることを確認してください。もし書かれていない場合は，欄の右側の▼をクリックして TrackBarRedChange を選びます。

2.5.3　実行

三原色を組み合わせて様々な色ができることを確認してください。
できるだけ金色に近い色を作ってみましょう。