【セッションNo．2】

Delphi/400技術セッション

VCL開発者が知っておきたいFireMonkeyアプリ開発のポイント

株式会社ミガロ.
RAD事業部　営業・営業推進課
尾崎　浩司

【アジェンダ】

1. Delphi/400 ２つの開発フレームワーク

2. FireMonkey アプリ開発入門

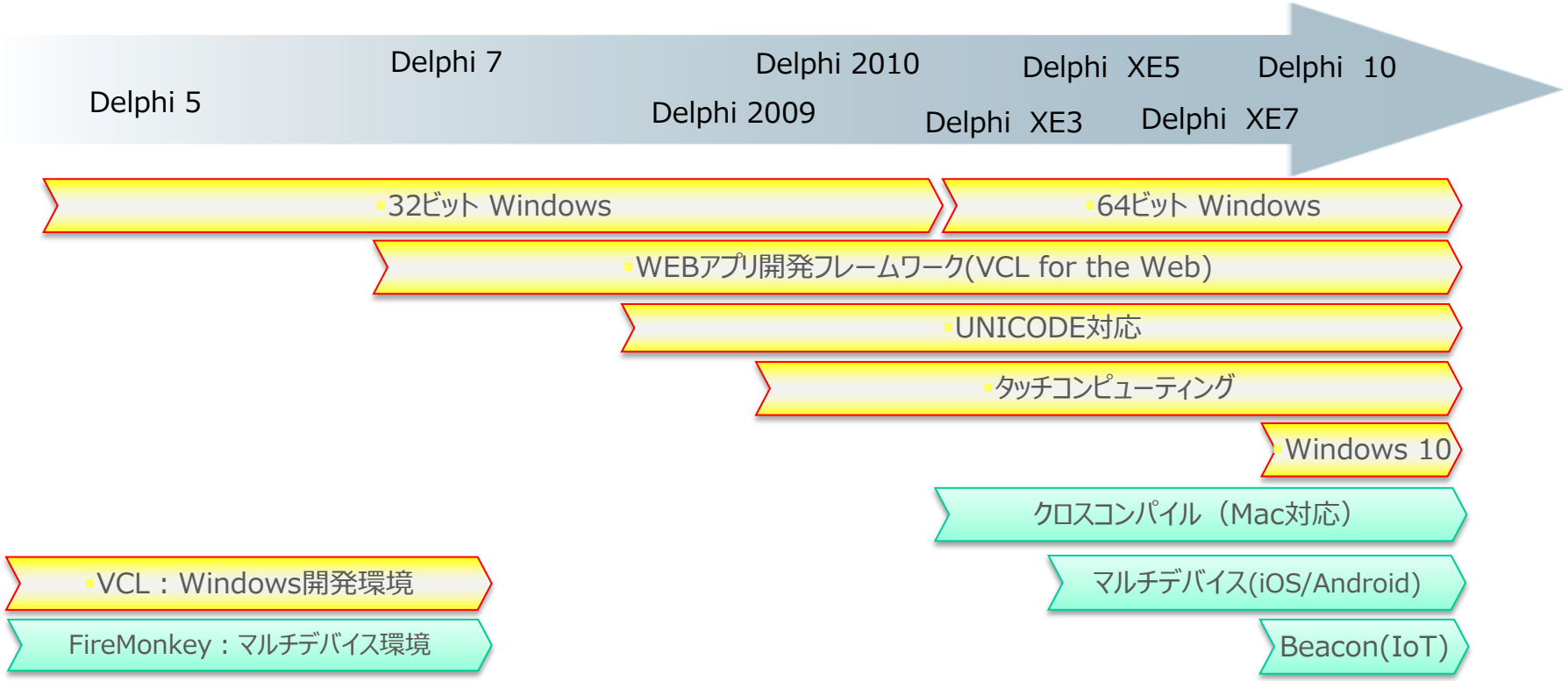
3. FireMonkey 効果的な機能の活用

4. まとめ

1．Delphi/400 ２つの開発フレームワーク

Delphi/400 ２つの開発フレームワーク

- VCLとFireMonkey
  - Delphi/400登場当時から、時代とともに進化をしながらも、一貫したWindowsネイティブ開発環境を提供するVCLフレームワーク
  - Delphi/400 XE3にてWin/Mac対応から始まり、現在ではWin/Mac/iOS/Androidの4つのプラットフォームに対応したFireMonkeyフレームワーク



■ VCLフレームワーク

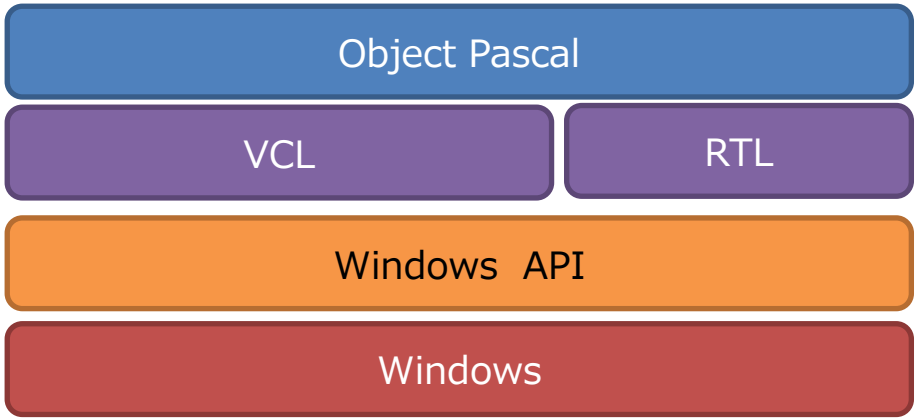
- Windowsアプリ開発専用フレームワーク
  - Windows API をラッピングしたフレームワーク
  - Windowsが提供するすべての機能が使用できる
  - 開発者は、VCLコンポーネント＋RTL（ランタイムライブラリ）を使用して開発

VCLフォームユニット

```
unit Unit1;

interface

uses
  Winapi.Windows, Winapi.Messages, System.SysUtils,
  System.Variants, System.Classes, Vcl.Graphics,
  Vcl.Controls, Vcl.Forms, Vcl.Dialogs;

type
  TForm1 = class(TForm)
  private
    { Private 宣言 }
  public
    { Public 宣言 }
  end;

var
  Form1: TForm1;
```

Vcl：VCLコンポーネントライブラリユニット
Winapi：WindowsAPIユニット
System：RTLユニット

VCLフレームワーク



FireMonkeyフレームワーク

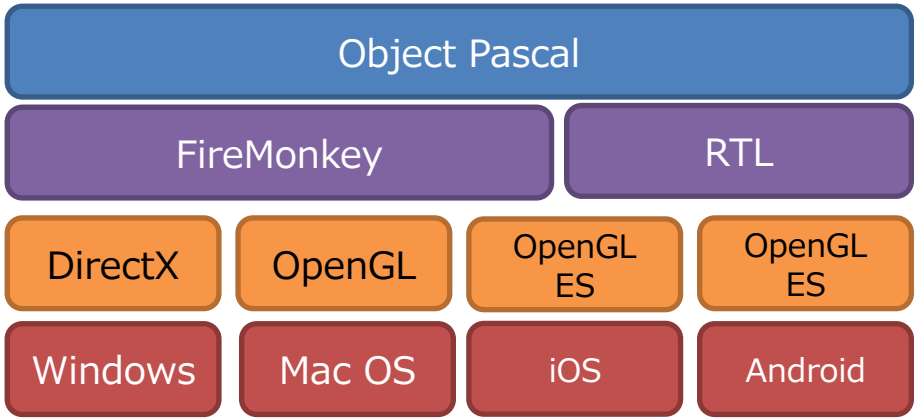
- マルチデバイスアプリ開発フレームワーク
  - グラフィックス処理装置（GPU）を使用したフレームワーク
  - OS固有のAPIに依存しない為、マルチデバイス化が可能
  - 開発者は、FireMonkeyコンポーネント＋RTLを使用して開発

FireMonkeyフォームユニット

```
unit Unit1;

interface

uses
  System.SysUtils, System.Types, System.UITypes,
  System.Classes, System.Variants, FMX.Types, FMX.Controls,
  FMX.Forms, FMX.Graphics, FMX.Dialogs;

type
  TForm1 = class(TForm)
  private
    { private 宣言 }
  public
    { public 宣言 }
  end;

var
  Form1: TForm1;
```

FMX：FireMonkeyコンポーネントライブラリユニット
System：RTLユニット（**VCLと共通**）

FireMonkeyフレームワーク



FireMonkey

- FireMonkeyは、スマートフォン等に対応したマルチデバイス用フレームワークだが、PCアプリ開発用途でも、もちろん使用可能

- メリット
  - Windowsに限定すれば、従来からのC/Sアプリと同じ開発が可能
    （BDEは使用できないが、dbExpress/FireDACはVCL同様使用可能）
  - WindosAPIに依存したVCLでは表現できないような多彩な部品が使用できる
  - アニメーションやEffect効果を組み合わせることで、モダンなアプリが構築できる

- 留意点
  - VCLと比べ、入力系コンポーネントやグリッドの機能が若干劣る
  - WindowAPIを直接使用するような場合、VCLの方が取り扱いやすい
  - VCLと比べバージョンアップ時の変更箇所が大きい場合がある

FireMonkey

- FireMonkeyに向いたPCアプリとは？
  - クライアントPC　マルチOS対応
    - Windows　に加え　Mac　も
      業務アプリクライアントとして活用

Windows

Mac

  - アニメーション効果等が実現しやすい特長を活用するもの
    - タブレットPCで動作するアプリケーション（キーボード入力操作の少ないもの）
    - ビューア（照会）系アプリケーション
    - デジタル・サイネージ（電子看板）

タブレットPC

画像ビューアソフト

空港のフライト情報

■ 今回のポイント

- PCアプリ開発におけるFireMonkey使用のポイントを紹介！

  - FireMonkey開発入門
    - FireMonkeyアプリ作成手順
    - VCLとFireMonkeyとの違い
    - FireMonkeyの特徴的なコンポーネントや機能

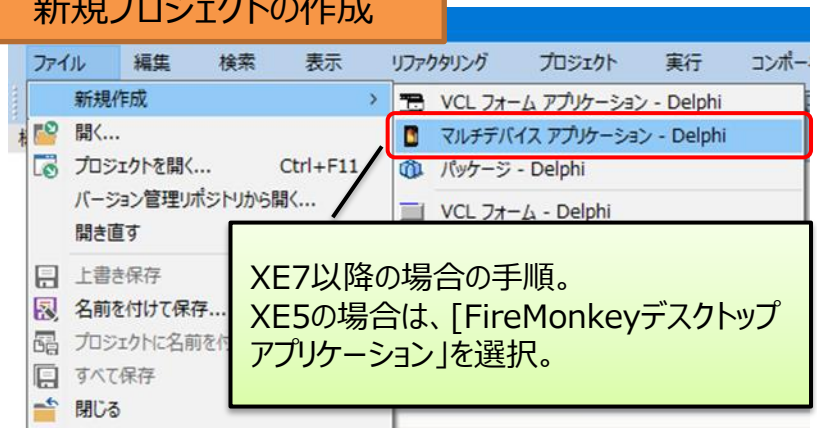
  - FireMonkey効果的な機能の活用
    - フォームレイアウト作成のコツ
    - ビジュアルスタイルの適用方法
    - Effectやアニメーション効果の使用方法

2．FireMonkey アプリ開発入門

FireMonkeyアプリ作成手順

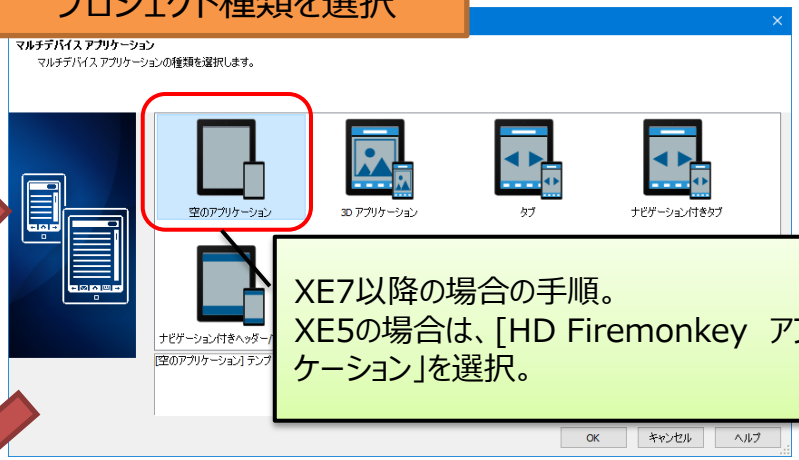
- FireMonkeyアプリケーション　プロジェクトの作成

新規プロジェクトの作成



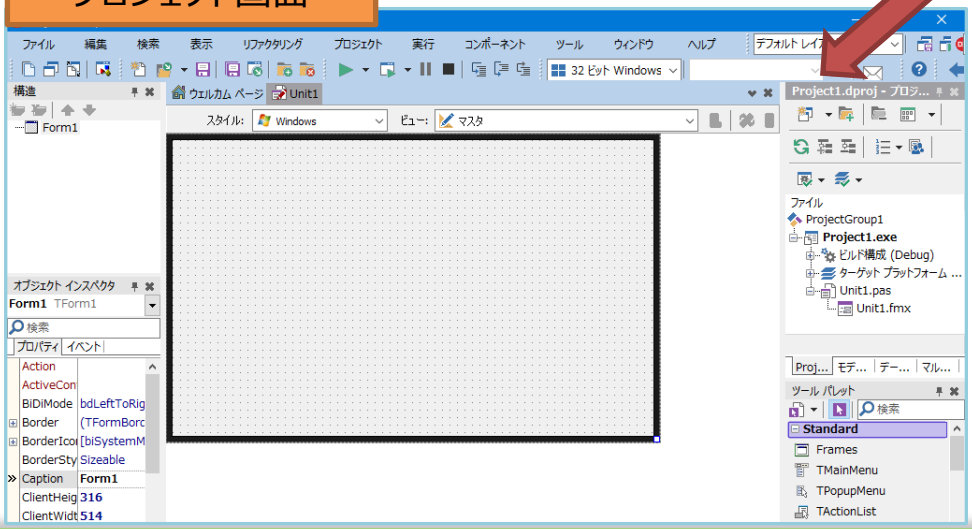
XE7以降の場合の手順。
XE5の場合は、[FireMonkeyデスクトップアプリケーション」を選択。

プロジェクト種類を選択

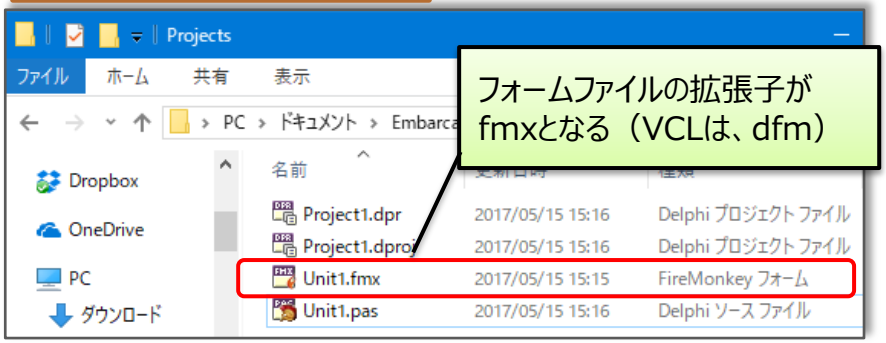


XE7以降の場合の手順。
XE5の場合は、[HD Firemonkey　アプリケーション」を選択。

プロジェクト画面



プロジェクトを保存



フォームファイルの拡張子がfmxとなる（VCLは、dfm）

FireMonkeyアプリ作成手順

- FireMonkeyアプリケーション　開発手順

コンポーネント

プロパティ、イベント

VCLと同様「ツールパレット」より選択。
VCLと同じ名前のものもあるが、互換性はない。

また、使用するプラットフォームにより選択可否が変化するものもある。
（例：iOS/Andoridでは TOpenDialogは選択不可）

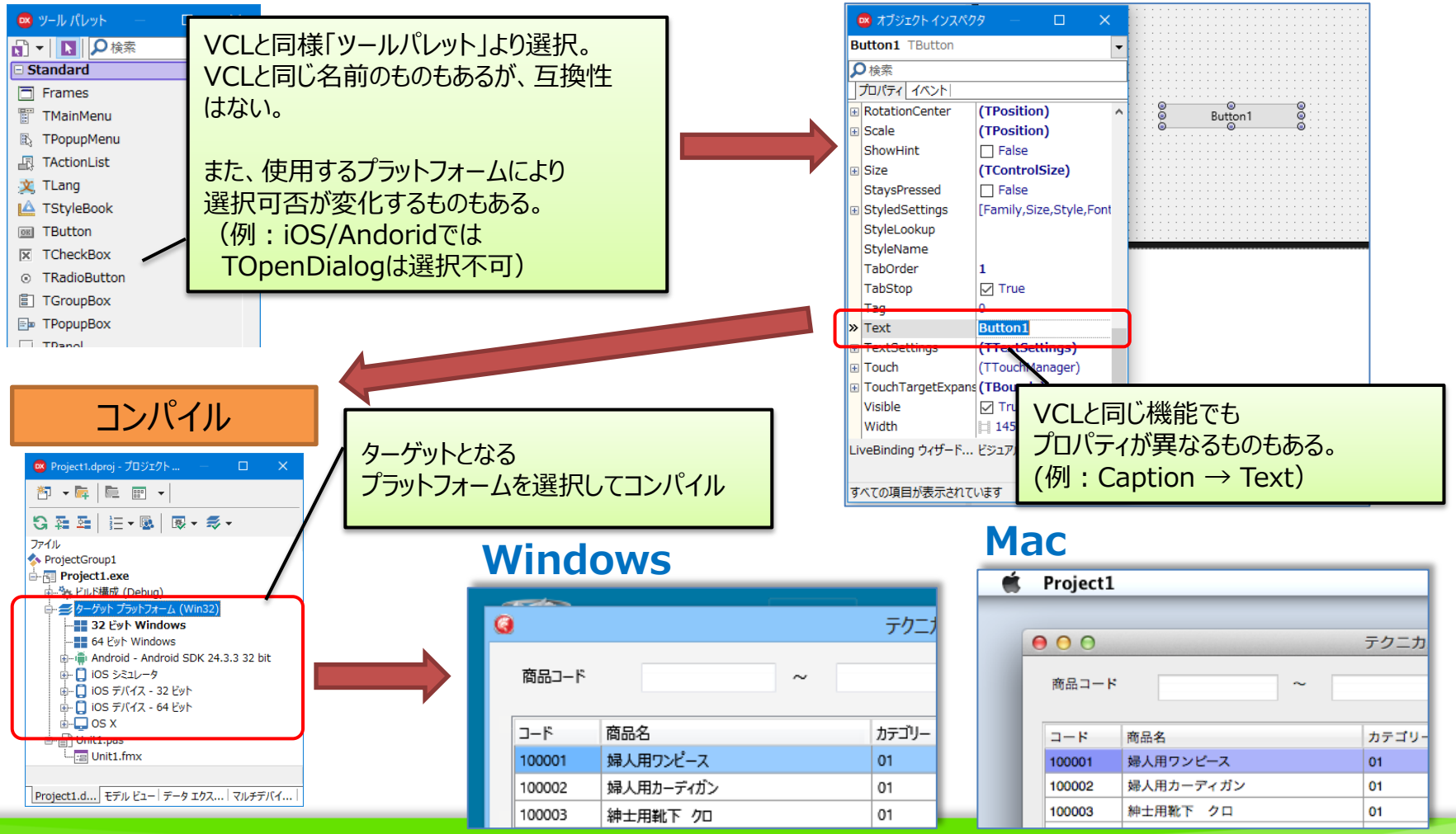
VCLと同じ機能でもプロパティが異なるものもある。
（例：Caption → Text）

コンパイル

ターゲットとなるプラットフォームを選択してコンパイル

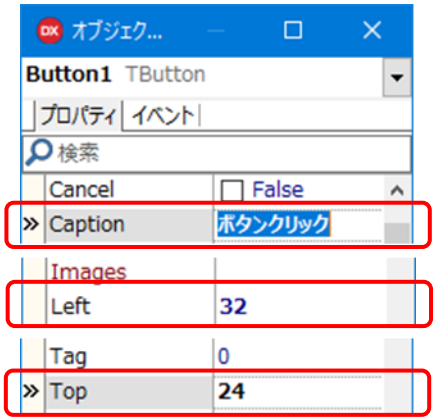
Windows

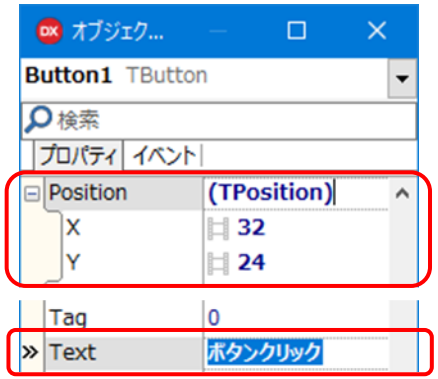
Mac

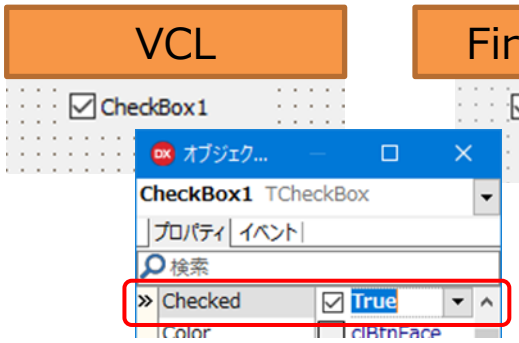


■ 一般的なVCLとFireMonkeyとの違い

- プロパティの違い
  - Caption → Text
  - Top, Left → Position
  - Checked → IsChecked





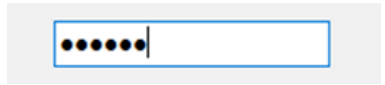


- TRagioGroupの置き換え
  - TGroupBox, TRadioButtonを使用
  - GroupNameプロパティで分類

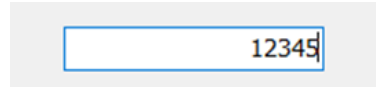
■ 一般的なVCLとFireMonkeyとの違い

- TEditの違い
  - 数値入力制御

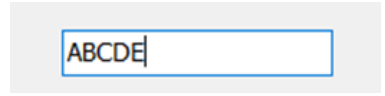
  - Password



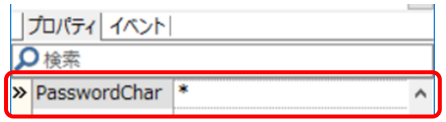
  - 配置

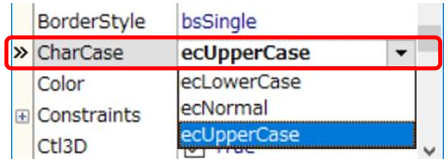


  - 小文字大文字変換

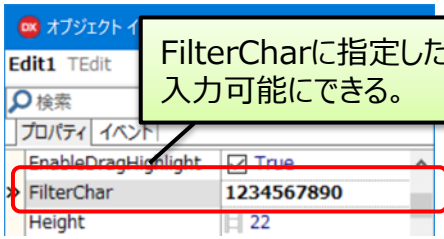


VCL

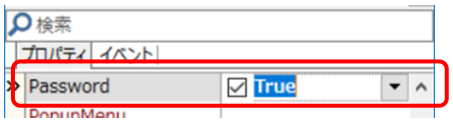


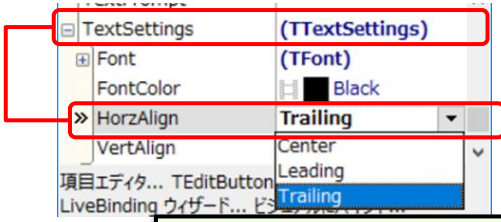


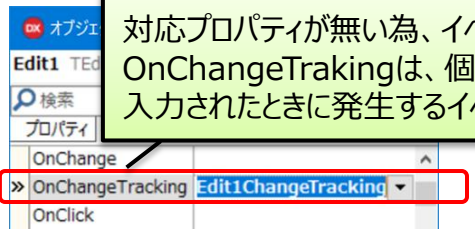
FireMonkey



FilterCharに指定した文字のみ
入力可能にできる。





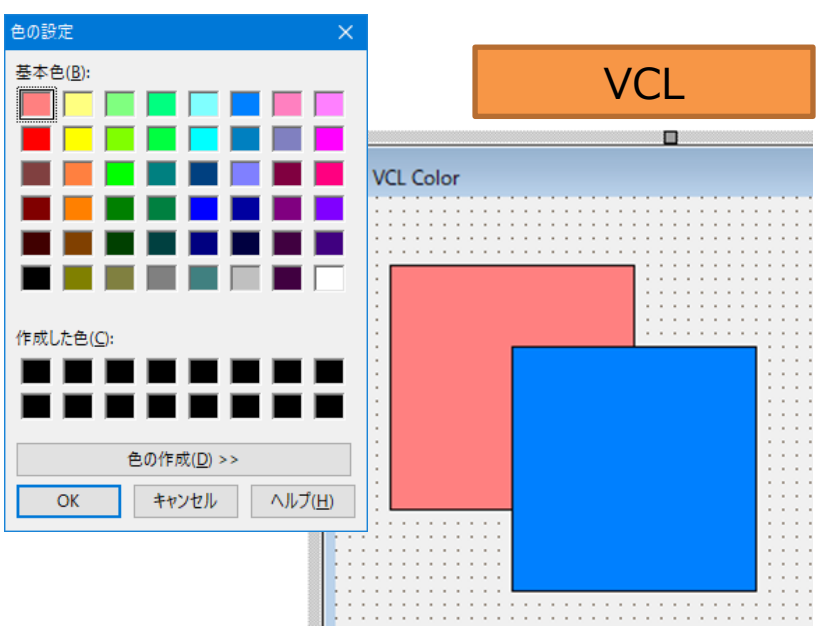


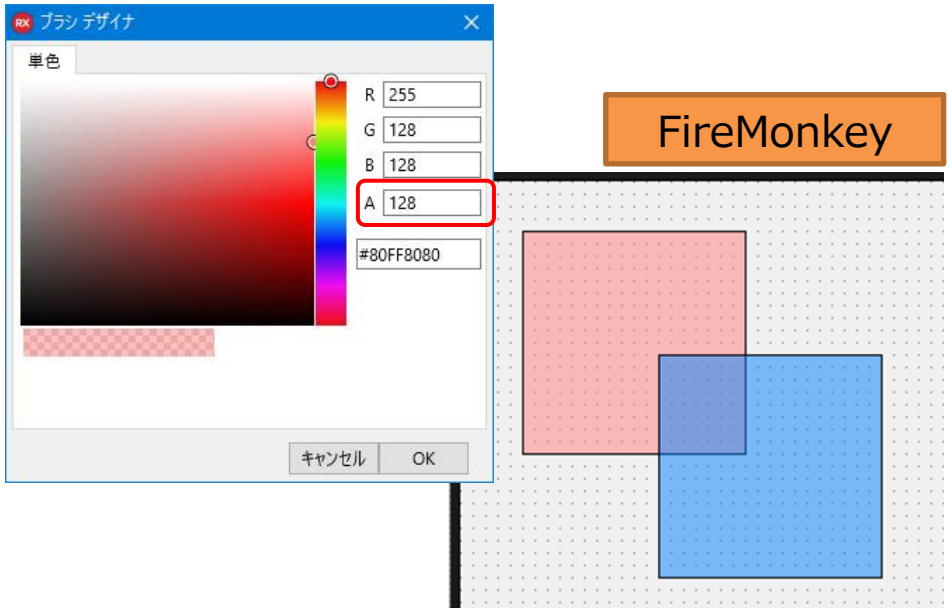
対応プロパティが無い為、イベントを作成。
OnChangeTrakingは、個々の文字が
入力されたときに発生するイベント。

```
procedure TForm1.Edit1ChangeTracking(Sender: TObject);
begin
  TEdit(Sender).Text := UpperCase(TEdit(Sender).Text);
end;
```

■ 一般的なVCLとFireMonkeyとの違い

- 色(Color)の扱い
  - VCL は R（赤）G（緑）B（青）で表現。（不透明）
  - FireMonkeyはRGBに加え A(Alpha：透過度）を持つ。





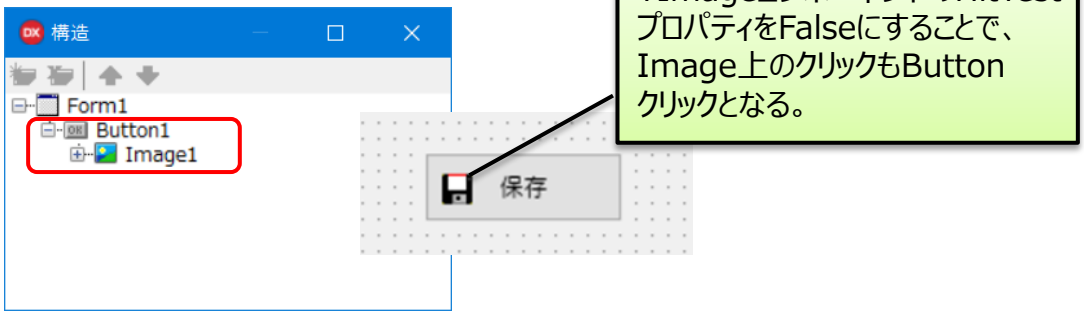
色定数（cl～）	意味
clRed	赤色
clBlue	青色
clWhite	白色

色定数（cla～）	意味
claRed	赤色
claBlue	青色
claWhite	白色

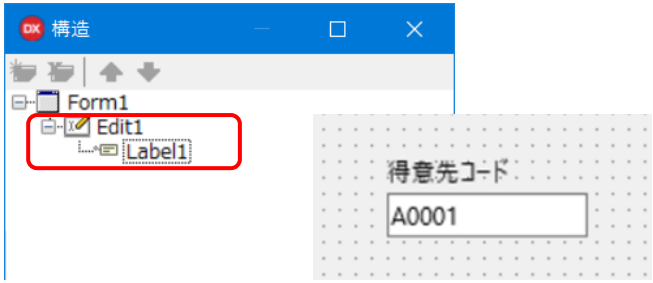
■ 一般的なVCLとFireMonkeyとの違い

- 親子関係
  - VCL は TPanelやTScrollBox等コンテナコンポーネントのみ。
  - FireMonkeyは全てのコンポーネントを組み合わせて親子関係にできる。

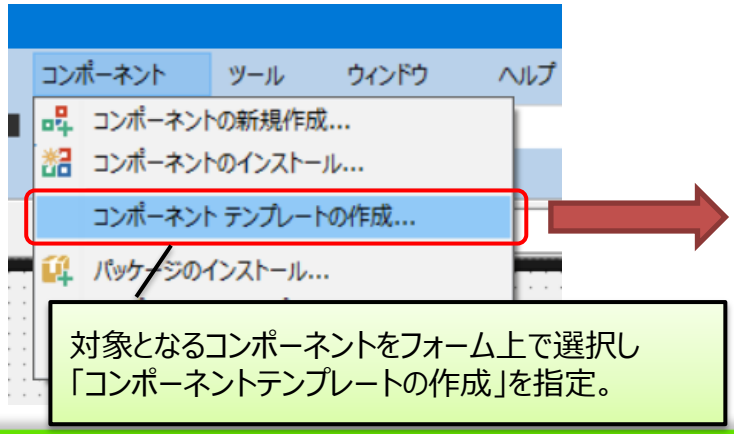
VCLのTBitBtn を TButton + TImage で表現

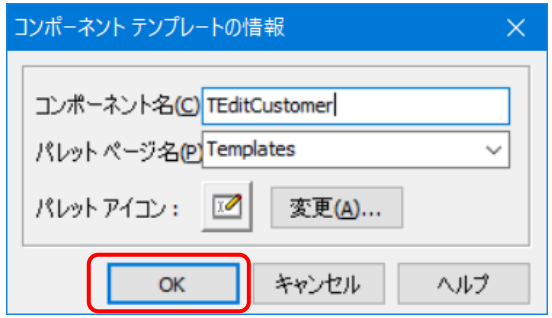


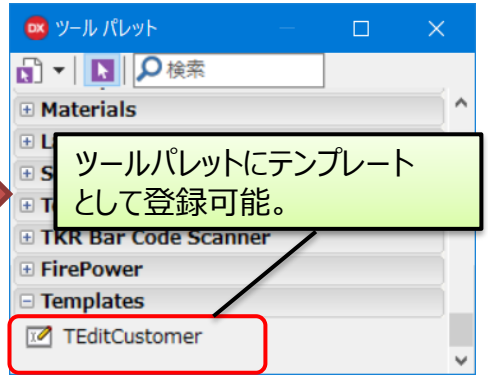
VCLのTLabeledEdit を TEdit+ TLabel で表現



- 頻出する組み合わせコンポーネントは、コンポーネントテンプレートに登録すると便利。



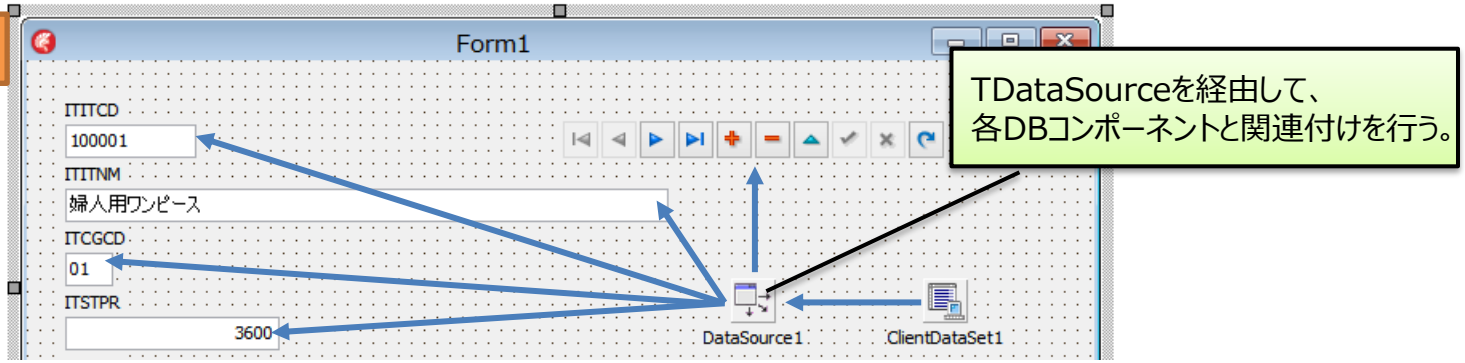




■ 一般的なVCLとFireMonkeyとの違い

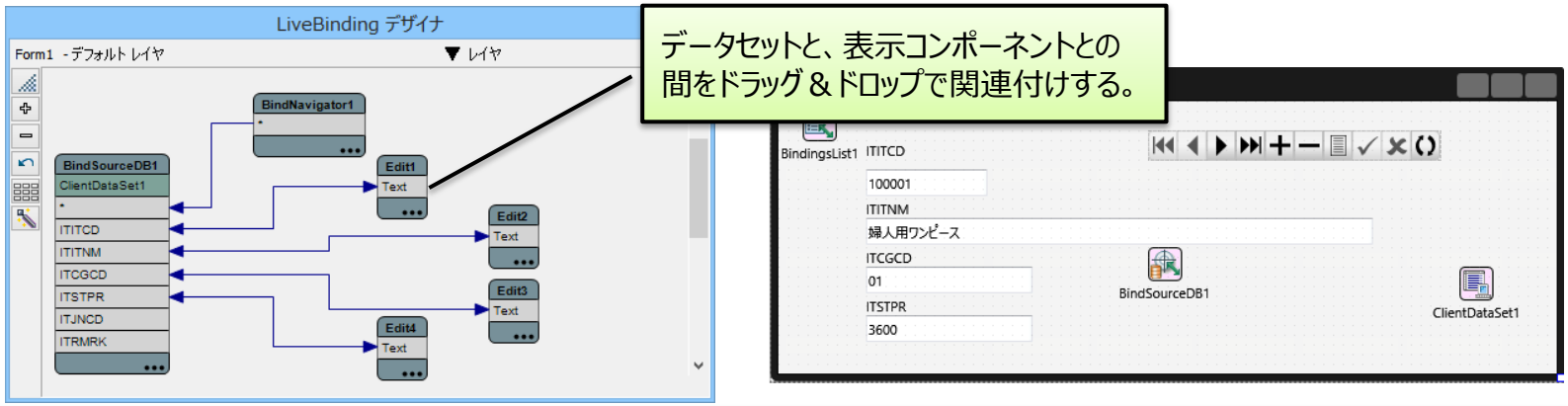
- データベース連結表示の取り扱い
  - VCLの場合、TDBEditやTDBGrid等のデータベース連結ビジュアルコンポーネントを使用

VCL



  - FireMonkeyには、データベース連結ビジュアルコンポーネントがない為、TEditやTStringGrid等のコンポーネントをLiveBindingを使用して連結

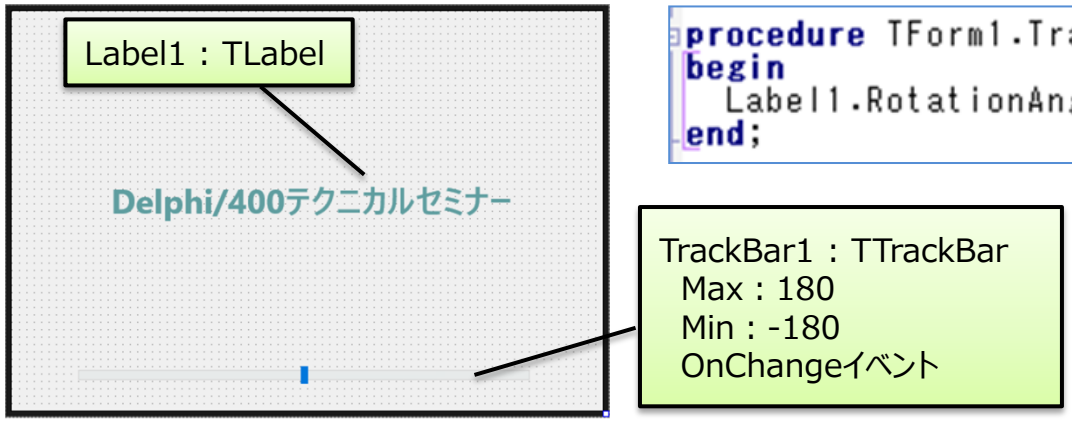
FireMonkey



FireMonkey コンポーネントの回転

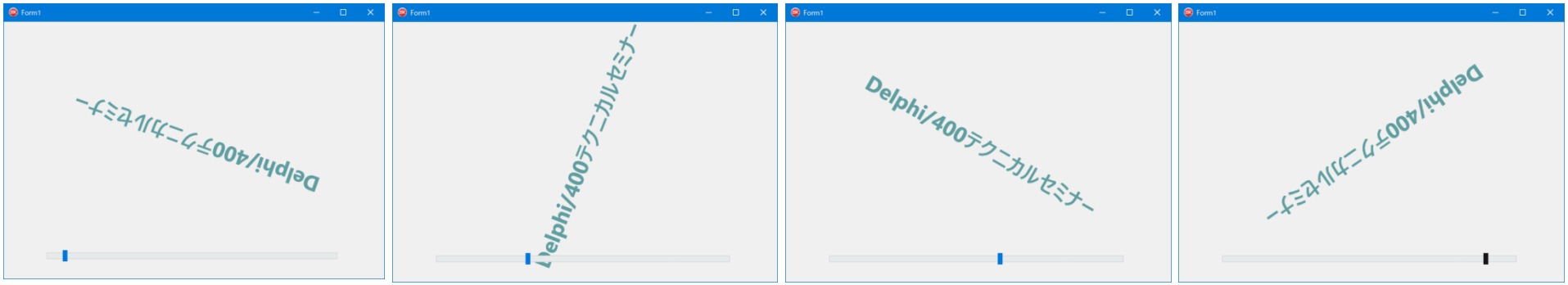
- コンポーネントは自由に回転できる
  - RotationAngleプロパティ：角度を指定
    正の数：時計回り、負の数：反時計回り

設計画面




```
procedure TForm1.TrackBar1Change(Sender: TObject);
begin
  Label1.RotationAngle := TrackBar1.Value;
end;
```

実行イメージ



FireMonkey 特有のコンポーネント

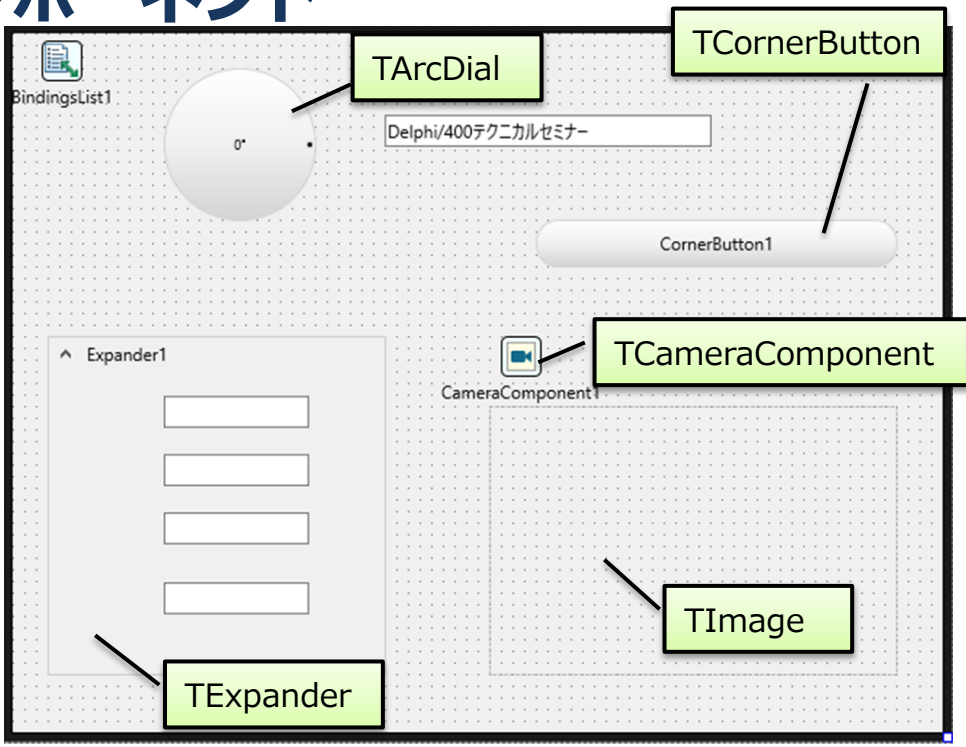
- TArcDial
  - つまみ型回転式ボタン
    - Value：角度（-180°～180°）

- TCornerButton
  - 角がカスタマイズできるボタン
    - CornerType：角の種類
    - XRadious / YRadious
      ：角の始点までの距離

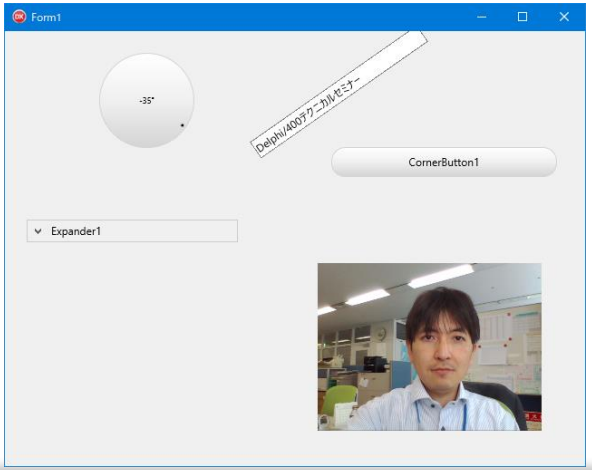
- TCameraComponent
  - カメラデバイスと対応する非ビジュアルコンポーネント
    - Active：カメラ有効/無効
    - SampleBufferToBitmap：ビットマップに出力
    - OnSampleBufforReady：出力準備完了

- TExpander
  - 展開・折り畳みのできるパネル

設計画面

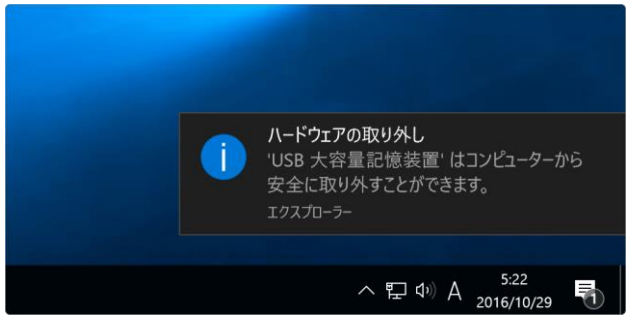


実行イメージ

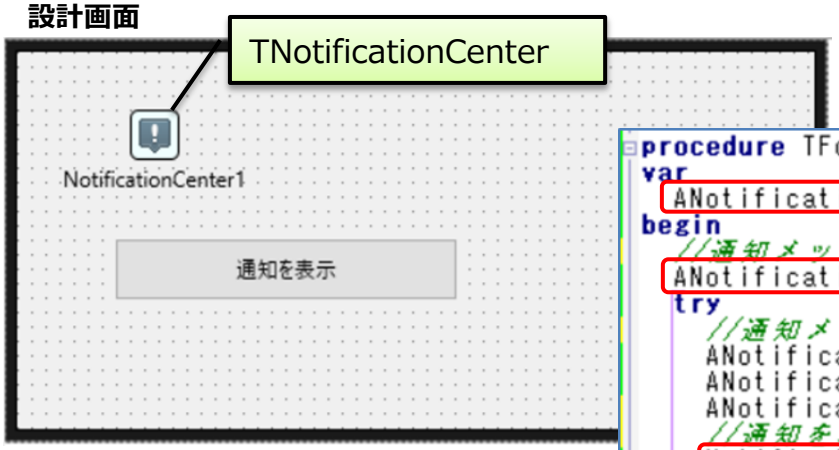


FireMonkey 通知メッセージ

- Windows10にあるアクションセンター
  - アプリが通知するメッセージを管理する機能。
  - Delphi/400 10 Seattleでは、Windows10に対応した通知メッセージの出力が可能。
    （この機能は、FireMokeyだけでなく、VCLでも使用可能）



設計画面



実行イメージ

通知メッセージを扱う変数

通知センターにメッセージ生成

通知メッセージ表示

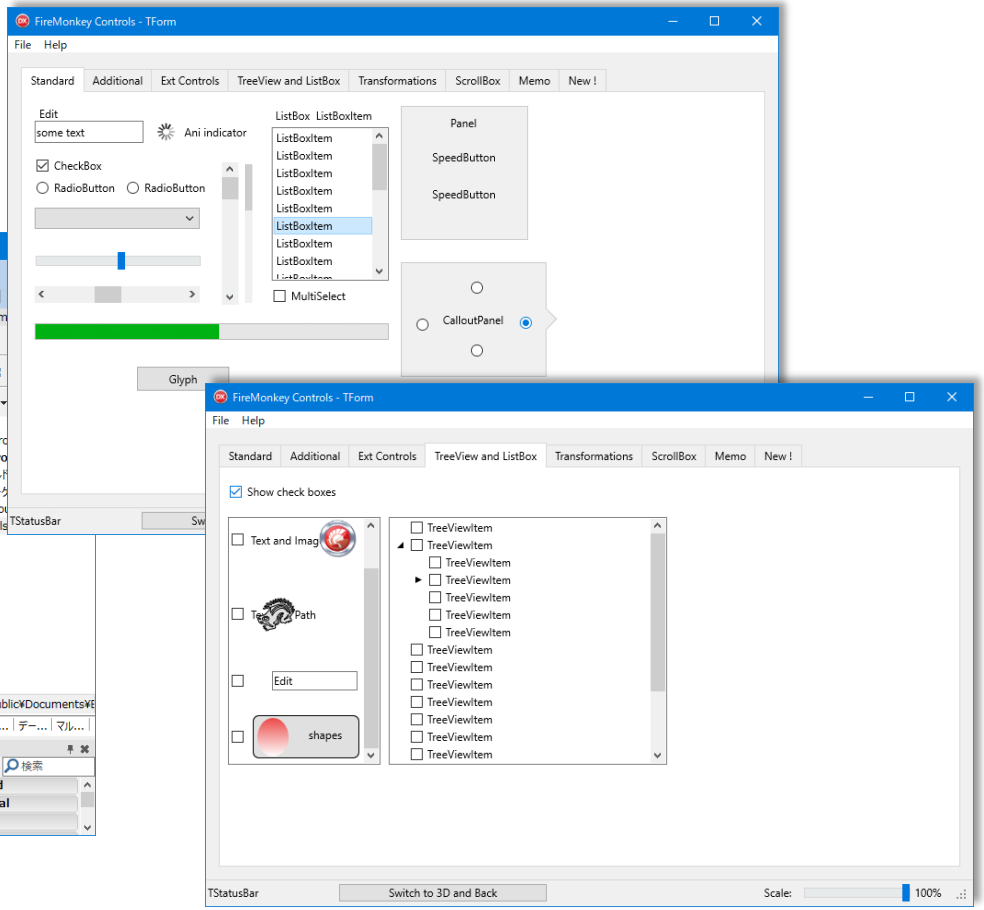
メッセージをクリックしたとき

```
procedure TForm1.Button1Click(Sender: TObject);
var
  ANotification: TNotification;
begin
  //通知メッセージのインスタンスを作成
  ANotification := NotificationCenter1.CreateNotification;
  try
    //通知メッセージのセット
    ANotification.Name := 'Delphi400TecSeminar';
    ANotification.Title := 'Delphi/400テクニカルセミナー';
    ANotification.AlertBody := '6月7日東京、6月13日大阪で開催';
    //通知を表示
    NotificationCenter1.PresentNotification(ANotification);
  finally
    //破棄
    ANotification.Free;
  end;
end;

procedure TForm1.NotificationCenter1ReceiveLocalNotification(Sender: TObject;
  ANotification: TNotification);
begin
  //通知をクリックしたときの処理
  if ANotification.Name = 'Delphi400TecSeminar' then
    ShowMessage('通知をクリックしました');
end;
```

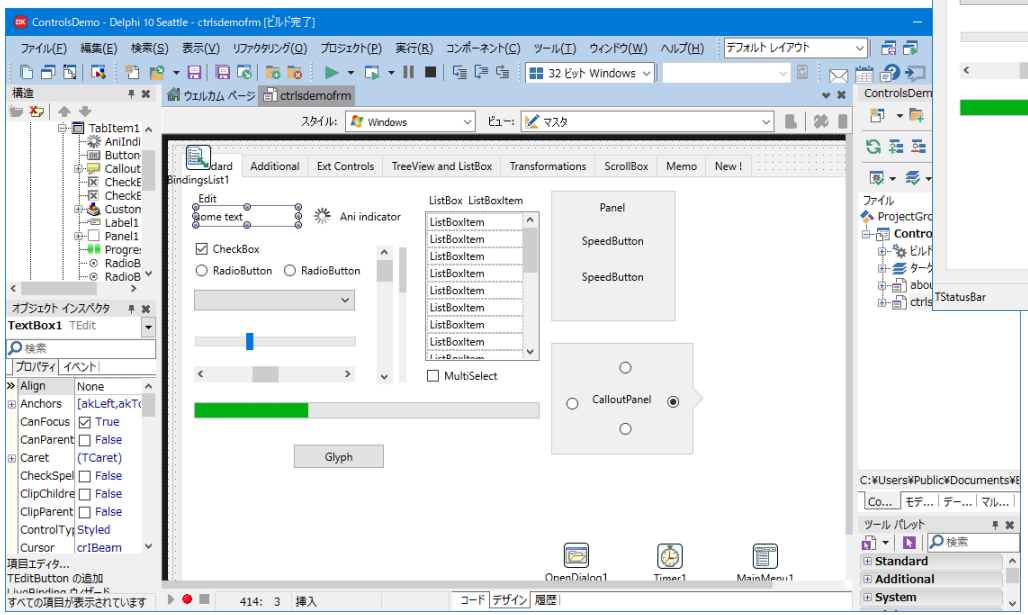
FireMonkey コンポーネントサンプル

- FireMonkeyコンポーネントのサンプルプログラム
  - C:¥Users¥Public¥Documents¥Embarcadero¥Studio¥17.0¥Samples¥Object Pascal¥Multi-Device Samples¥User Interface¥ControlsDesktop¥ControlsDemo.dpr

実行イメージ



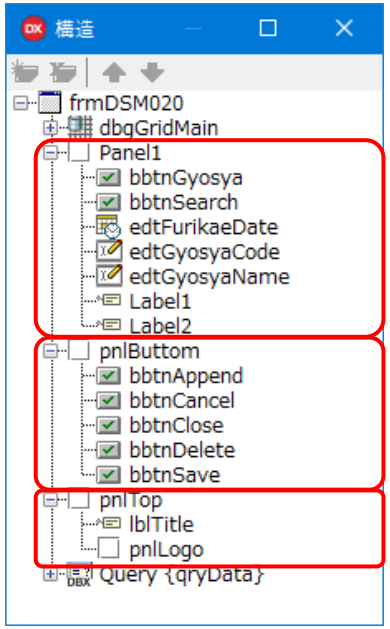
サンプルプログラム（設計画面）

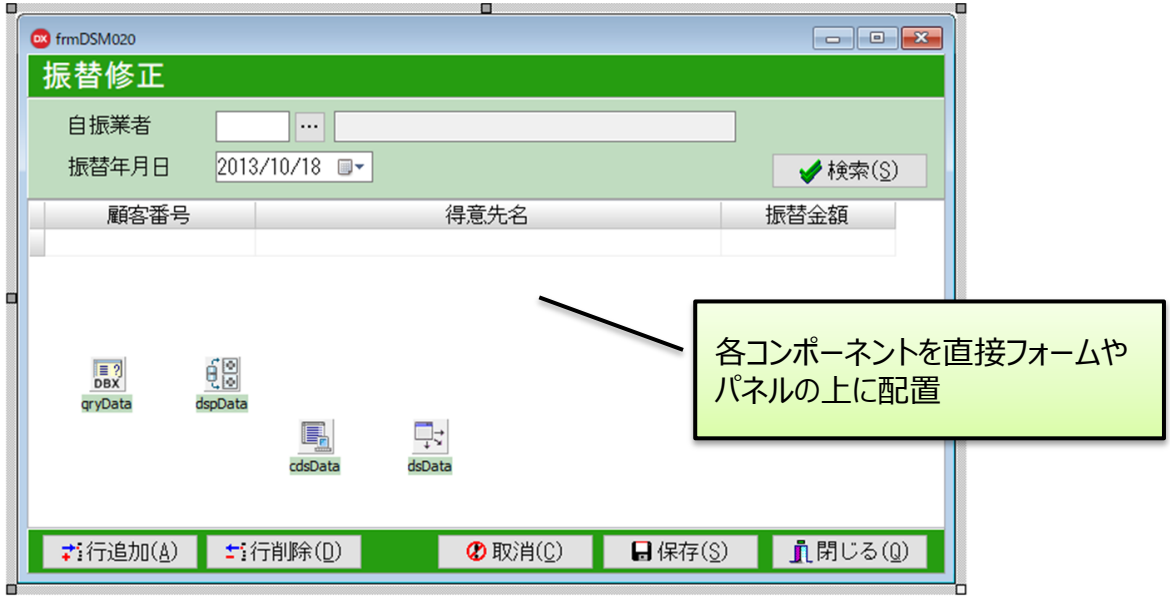


3．FireMonkey 効果的な機能の活用

■ フォームレイアウトのコツ

- VCLアプリにおけるフォームレイアウト（固定配置）
  - フォーム上に、直接コンポーネントを配置してレイアウトを作成
  - TPanel等を使ってエリアを作成（配置は、Alignプロパティを指定（alTop/alClient等））
  - 各Panelの下に、各コンポーネントを固定配置（Left, Top, Width, Heightプロパティを指定）
  - 右側にあるコンポーネントには、Anchorsプロパティ(akRight, akButtom)をセット

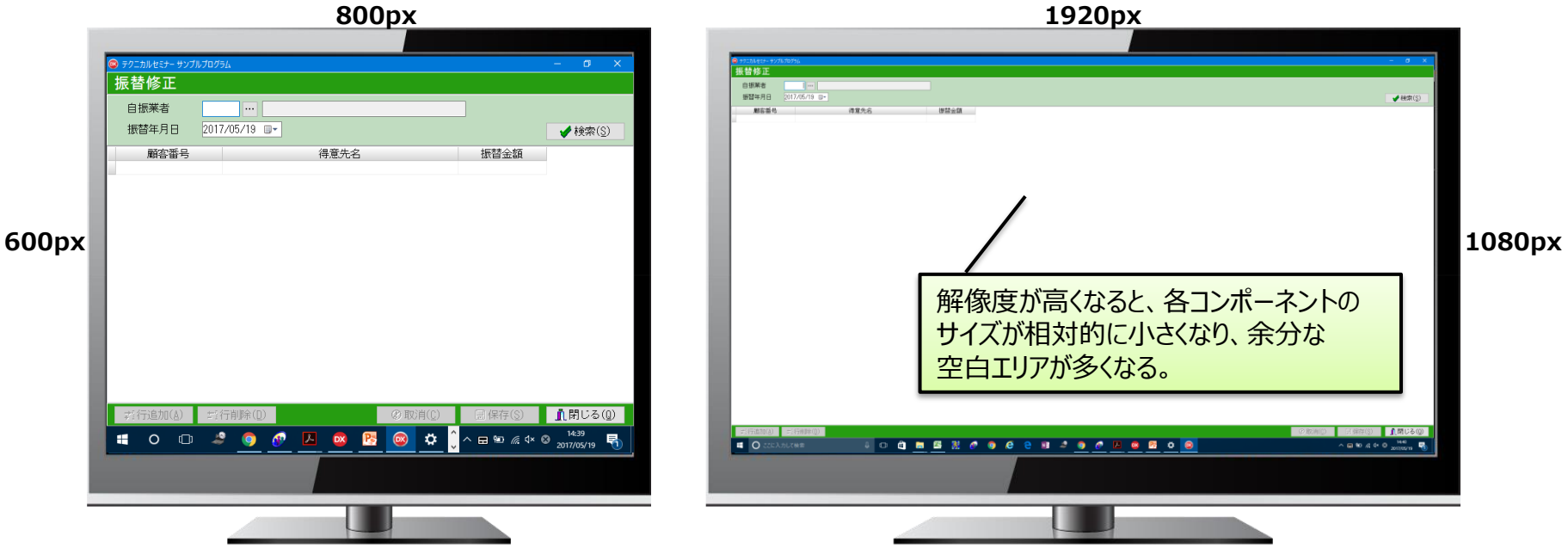




- FireMonkeyでも同様の固定配置で作成可能

■ フォームレイアウトのコツ

- 固定配置アプリケーションを実行（フォーム最大化）
  - フォントやコンポーネントのサイズがピクセル固定となる為、ディスプレイサイズや、解像度に応じた画面の柔軟な対応が難しい。

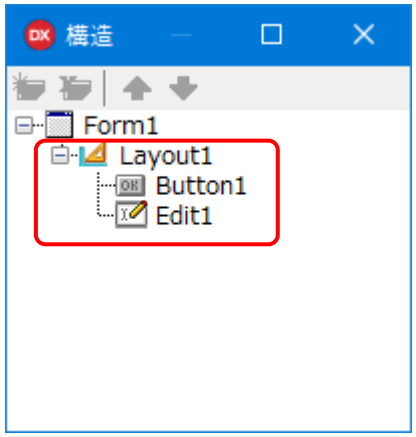


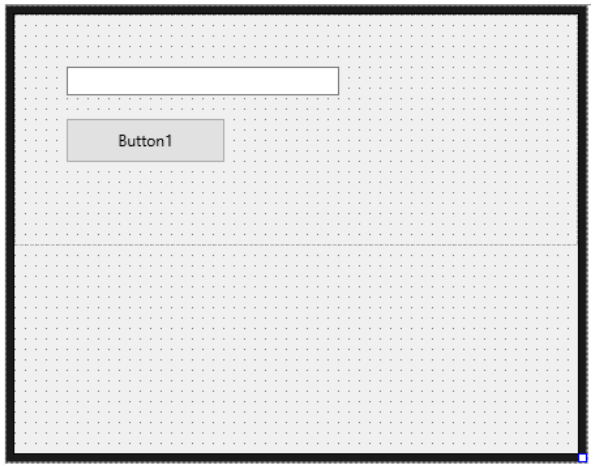
- マルチデバイス対応のFireMonkeyの場合、
  - Windowsだけでなく、RetinaディスプレイのMacや、スマートフォン、タブレット等あらゆるサイズ、解像度のマシンでの動作が要求されていく為、固定配置だと対応が困難になる可能性がある！

ピクセル固定としないレイアウト作成方法を紹介！

■ フォームレイアウトのコツ

- FireMonkeyでは、レイアウトコンポーネントを使用した配置が可能
  - TLayout 
    - 親となるコンポーネント（レイアウトコンテナ）
    - 自身は、表示されない。

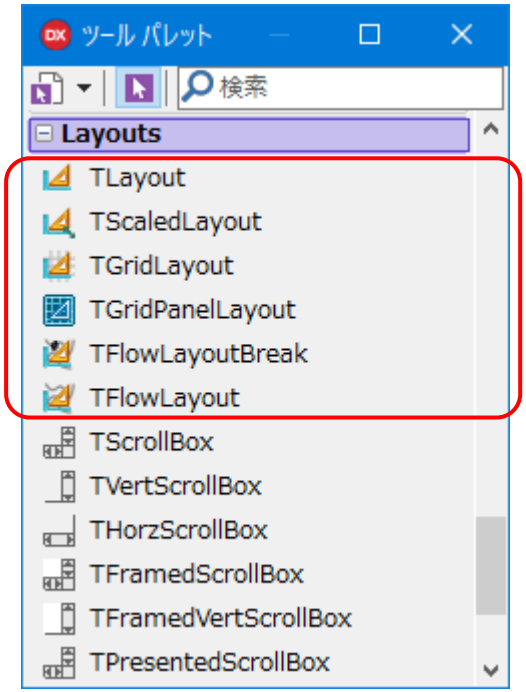




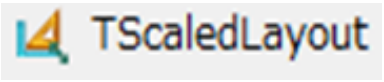
  - Layout自体のVisibleを変更すると、レイアウトの全てのコンポーネントの表示/非表示を設定可能

```
Layout1.Visible := False;
```

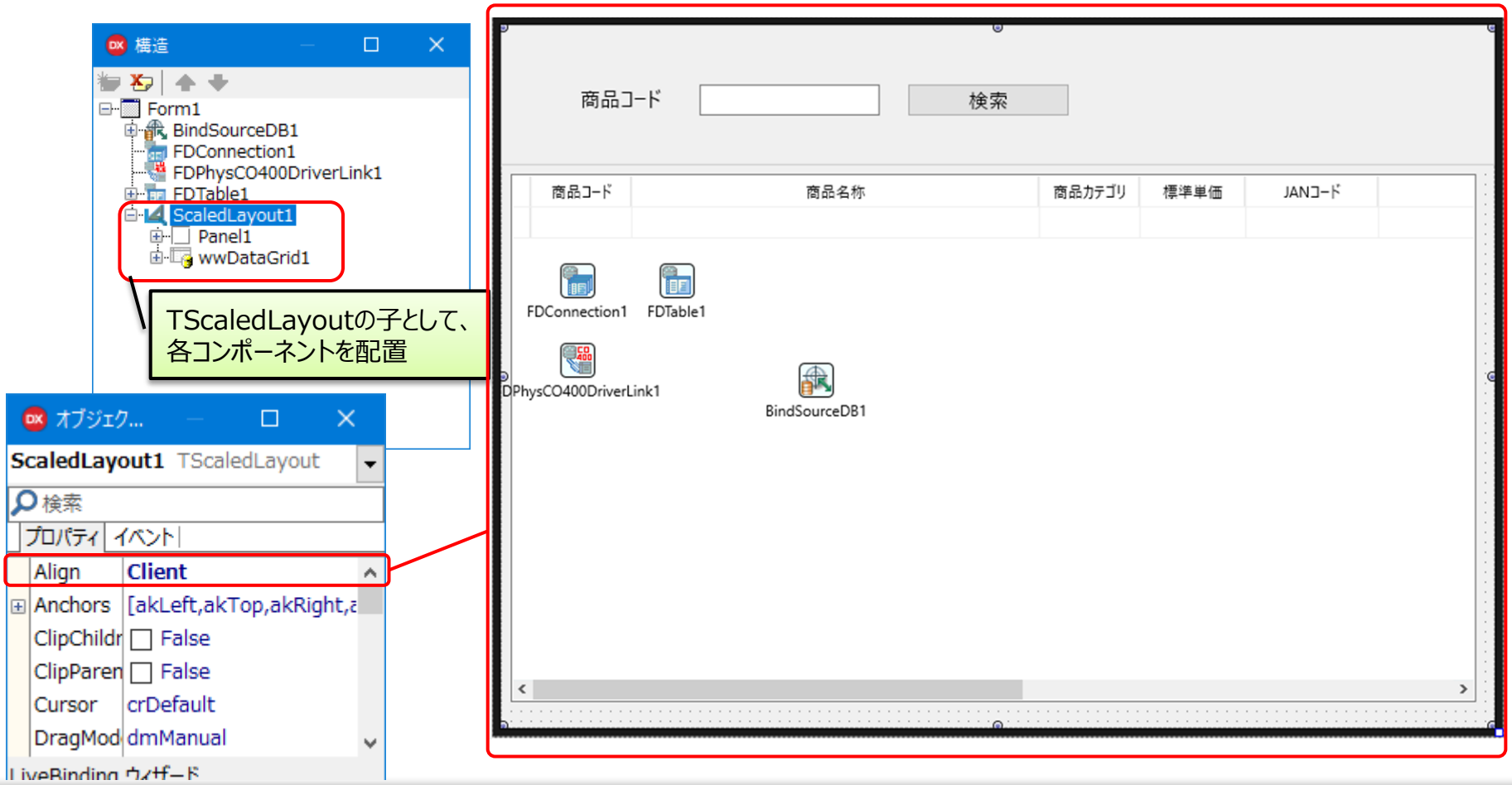
→ Button1, Edit1が共に非表示となる。



■ フォームレイアウトのコツ

- TScaledLayout 

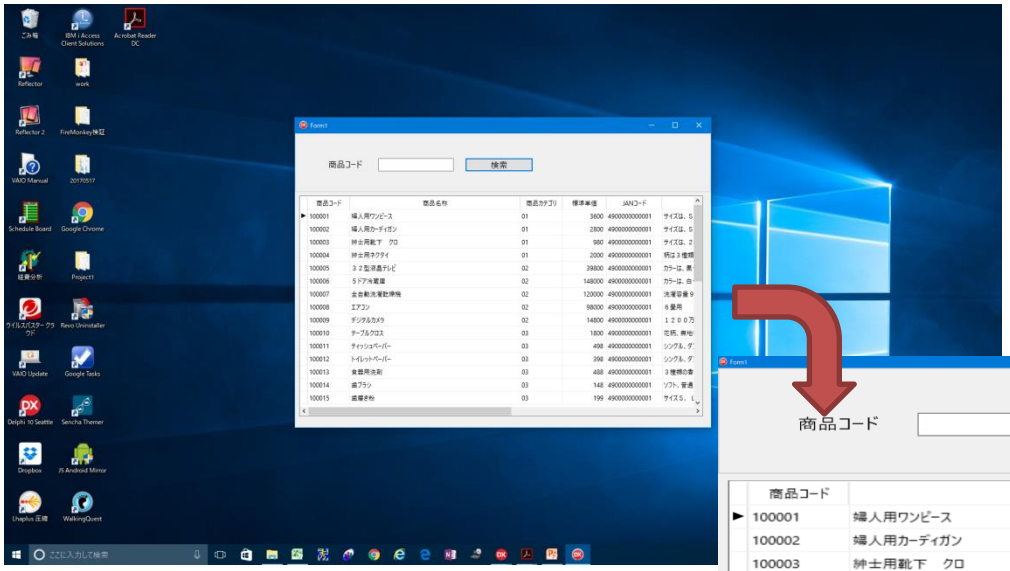
  - 縮小/拡大可能なレイアウト。 TScaledLayoutの子に配置したコンポーネントは、レイアウトのサイズ変更にあわせて縮尺が自動的に調整される。



■ フォームレイアウトのコツ

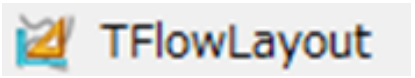
- 固定配置でも、高解像度ディスプレイへの対応が可能

実行イメージ

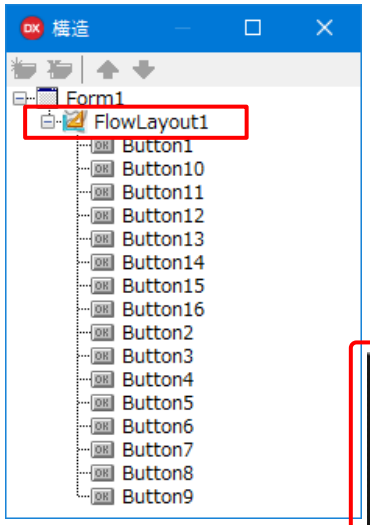


解像度や画面サイズが変わっても、破たんしない画面作成が可能！

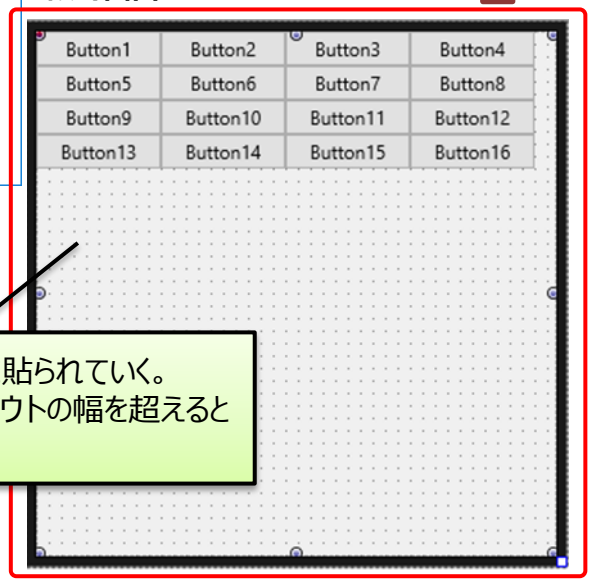
■ フォームレイアウトのコツ

- TFlowLayout 

  - 段落内の単語と同じように子コンポーネントを整列させるレイアウト

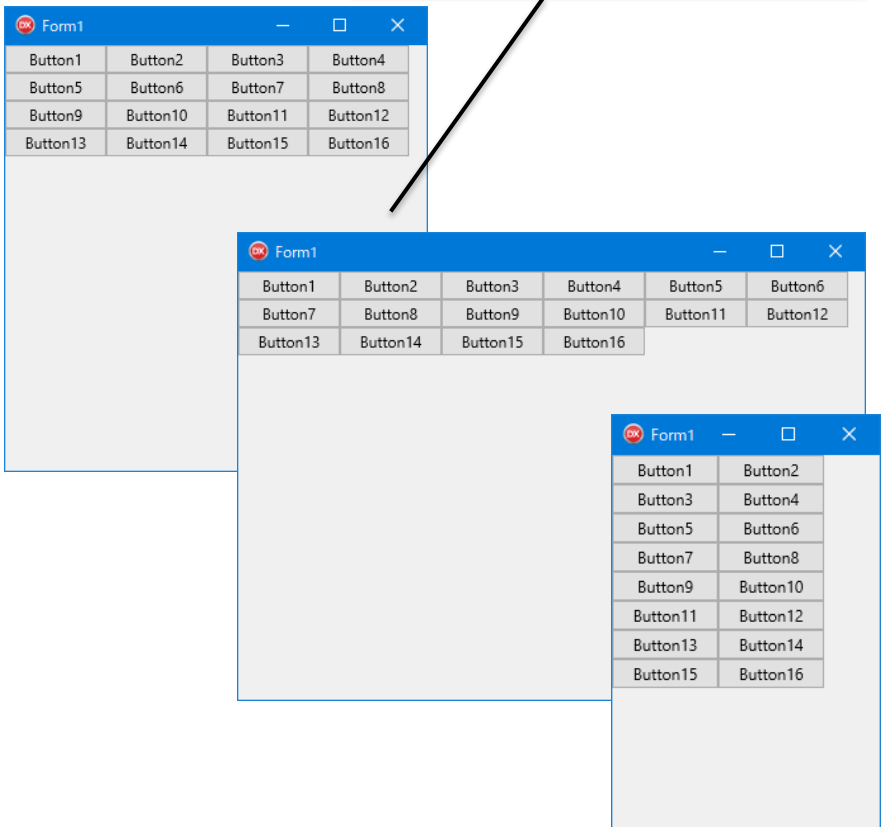


設計画面



部品は、左はしから順番に貼られていく。
部品の幅の合計が、レイアウトの幅を超えると
自動的に改行される。

実行イメージ



画面幅の変更に応じて、
部品の配置が自動的に調整される。

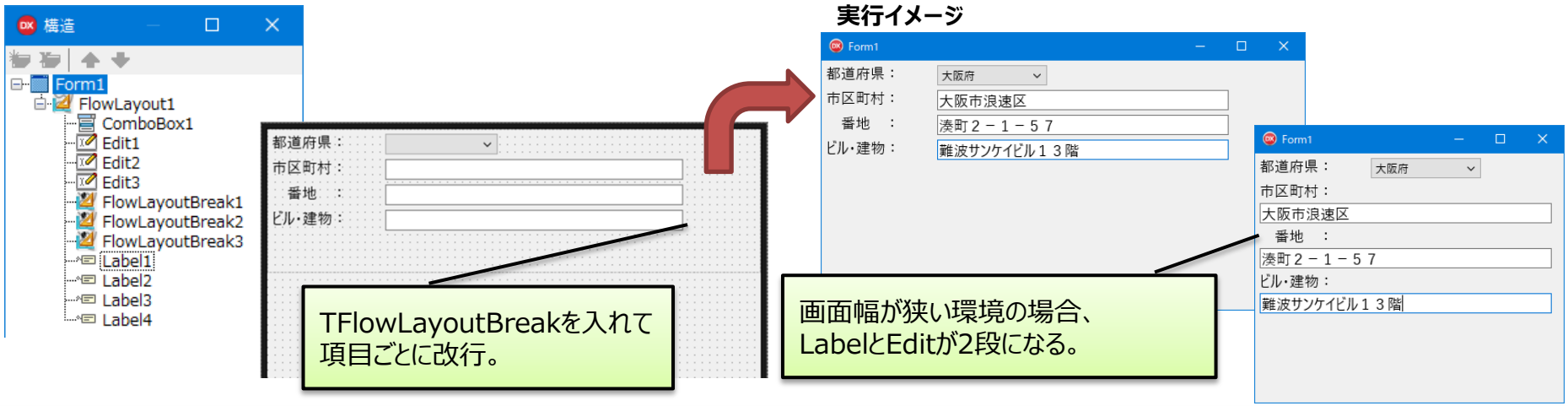
- TFlowLayoutBreak を途中に入れると任意の位置で改行可能。 

■ フォームレイアウトのコツ

- TFlowLayout
  - 主なプロパティ

ﾌﾟﾛﾊﾟﾃｨ	機能
FlowDirection	コンポーネントを整列する方向（左→右 or 右→左）
HorizontalGap	コンポーネント間の距離
VerticalGap	行間の距離
Justify	各行の配置（左寄せ、右寄せ、中央寄せ、等間隔）
JustifyLastLine	最終行の配置

  - 使用例

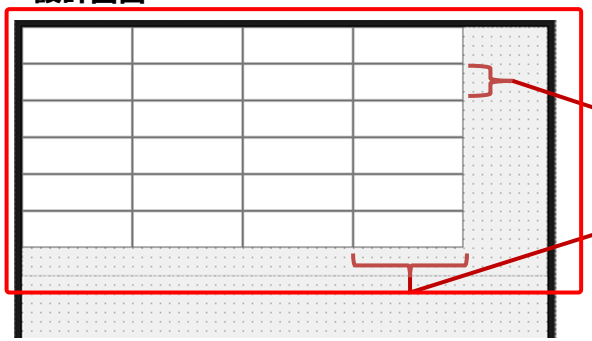


■ フォームレイアウトのコツ

- TGridLayout

  - 各セルのサイズが等しいグリッド内に子コンポーネントを整列させるレイアウト

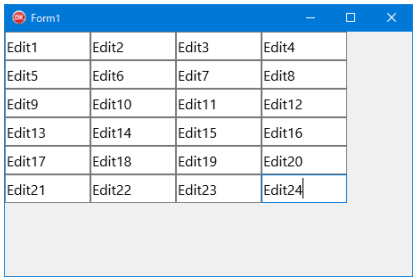
設計画面



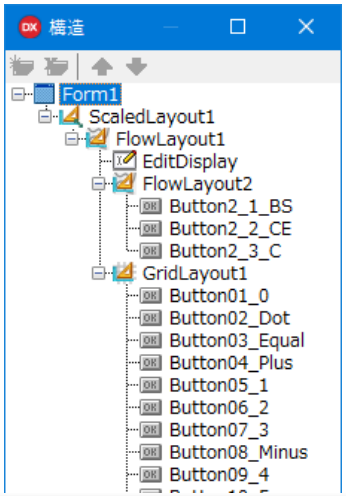
  - 主なプロパティ

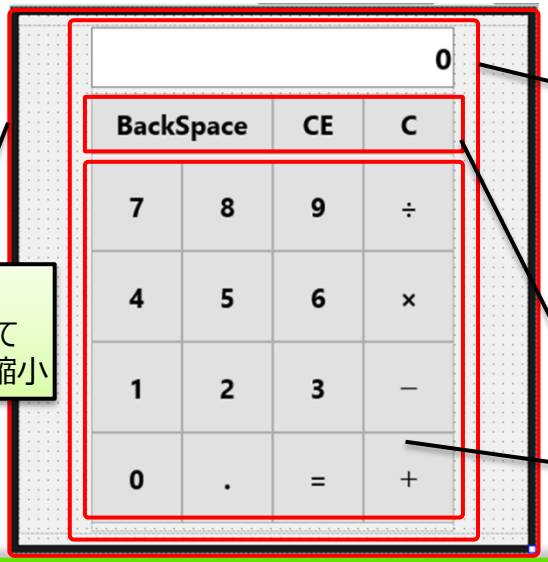
ﾌﾟﾛﾊﾟﾃｨ	機能
ItemHeight	1つのセルの高さ
ItemWidth	1つのセルの幅
Orientation	セルを並べる方向 （左→右 or 上→下）

実行イメージ



- レイアウト組み合わせ例





ScaledLayout1
フォームサイズに連動して
コンポーネントを拡大/縮小

FlowLayout1
以下のコンポーネントを整列配置
EditDisplay
FlowLayout2
GridLayout1

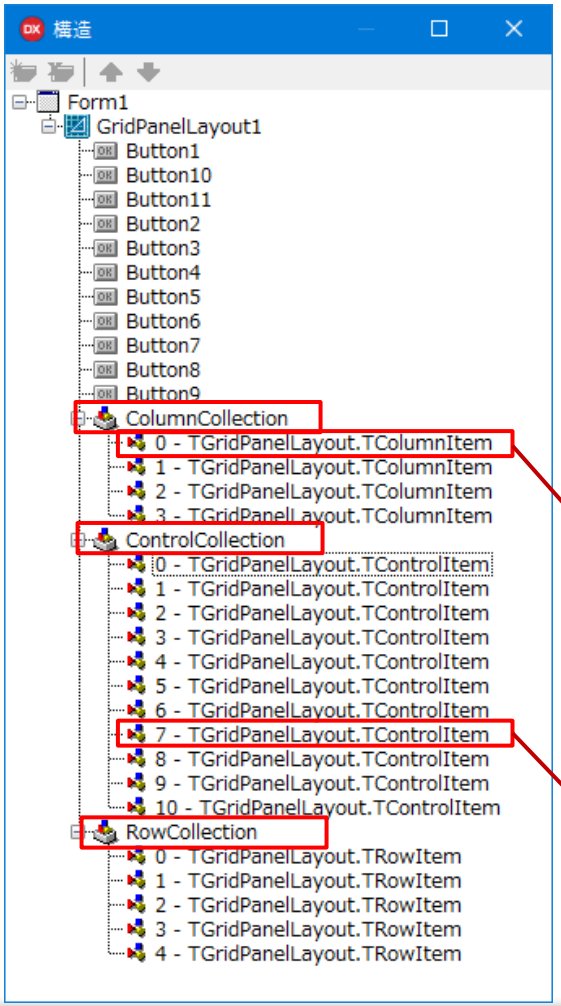
FlowLayout2
サイズの異なるボタンを整列配置

GridLayout1
各ボタン（セル）の高さと幅を指定して配置

■ フォームレイアウトのコツ

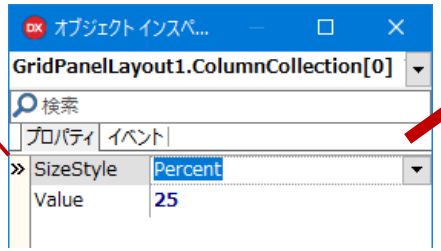
- TGridPanelLayout 

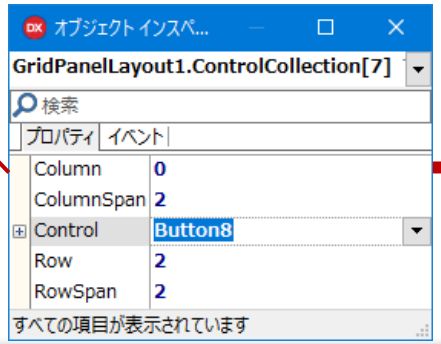
  - 各コンポーネントがグリッドパネル上のセル内に配置されるレイアウト

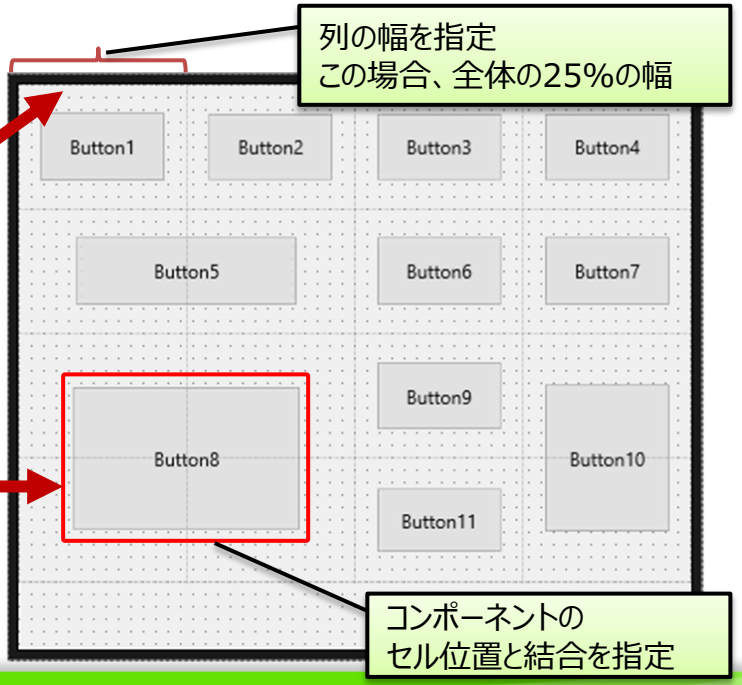


主な
プロパティ

ﾌﾟﾛﾊﾟﾃｨ	機能
ColumnCollection	グリッドの列情報を保持するコレクション
RowCollection	グリッドの行情報を保持するコレクション
ControlCollection	グリッドに配置されるコンポーネントのコレクション

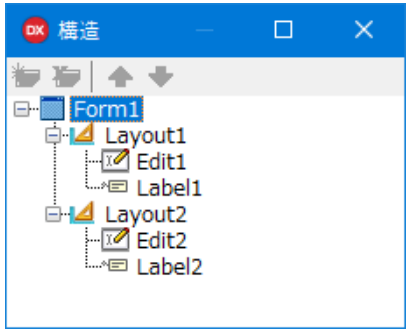






フォームレイアウトのコツ

- コンポーネントの配置
  - 固定配置によるコンポーネント貼り付け
    - Position
    - Width
    - Height



Edit1
Position.X , Y = 104, 14
Width = 525
Height = 22

設計画面

氏名：

Label1
Position.X , Y = 8, 16
Width = 65
Height = 17

  - 配置を定義するプロパティを設定して貼り付け
    - Align
    - Padding
    - Margins

Label2
Align = Left
Width = 65

Edit2
Align = Client
Margins=(
Buttom=0, Left=30
Right = 0, Top = 0)

設計画面

氏名：

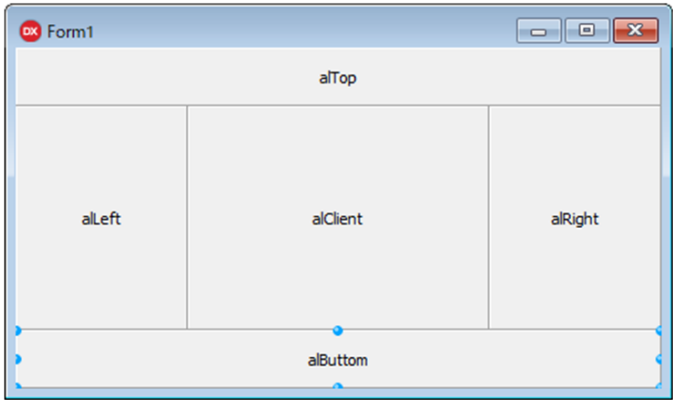
氏名：

LayOut2
Padding=(
Buttom=13, Left = 10
Right = 10, Top = 13)

■ フォームレイアウトのコツ

- Alignプロパティ
  - コンポーネントの整列オプション（上、左、クライアント等）を設定
  - VCLにもAlignプロパティがあり、下記指定が可能（alTop, alLeft, alRight, alButtom, alClient）
  - FireMonkeyの場合、さらに多彩な指定が可能

VCLのAlignプロパティ　設定例

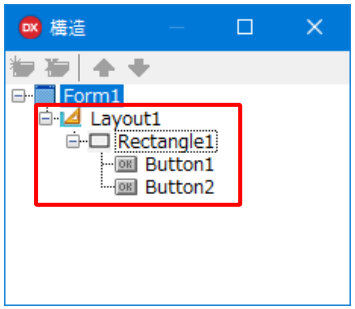


Alignプロパティ　設定値一覧

設定値	機能
Top, Left, Right, Buttom	親コンポーネントの1辺に寄せて、空いている領域いっぱいに広げて表示（VCLと同じ）
Cleint	空いている領域を埋め尽くして表示（VCLと同じ）
Fit, FitLeft, FitRight	親項目の中で最大化（コンポーネントの縦横比は維持）
Vertical, VertCenter, Horizontal, HorzCenter	幅あるいは高さの一方向をリサイズ
Center	親コンポーネントの中央に表示
Containts	親コンポーネント全体に表示
Scale	親コンポーネントのサイズにあわせてサイズが変更

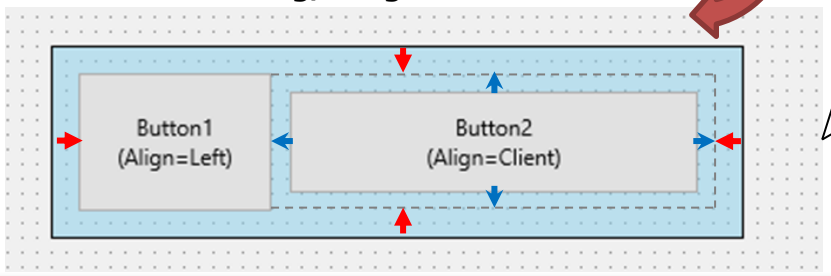
■ フォームレイアウトのコツ

- Margins
  - コンポーネント間の余白設定
    - Top/Left/Right/Buttom：距離をピクセル単位で指定
- Padding
  - 親コンポーネントと子コンポーネントまでの距離
    - Top/Left/Right/Buttom：距離をピクセル単位で指定



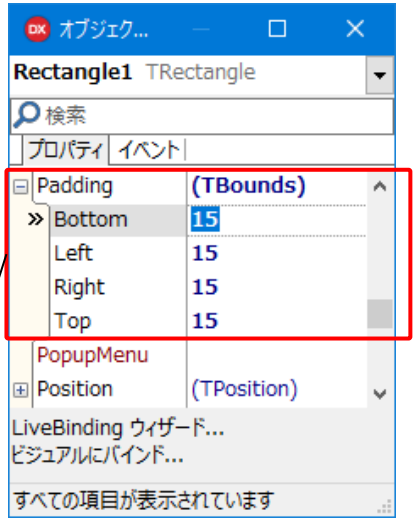
設計画面　（Alignのみ指定）

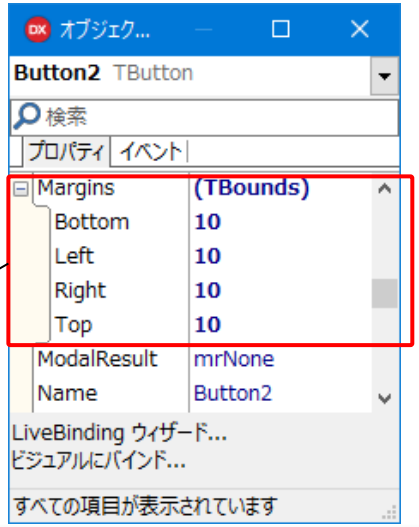
設計画面　（Padding/Marginを追加）



Padding

Margins





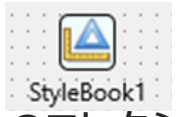
■ スタイルの適用

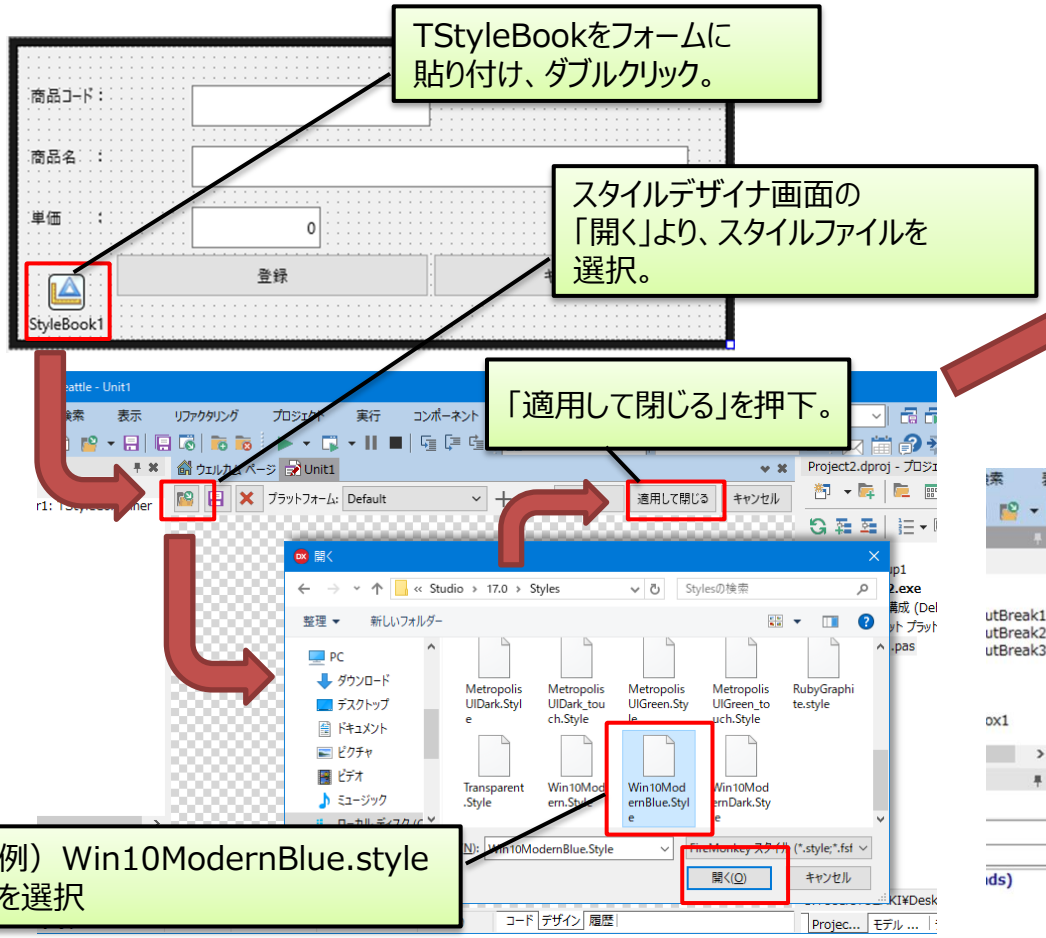
- ルックアンドフィールのカスタマイズ
  - VCLでは、色やフォントのカスタマイズは、直接コンポーネントのFontプロパティや、Colorプロパティを編集することが多い。

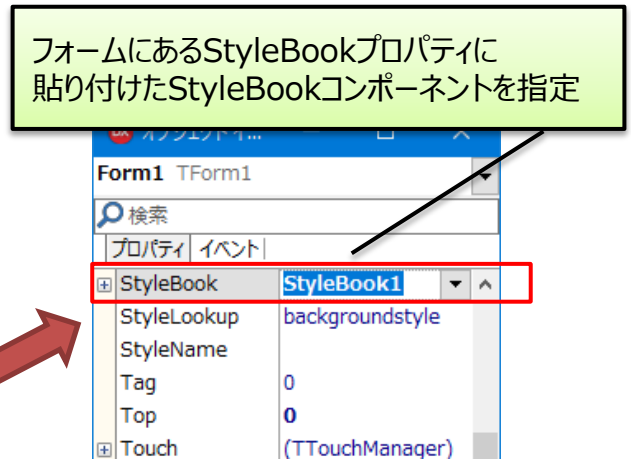
  - FireMonkeyでは、各コンポーネントの色やフォント等の情報をスタイルとして定義して、スタイルをコンポーネントに割り当てる手法をとる。
    （HTMLでいうところの、CSS（Cascading Style Sheets）のようなイメージ）

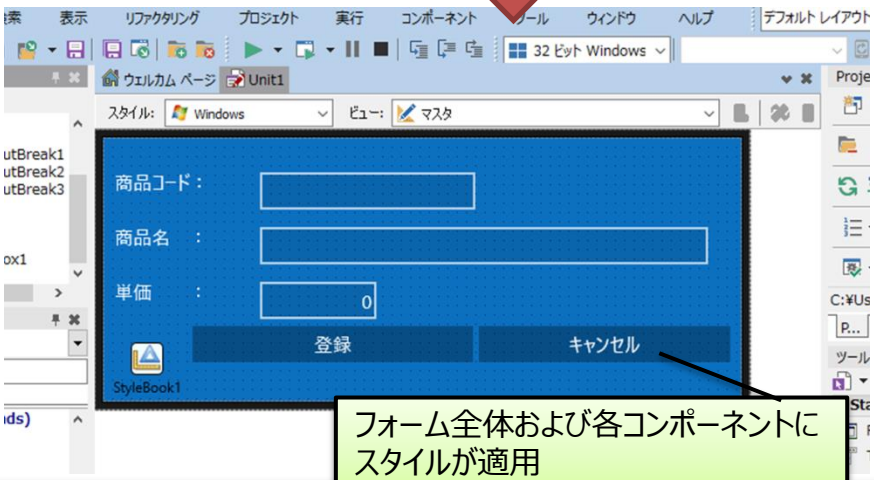
■ スタイルの適用

- TStyleBook
  - フォームのスタイルのコレクションを保持





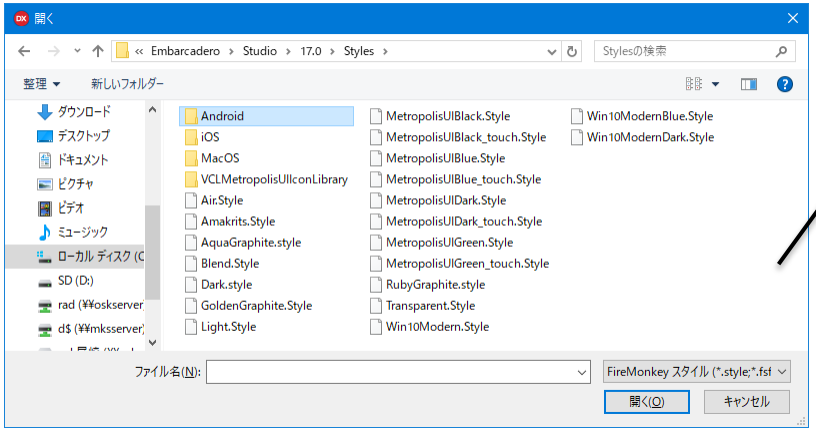




■ スタイルの適用

- TStyleBook
  - デフォルトのスタイルファイルは、下記フォルダに収録（Delphi/400 10 Seattleの場合）

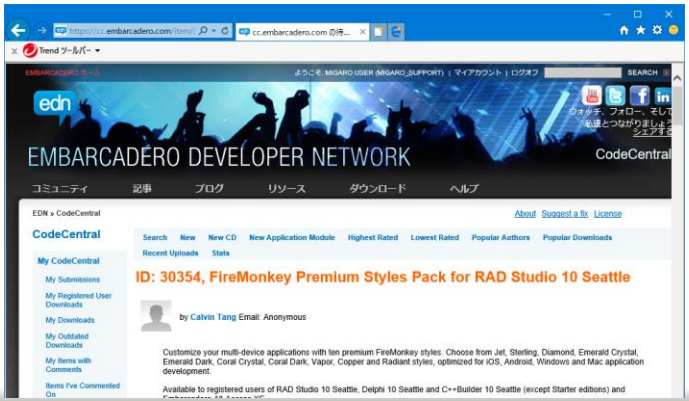
    C:¥Users¥Public¥Documents¥Embarcadero¥Studio¥17.0¥Styles¥



拡張子　.style

Delphi/400 10 Seattleの場合、
20種類のスタイルを収録

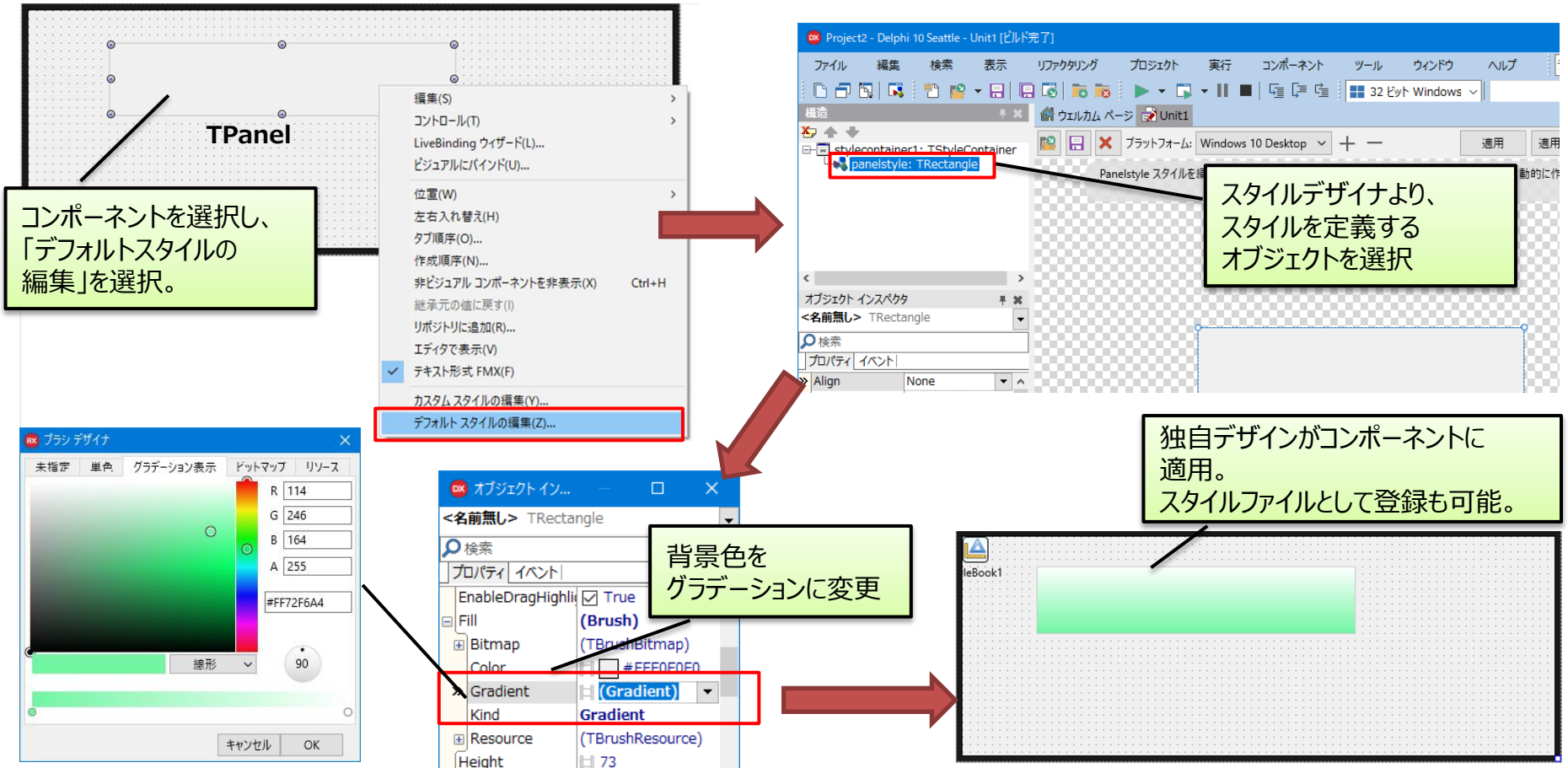
  - エンバカデロ登録ユーザーダウンロードページより、追加のスタイルもダウンロード可能



https://cc.embarcadero.com/item/30354

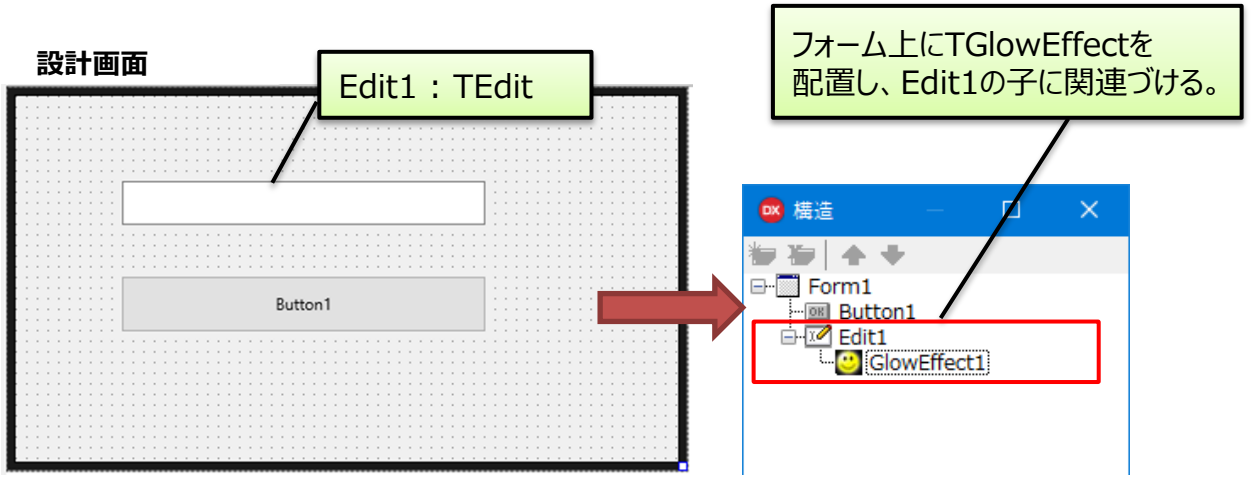
■ スタイルの適用

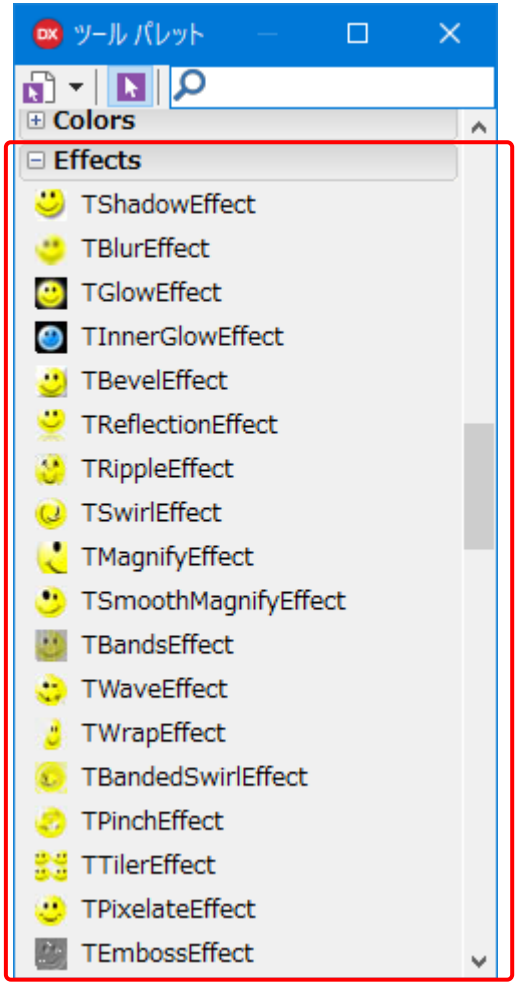
- スタイルのカスタマイズ
  - 選択したスタイルに対し、スタイルデザイナを使用して追加カスタマイズが可能
  - コンポーネント単位に独自のスタイルを設定することも可能



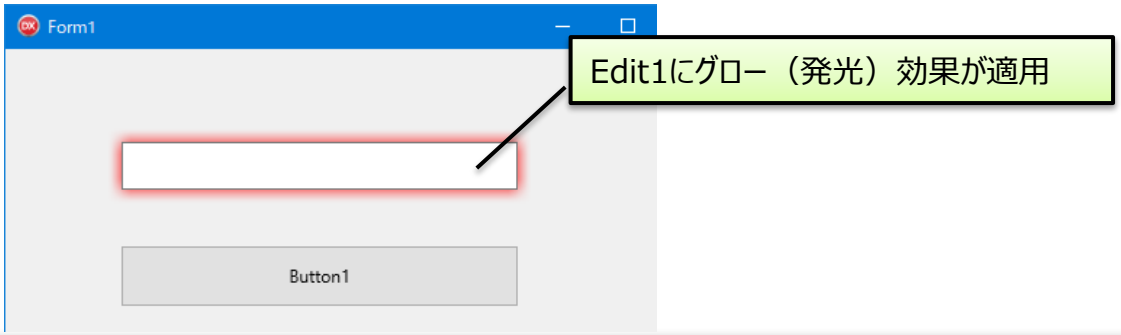
Effect効果

- Effect　：　コンポーネントに対する画像効果
  - コンポーネントの外観に画像効果を加える機能
  - Effectsカテゴリに色々な効果が登録されている。



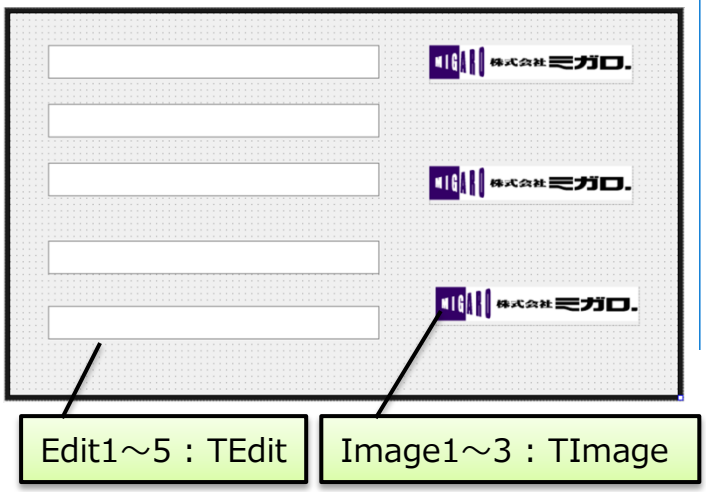


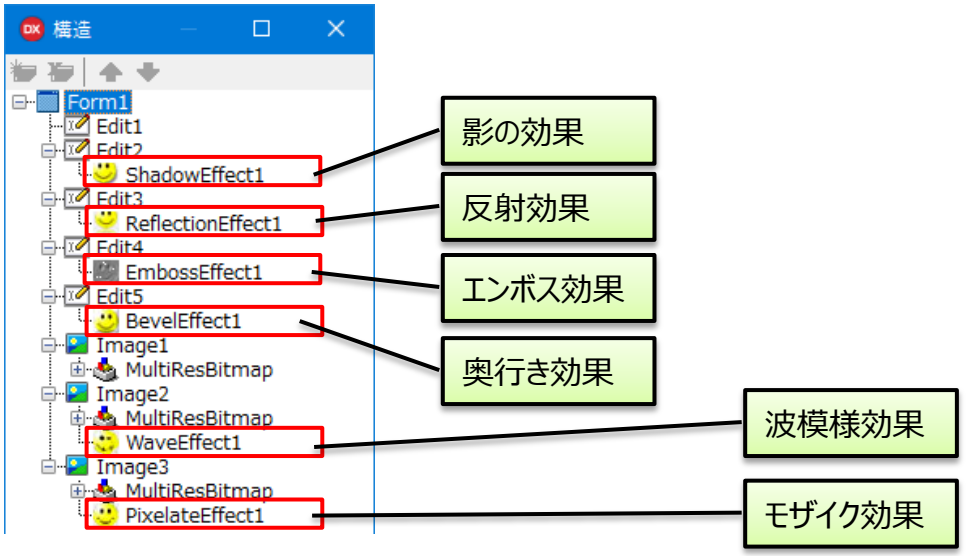
実行イメージ



Effect効果

- Effect効果の例

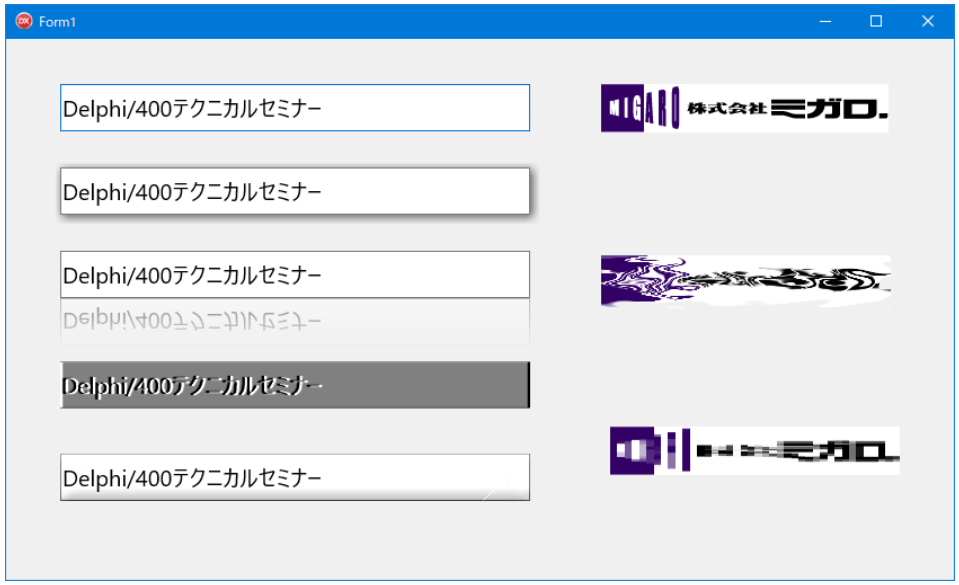




- 効果の詳細は、エンバカデロDocWiki「FireMonkeyの画像効果」に記載。

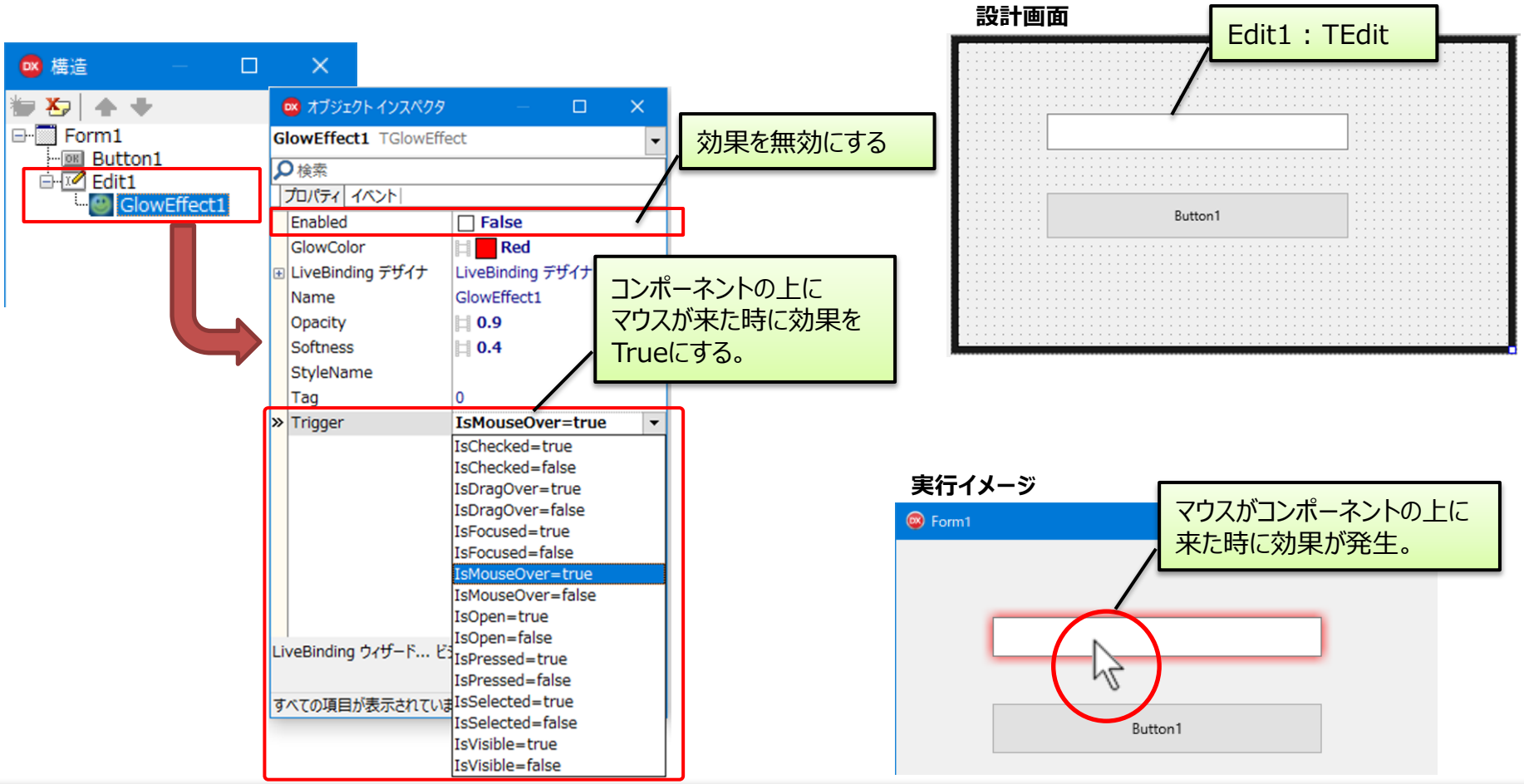
http://docwiki.embarcadero.com/RADStudio/Seattle/ja/FireMonkey_の画像効果

実行イメージ



Effect効果

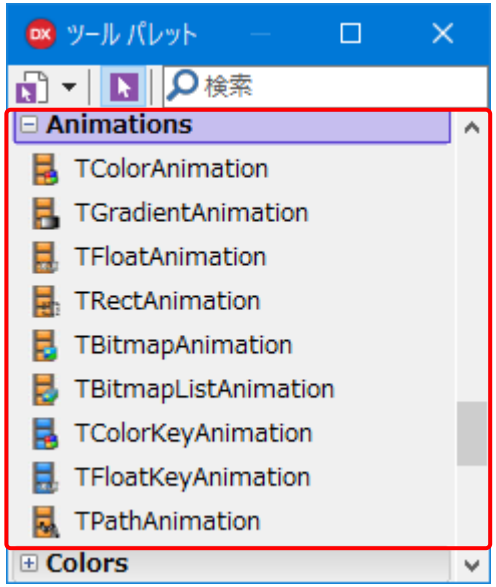
- Effect効果 実行条件の指定
  - Effect効果を実行する条件（トリガー）を指定可能



■ アニメーション効果

- アニメーション：プロパティの値を変更する仕組み
  - 時間の経過にあわせて、プロパティの値を変化させていく。
  - 任意のタイミングで開始/終了したり、トリガーによって実行する。

  - アニメーションの詳細は、エンバカデロDocWiki「FireMonkeyのアニメーション効果」に記載
    http://docwiki.embarcadero.com/RADStudio/Seattle/ja/FireMonkey_のアニメーション効果



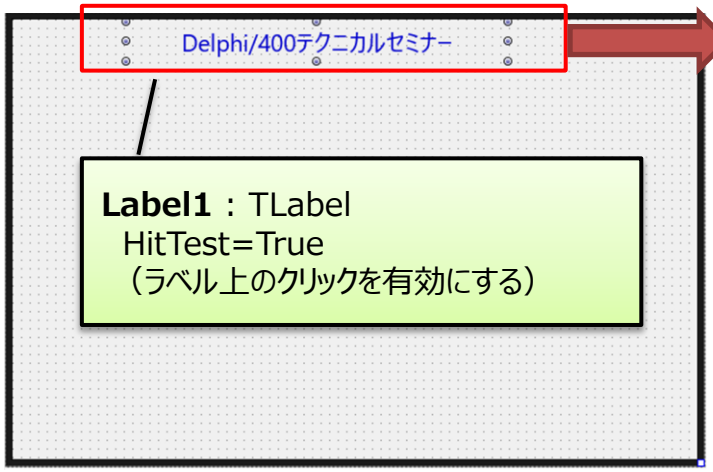
- TFloatAnimation
  - 数値型プロパティに対するアニメーション
  - 主なプロパティ

ﾌﾟﾛﾊﾟﾃｨ	機能
StartFromCurrent	True：現在のプロパティ値を初期値とする
StartValue	開始値　（StartFromCurrent=Falseの場合）
StopValue	終了値
Duration	アニメーション時間（秒）
Loop	True：アニメーションを繰り返す

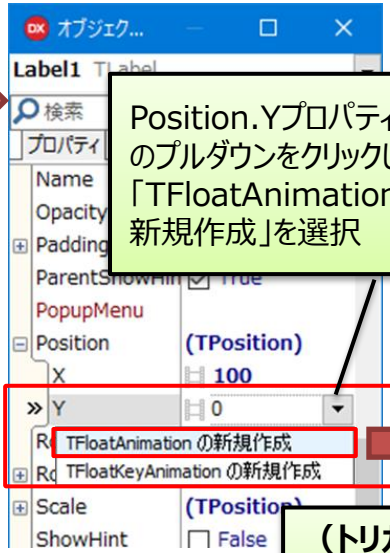
■ アニメーション効果

- アニメーション効果の例

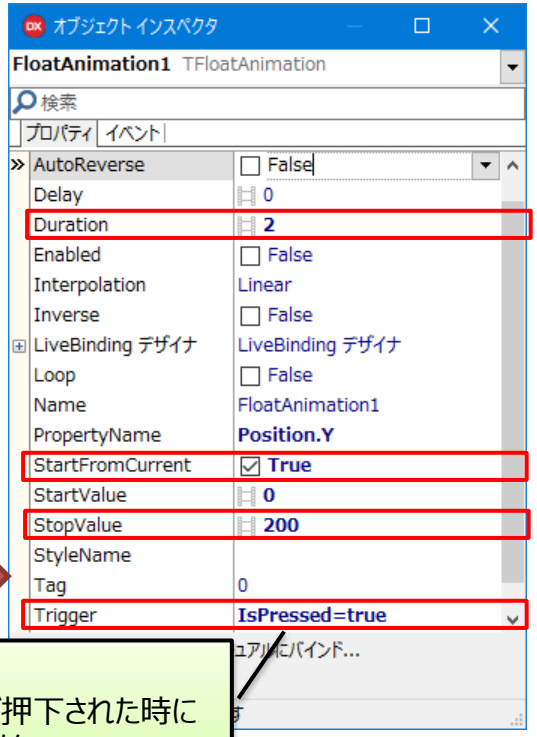
設計画面



Label1：TLabel
HitTest=True
（ラベル上のクリックを有効にする）

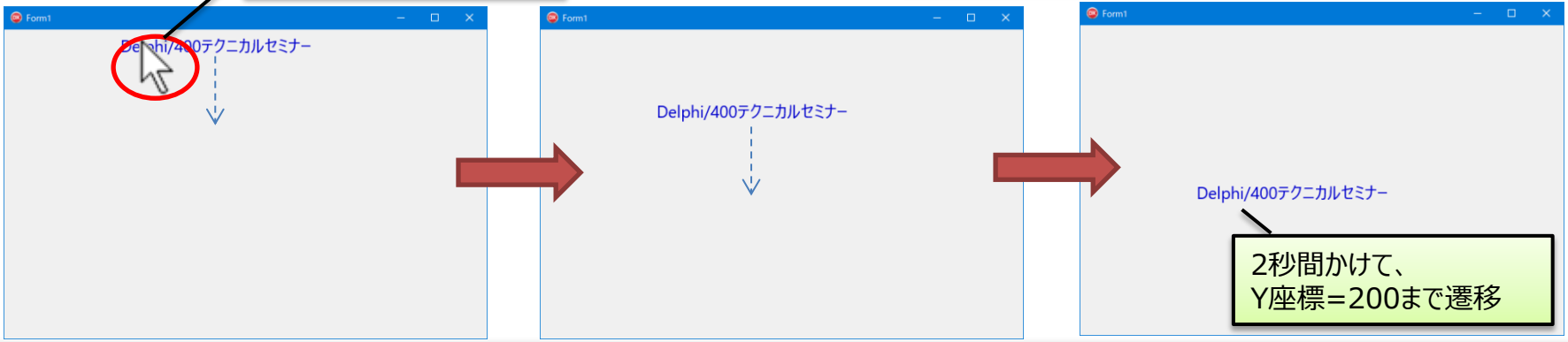


Position.Yプロパティのプルダウンをクリックし、「TFloatAnimationの新規作成」を選択



（トリガー）
コンポーネントが押下された時にアニメーション開始

実行イメージ



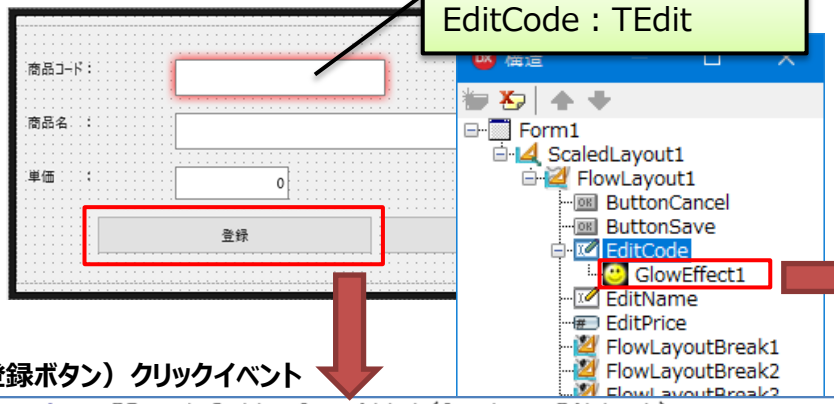
ラベルをクリックすると、文字が下にスライド開始

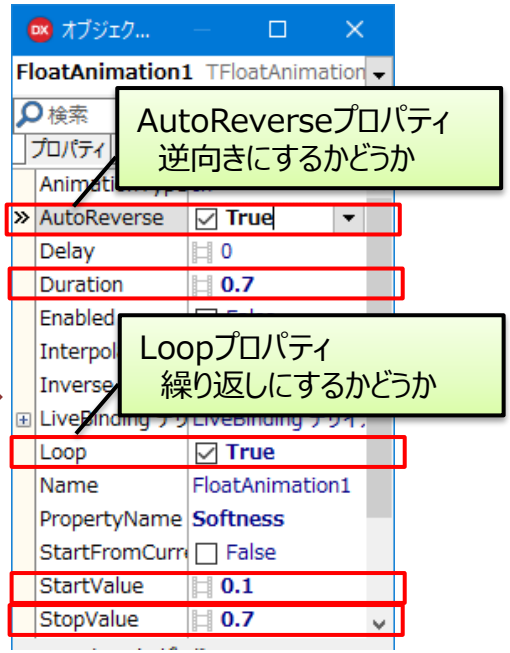
2秒間かけて、Y座標=200まで遷移

■ アニメーション効果

- エフェクトとアニメーション効果の組み合わせ例
  - エフェクトの効果具合をアニメーション制御することで、動きのあるエフェクトが作成可能。

設計画面

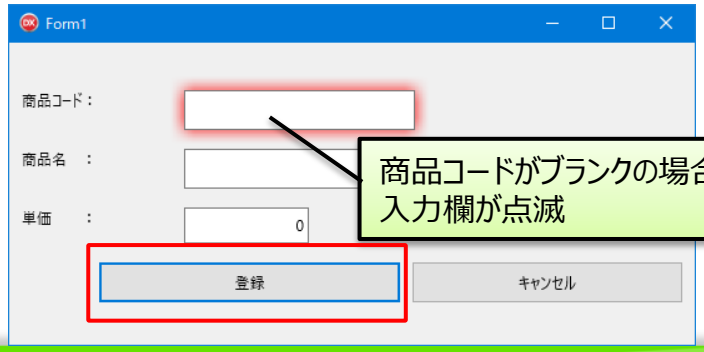




(登録ボタン) クリックイベント

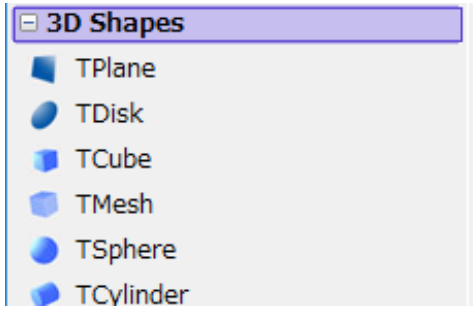
```
procedure TForm1.ButtonSaveClick(Sender: TObject);
begin
  GlowEffect1.Enabled := False;  //エフェクトOFF
  FloatAnimation1.Stop;          //アニメーションストップ

  //エラーチェック
  if EditCode.Text = '' then
  begin

    GlowEffect1.Enabled := True; //エフェクトON
    FloatAnimation1.Start;       //アニメーションスタート
    EditCode.SetFocus;

    MessageDlg('商品コードが未入力です。', TMsgDlgType.mtError,
      [TMsgDlgBtn.mbOK], 0);

    Abort;
  end;

  //更新処理
      ...

end;
```

実行イメージ

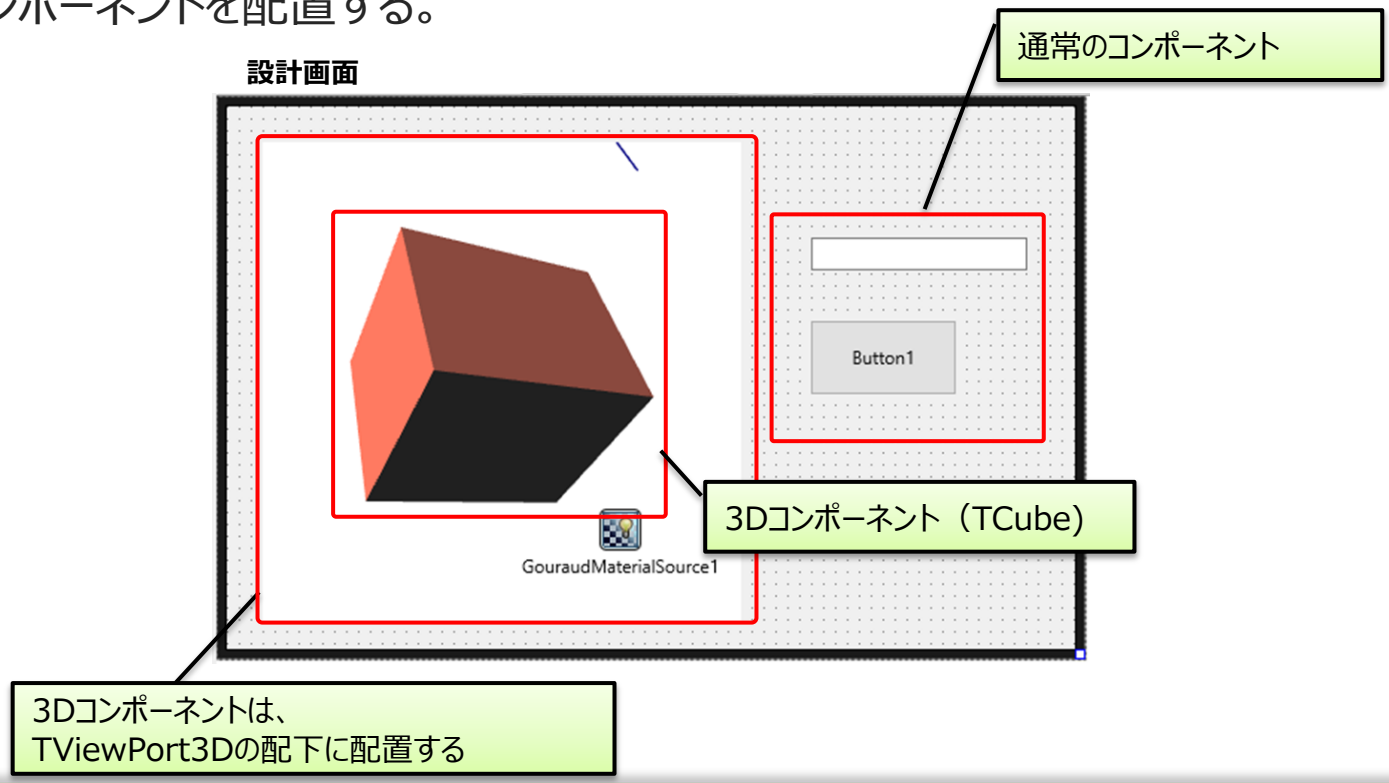


■ 3D効果

- TViewPort3DとTLayer3D
  - FireMonkeyには、3Dコンポーネントも多数用意されている。



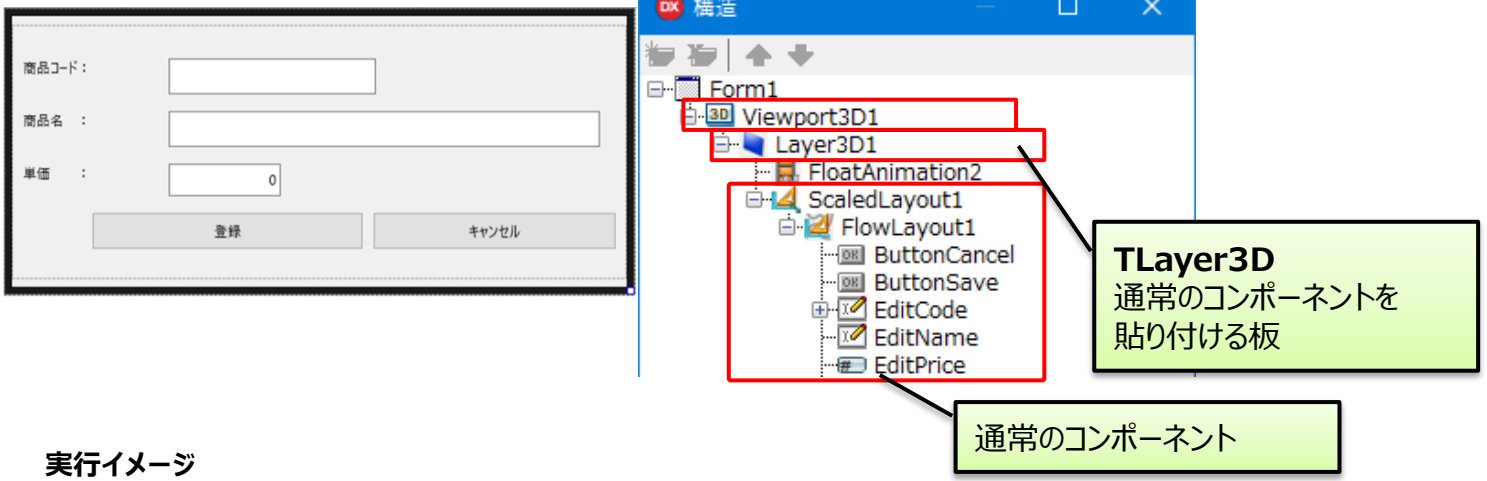
  - フォームに3Dコンポーネントを追加する場合、3D空間の土台としてTViewPort3Dコンポーネントを配置する。



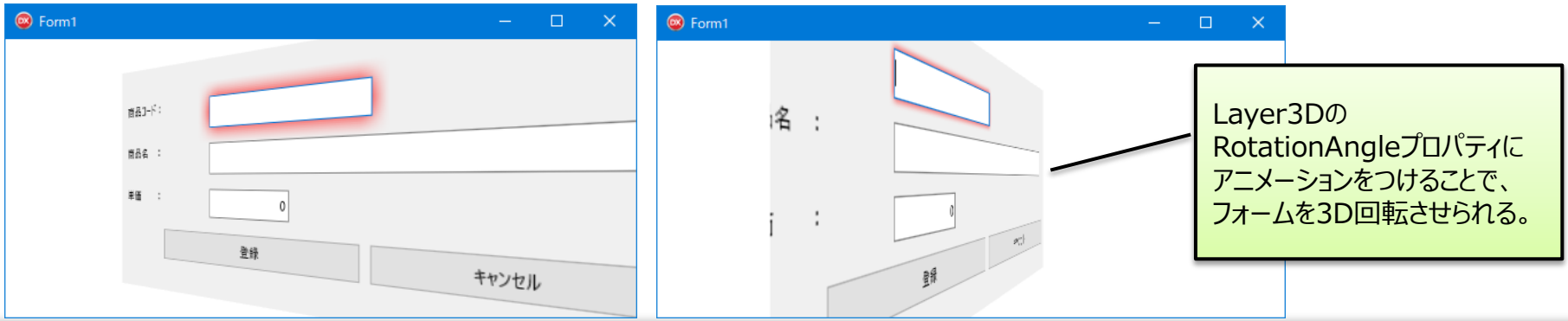
■ 3D効果

- TViewPort3DとTLayer3D
  - 3Dコンポーネントの一つに、通常コンポーネントを貼り付ける板となるTLayer3Dがある。これを使用すると、通常コンポーネントのフォームに3Dアニメーション効果をつけられる。

設計画面



実行イメージ



５．　まとめ

■ まとめ

- **FireMonkey　アプリ開発入門**
  - FireMonkeyアプリ作成手順
  - VCLとFireMonkeyとの一般的な開発手法の違い
  - FireMonkey特有の機能やコンポーネント

- **FireMonkey 効果的な機能の活用**
  - レイアウトコンポーネントを使用したフォームレイアウト作成方法
  - コンポーネントの配置方法
  - スタイルを使用した画面のカスタマイズ
  - コンポーネントに対するEffect効果の設定方法
  - アニメーション効果の使用方法

ご清聴ありがとうございました。