18　トランプ (1)

トランプのゲームを作るための準備をします。

18.1　トランプの絵柄をダウンロードする

トランプの札 52 枚と裏面および無地の 54 個の絵柄を 1 枚のシートにしたビットマップファイルを解凍するプログラムがホームページにあります。このプログラムは Trump というフォルダーを作ってその中にトランプシートのビットマップファイルを解凍します。

```
http://www.ss.u-tokai.ac.jp/~ooya/Jugyou/JoronB/TrumpBMP.exe
```

(1) ダウンロードして実行する

(2) 「解凍する場所」に Trump を作りたいフォルダーを指定して「ＯＫ」する。

18.2　トランプの例プログラム

(1) ［ファイル｜新規作成］でアプリケーションを作成する。

フォルダー	Trump	先ほどビットマップファイルを解凍したフォルダー
ユニット	ExampleU	
プロジェクト	ExampleP	

(2) フォームのプロパティ。

Name	FormMain
Caption	トランプ使用例　学生証番号　氏名

18.3　トランプのユニット

トランプのいろいろなゲームに共通の，トランプを作ったりシャッフルしたりするプロシージャは独立したユニットに作っておくと便利なので，今回はメインとは別にもう 1 つフォームを使います。

(1) ［ファイル｜新規作成］でフォームを作成して,［ファイル｜名前をつけて保存］する。

ユニット	TrumpU

(2) フォームのプロパティ。

Name	FormTrump
Caption	トランプのシート

(3) Image を FormTrump の中に置く。

Name	ImageTrump
Top	0
Left	0
AutoSize	True
Picture	下記のようにする

Picture 欄の右欄の … をクリックすると，“画像の設定” というダイアログが出ます。「読み込み」をして，さっきダウンロードした Trump.bmp を読み込み,「Ok」で確定します。

フォームを広げるとシート全体が見えます。確認したらフォームを小さくしてかまいません。

以後，FormTrump を変更することはないので，右上にある __□× の __ をクリックして最小化してしまう方が後の作業がしやすいでしょう。

ユニットは，ExampleU と TrumpU の両方とも変更していきます。間違えないように注意してください。

(4) ユニット ExampleU を書き換える。

FormMain を少し下の方にずらすと，ユニットが現れます。ExampleU のタブをクリックして，ExampleU を表示させます。

上の方にある uses の最後に TrumpU を追加します。

```
uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  TrumpU;
```

これで，ExampleU の中で TrumpU の public 部のデータやメソッドが使えるようになりました（private 部のデータやメソッドは他のユニットでは使えません）。

(5) メインパネル

Panel を FormMain の中に置く。

Name	PanelMain
Align	alBottom
Caption	

(6) 終了ボタン

Button をメインパネルの中，右端に置く。

Name	ButtonClose	
Caption	終了	
Anchors.Left	False	
Anchors.Right	True	FormMain のサイズを変更しても右端にある

OnClick ハンドラー

```
(******************** FormMain の Method ********************)
(******************** Event Handler ********************)
procedure TFormMain.ButtonCloseClick(Sender: TObject);
begin
  Close;
end; {ButtonCloseClick}
```

(7) 実行ボタン

Button を PanelMain の中に置く。

Name　ButtonJikkou

Caption　実行

OnClick ハンドラー

```
procedure TFormMain.ButtonJikkouClick(Sender: TObject);
var
  CanvasMoto : TCanvas;
begin
  CanvasMoto := FormTrump.ImageTrump.Canvas;
  Canvas.CopyRect(Rect(100,100,172,196),CanvasMoto,Rect(432,192,504,288));
  Canvas.CopyRect(Bounds(300,300,72,96),CanvasMoto,Bounds(720,288,72,96));
end; {ButtonJikkouClick}
```

(8) 実行

実行ボタンを押すと，FormMain の Canvas に，FormTrump にあるトランプのシートから 2 枚のカードの絵をコピーします。

CopyRect(コピー先長方形，コピー元キャンバス，コピー元長方形)

の長方形の指定のしかたは 2 通りあります。

(1) Rect(x_1, y_1, x_2, y_2)　(x_1, y_1) と (x_2, y_2) を対角線の両端とする長方形
(2) Bounds$(x_1.y_1, w, h)$　(x_1, y_1) を頂点とし横幅が w，縦幅が h の長方形

$x_2 < x_1$ すなわち $w < 0$ にすることもできます。コピー元と，コピー先の長方形の大きさを変えてもかまいません。それぞれどんな絵に変わるでしょうか，次のように追加して見てみましょう。

```
procedure TFormMain.ButtonJikkouClick(Sender: TObject);
var
  CanvasMoto : TCanvas;
begin
  CanvasMoto := FormTrump.ImageTrump.Canvas;
  Canvas.CopyRect(Rect(100,100,172,196),CanvasMoto,Rect(432,192,504,288));
  Canvas.CopyRect(Rect(272,100,200,196),CanvasMoto,Rect(432,192,504,288));
  Canvas.CopyRect(Rect(100,296,172,200),CanvasMoto,Rect(432,192,504,288));
  Canvas.CopyRect(Rect(272,296,200,200),CanvasMoto,Rect(432,192,504,288));
  Canvas.CopyRect(Bounds(300,300, 72, 96),CanvasMoto,Bounds(720,288,72,96));
  Canvas.CopyRect(Bounds(500,300,-72, 96),CanvasMoto,Bounds(720,288,72,96));
  Canvas.CopyRect(Bounds(300,400,144,192),CanvasMoto,Bounds(720,288,72,96));
  Canvas.CopyRect(Bounds(600,300, 36, 96),CanvasMoto,Bounds(720,288,72,96));
end; {ButtonJikkouClick}
```

18.4　トランプを作る

シートのままでは使いようがありません。52 個の Image に分割したいのですが，これを設計時に手作業で行うのは大変なので，実行時にプログラムが行うようにします。

それぞれの Card を Image を拡張したコンポーネントとして作り，それら 52 個をまとめて Trump という Object にします。

(1)　ユニット TrumpU に追加する。

```
type
  TSuit   = (stClub,stDiamond,stHeart,stSpade,stOther);
  TRank   = 1..13;
  TCardNo = 0..51; (* カード:  0=クラブA,    ..., 12=クラブK    *)
                   (*          13=ダイヤA,    ..., 25=ダイヤK    *)
                   (*          26=ハートA,    ..., 38=ハートK    *)
                   (*          39=スペードA, ..., 51=スペードK *)
  TCard   = class (TImage)
              private
                Omote  : Boolean;
              public
                Suit   : TSuit;
                Rank   : TRank;
                constructor Create(AOwner : TComponent); override;
                procedure OmoteNiSuru;
                procedure UraNiSuru;
                procedure Uragaesu;
            end;
  TTrump  = class(TObject)
              public
                Cards : array [TCardNo] of TCard;
                procedure Sakusei(Oya : TWinControl);
            end;
  TFormTrump = class(TForm)
    ImageTrump: TImage;
  private
  public
  end;

var
  FormTrump: TFormTrump;

implementation

{$R *.DFM}
(********** TCard の Method **********)
constructor TCard.Create(AOwner : TComponent);
            (* Card を生成する *)
  begin
    inherited;                          // Image を生成
    Parent  := TWinControl(AOwner);     // 親（配置する場所）
    Width   := 72;                      // 横幅
    Height  := 96;                      // 縦幅
    Stretch := True;                    // 拡大縮小できるように
    Visible := True;                    // 表示する
  end; {Create}
```

```
procedure TCard.OmoteNiSuru;
          (* 表の絵にする，すなわちシートから絵柄をコピーする *)
  var
    RectMoto : TRect;
  begin
    RectMoto := Bounds((Rank-1)*72,Ord(Suit)*96,72,96);
    with Canvas do
      CopyRect(ClipRect,FormTrump.ImageTrump.Canvas,RectMoto);
    Omote := True;
  end; {OmoteNiSuru}

procedure TCard.UraNiSuru;
          (* 裏の絵にする *)
  var
    RectMoto : TRect;
  begin
    RectMoto := Bounds(0*72,4*96,72,96);
    with Canvas do
      CopyRect(ClipRect,FormTrump.ImageTrump.Canvas,RectMoto);
    Omote := False;
  end; {UraNiSuru}

procedure TCard.Uragaesu;
          (* 表裏を，反対にする *)
  begin
    if Omote
      then UraNiSuru
      else OmoteNiSuru;
  end; {Uragaesu}

(********** TTrump のメソッド **********)
procedure TTrump.Sakusei(Oya : TWinControl);
          (* 52 枚のカードを作成する *)
  var
    CardNo: TCardNo;
  begin
    for CardNo := 0 to 51 do
      begin
        if Cards[CardNo] = Nil                  // まだ作ってないなら
          then Cards[CardNo] := TCard.Create(Oya);    // Oya の中に生成する
        with Cards[CardNo] do
          begin
            Left    := CardNo*12;               // 左端の位置
            Top     := CardNo*8;                // 上端の位置
            Tag     := CardNo;                  // 札から番号がわかるようにするため
            Suit    := TSuit(CardNo div 13);    // スート Ord(Suit) = 0〜3
            Rank    := (CardNo mod 13)+1;       // ランク 1〜13
            OmoteNiSuru;                        // 表の絵にする
          end;
      end;
  end; {TrumpSakusei}

end.
```

(2) ExampleU のユニットとフォームを変更する。

(a) TFormMain の private 部に追加

```
  private
    Trump : TTrump;
```

(b) FormMain の OnCreate ハンドラー。

```
procedure TFormMain.FormCreate(Sender: TObject);
begin
  Trump := TTrump.Create;
end; {FormCreate}
```

(c) Button をメインパネルの中に置く。

Name	ButtonSakusei
Caption	作成

OnClick ハンドラー

```
procedure TFormMain.ButtonSakuseiClick(Sender: TObject);
begin
  Trump.Sakusei(Self);
  ButtonSakusei.Visible := False;    // 作成は 1 度だけ
  ButtonJikkou.Enabled := True;
end; {ButtonSakuseiClick}
```

(d) ButtonJikkouClick を変更する。

Enabled	False

OnClick ハンドラー

```
procedure TFormMain.ButtonJikkouClick(Sender: TObject);
var
  CardNo : TCardNo;
begin
  for CardNo := 26 to 38 do
    Trump.Cards[CardNo].Uragaesu;
end; {ButtonJikkouClick}
```

(3) 実行する。

作成ボタンを押すと，52 枚のカードが少しずつずれて（Sakusei で指定した Left,Top の値による）描かれます。

実行ボタンを押す度に，ハートの札（カード番号 26〜38）が裏返しになります。

18.5　画像反転

トランプのゲーム中に選択したカードがわかりやすいように色を反転する（補色にする）機能を追加します。

(1) TrumpU を変更する。

(a) TCard の定義に追加する。

```
  TCard   = class (TImage)
              private
                Omote  : Boolean;
                Nega   : Boolean;
              public
                Suit   : TSuit;
                Rank   : TRank;
                constructor Create(AOwner : TComponent); override;
                procedure OmoteNiSuru;
                procedure UraNiSuru;
                procedure Uragaesu;
                procedure PosiNiSuru;
                procedure NegaNiSuru;
                procedure HantenSuru;
            end;
```

(b) 追加した 3 つのメソッドの実現部を書く。

```
(********** TCard のメソッド **********)
procedure TCard.HantenSuru;
          (* ポジネガを，反対にする *)
  begin
    with Canvas do
      begin
        Pen.Mode := pmNot;
        Rectangle(ClipRect);
      end;
    Nega := not Nega;
  end; {HantenSuru}

procedure TCard.PosiNiSuru;
          (* ポジにする *)
  begin
    if Nega
      then HantenSuru;
  end; {PosiNiSuru}

procedure TCard.NegaNiSuru;
          (* ネガにする *)
  begin
    if not Nega
      then HantenSuru;
  end; {NegaNiSuru}
```

(c) TCard.OmoteNiSuru，UraNiSuru を変更する。

CopyRect は Pen.Mode の影響を受けないので，元の画像の色は変化しません。すなわち，必ずポジになります。ゆえに，Card の Nega プロパティをそれに合わせるようにします。

```
procedure TCard.OmoteNiSuru;
          (* 表の絵にする *)
  var
    RectMoto : TRect;
  begin
    RectMoto := Bounds((Rank-1)*72,Ord(Suit)*96,72,96);
    with Canvas do
      CopyRect(ClipRect,FormTrump.ImageTrump.Canvas,RectMoto);
    Omote := True;
    Nega  := False;
  end; {OmoteNiSuru}

procedure TCard.UraNiSuru;
          (* 裏の絵にする *)
  var
    RectMoto : TRect;
  begin
    RectMoto := Bounds(0*72,4*96,72,96);
    with Canvas do
      CopyRect(ClipRect,FormTrump.ImageTrump.Canvas,RectMoto);
    Omote := False;
    Nega  := False;
  end; {UraNiSuru}
```

(2) ExampleU の ButtonJikkouClick を変更する。

```
procedure TFormMain.ButtonJikkouClick(Sender: TObject);
var
  CardNo : TCardNo;
begin
  for CardNo := 26 to 38 do
    Trump.Cards[CardNo].Uragaesu;
  for CardNo := 0 to 51 do
    if Odd(CardNo)
      then Trump.Cards[CardNo].HantenSuru;
end; {ButtonJikkouClick}
```

(3) 実行する。

1 枚おきにポジとネガになります。

18.6　前面移動，後面移動

Delphi のコンポーネント（Panel,Button,Image,...）は後から置いたものが前に置いたものより前面に置かれます。

TTrump.Sakusei で 52 枚の Card（Image を拡張したコンポーネント）を Oya に配置しますが，クラブのAからスペードのKの順で生成するので，最初に置いたクラブのAが最も奥に，最後に置いたスペードのKが最も手前に配置されます。

(1) 生成する順序を変えてみる。

TTrump.Sakusei の

```
for CardNo := 0 to 51 do
```

を

```
for CardNo := 51 downto 0 do
```

に変更して，実行してみましょう。

配置する場所は，Left と Top で決まりますから変わりません。重なっている部分のどちらが前面（手前）になるかどうかの違いです。

理解したら，TTrump.Sakusei を元に戻してください。

(2) ExampleU の ButtonJikkouClick を変更する。

コンポーネントには，前面に移動する BringToFront と，後面に移動する SendToBack があります。これを用いて，実行中に前後を変更することができます。

```
procedure TFormMain.ButtonJikkouClick(Sender: TObject);
var
  CardNo : TCardNo;
begin
  for CardNo := 26 to 38 do
    Trump.Cards[CardNo].Uragaesu;
  for CardNo := 0 to 51 do
    if Odd(CardNo)
      then Trump.Cards[CardNo].HantenSuru
      else Trump.Cards[CardNo].BringToFront;
end; {ButtonJikkouClick}
```

実行して確かめましょう。

“CardNo := 0 to 51” を “CardNo := 51 downto 0” に変えたり，“BringToFront” を “SendToBack” に変えて実行してみましょう。

(3) カードを小さくして，スート別に並べるように変更する。

```
procedure TFormMain.ButtonJikkouClick(Sender: TObject);
var
  CardNo : TCardNo;
begin
  for CardNo := 0 to 51 do
    begin
```

```
        with Trump.Cards[CardNo] do
          begin
            Height := 72;
            Width  := 54;
            Top    := Ord(Suit)*80;
            Left   := Rank*36;
          end;
      end;
end; {ButtonJikkouClick}
```

(4) **問題**

下図のようにカードを配置するように，ButtonJikkouClick を書き換えなさい。

その他の部分，特に TrumpU を書き換えてはいけません。