embarcadero
Developer Camp

【B2】Delphi テクニカルセッション

「見た目で楽しい Delphi プログラミング」

会社名　株式会社シリアルゲームズ
名前　細川　淳

アジェンダ

- VCL / FireMonkey のスタイルの触りを紹介します
- VCL スタイル
  - TStyleManager
- FireMonkey スタイル
  - TStyleBook

embarcadero
Developer Camp

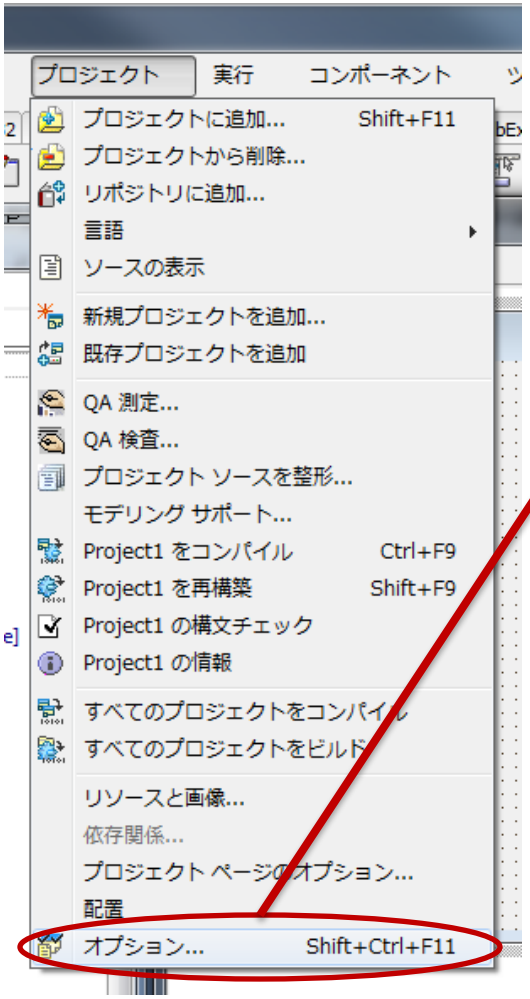
VCL スタイル

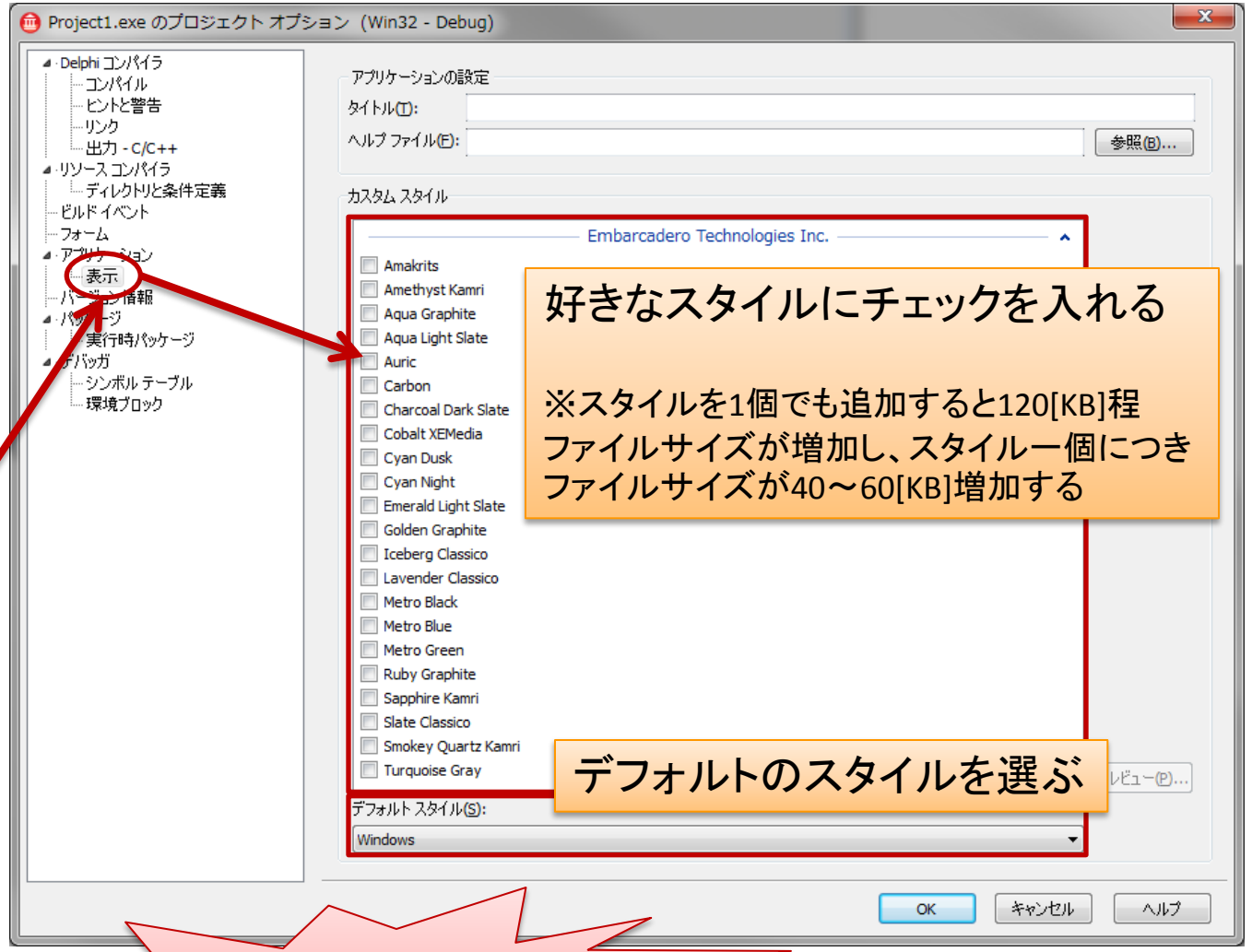
VCL スタイル

- Delphi XE 以前
  - TThemeServices
    - OS が提供する「テーマ」を利用するクラス
    - Windows XP Style (Luna など)に対応するために追加されました
    - ※XE2 以降でも依然として利用可能

- Delphi XE2 以降
  - TStyleManager
    - VCL / FireMoneky に新しく追加された「スタイル」を利用するクラス
    - スタイルを利用すると、OS のビジュアル（見た目）から解放されます
      →特にマルチプラットフォームである FireMonkey において重要

VCL スタイルの一番簡単な使い方





好きなスタイルにチェックを入れる

※スタイルを1個でも追加すると120[KB]程ファイルサイズが増加し、スタイル一個につきファイルサイズが40～60[KB]増加する

デフォルトのスタイルを選ぶ

たったこれだけ！

embarcadero
Developer Camp

VCL スタイルを適用するデモンストレーション

embarcadero
Developer Camp

VCL スタイルを
プログラムから使う

VCL スタイル - TStyleManager

- TStyleManager.TrySetStyle
  - スタイルを設定するメソッド
  - とりあえずこのメソッドさえ知っていればスタイルの適用は可能

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  TStyleManager.TrySetStyle('Amakrits'); // スタイルの名前
end;
```

VCL スタイル - TStyleManager

- TStyleManager.StyleNames
  - 使用可能なスタイルを返すメソッド

```
procedure TForm1.Button2Click(Sender: TObject);
var
  StyleName: String;
begin
  for StyleName in TStyleManager.StyleNames do
    ListBox1.Items.Add(StyleName);
end;
```

名前をリストアップして、それを適用します

```
procedure TForm1.Button3Click(Sender: TObject);
begin
  // SetStyle は内部で TrySetStyle を呼び出し、失敗すると例外をあげる
  if (ListBox1.ItemIndex > -1) then
    TStyleManager.SetStyle(ListBox1.Items[ListBox1.ItemIndex]);
end;
```

embarcadero
Developer Camp

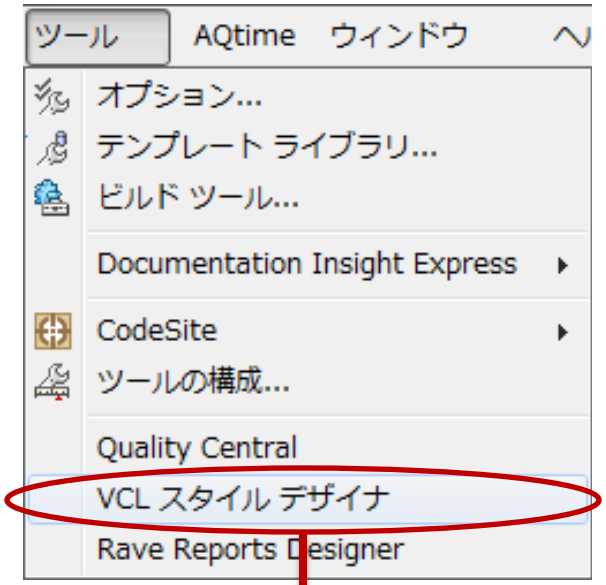
VCL スタイルを プログラムから 適用する デモンストレーション

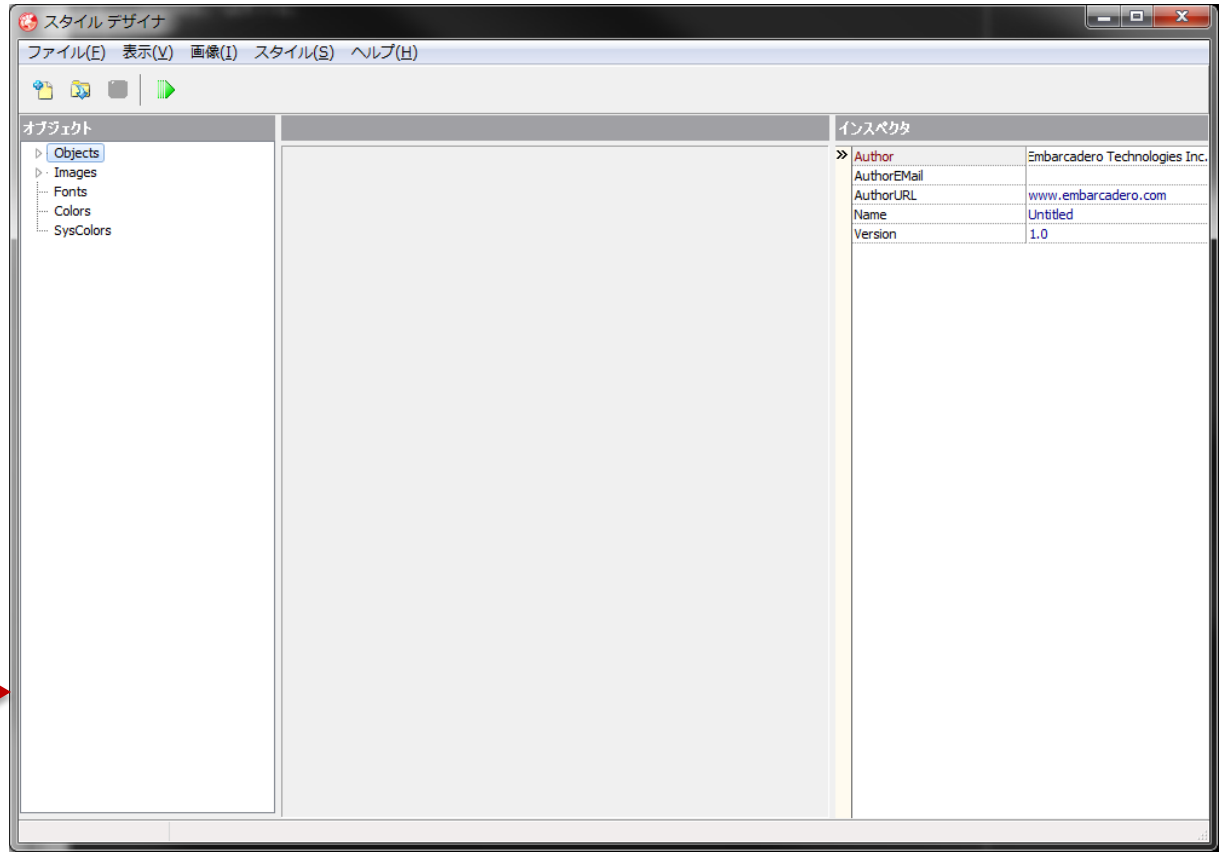
embarcadero
Developer Camp

VCL スタイルを自分で作ってみる

VCL スタイル - VCL スタイルデザイナ

自分でスタイルを作成する場合、VCL スタイルデザイナを使います。



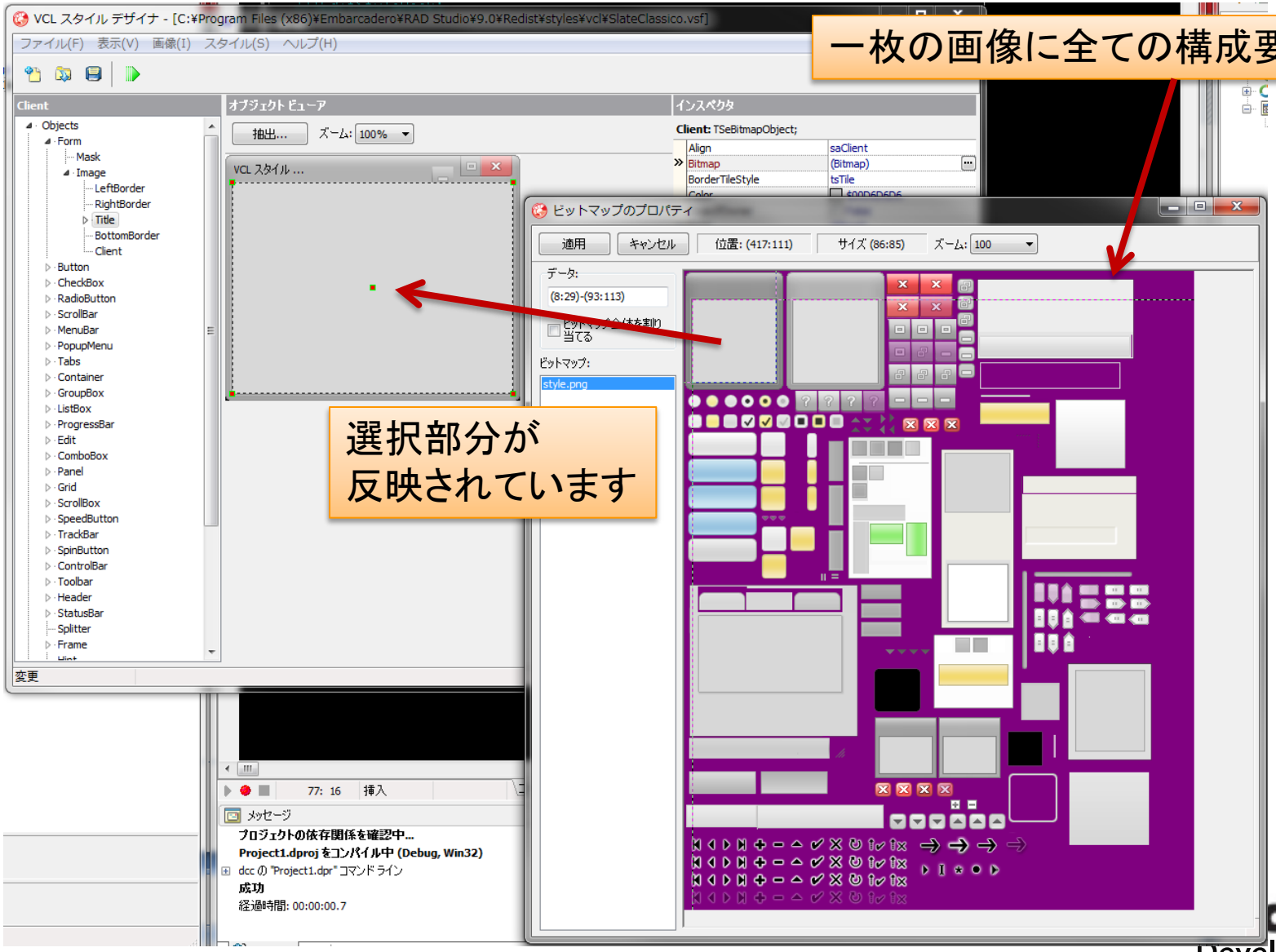


VCL スタイル - VCL スタイルデザイナ

- スタイルは基本的に各部分部分を表す画像で提供されます
  - ゲームのように一枚の画像からパーツを切り出して使います

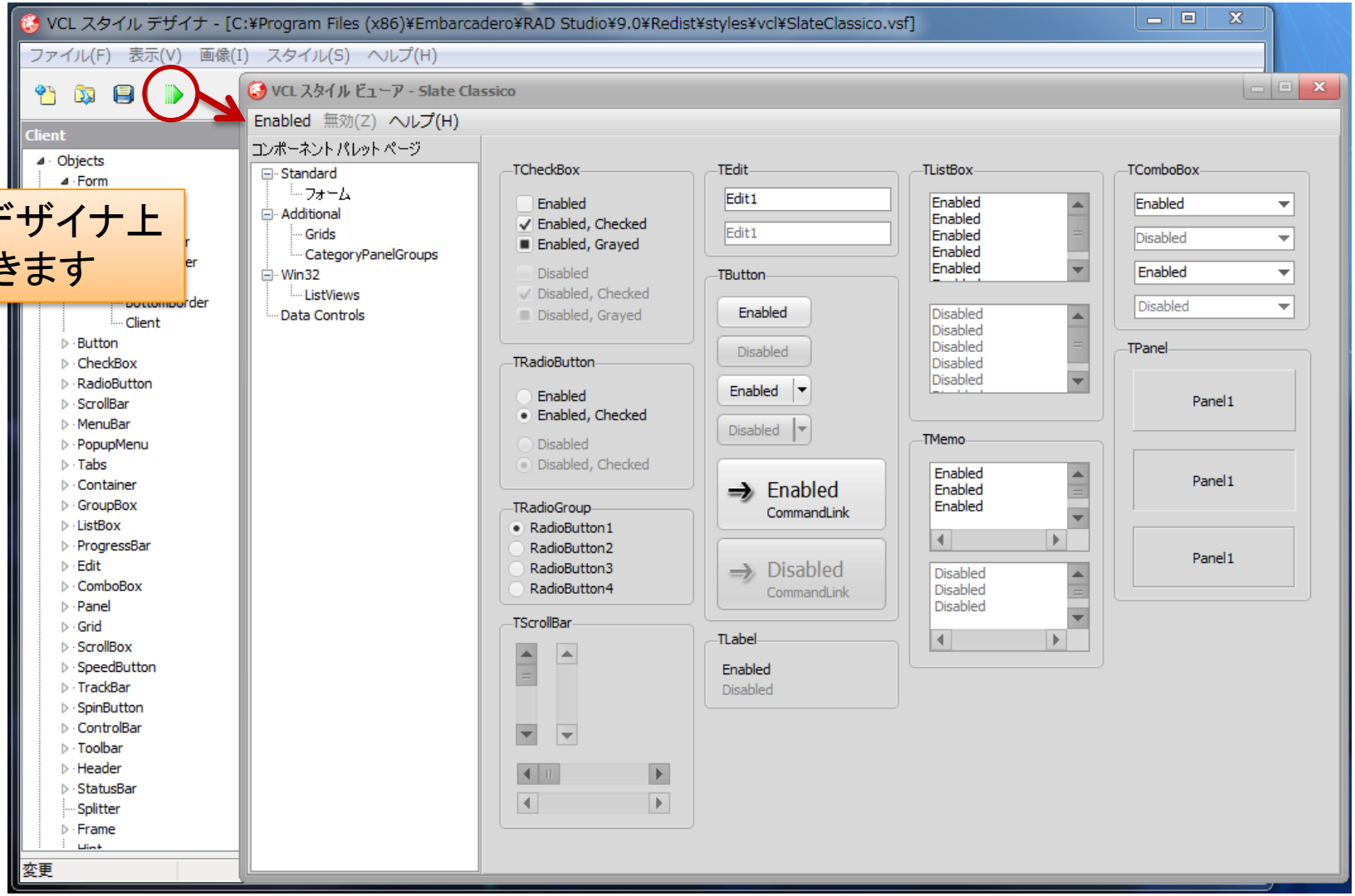
- 設定項目が多い！
  - 自分のスタイルをゼロから作ろうと思っても挫折します
  - 既存のスタイルを修正していくと良いでしょう

VCL スタイル - VCL スタイルデザイナ



VCL スタイル - VCL スタイルデザイナ

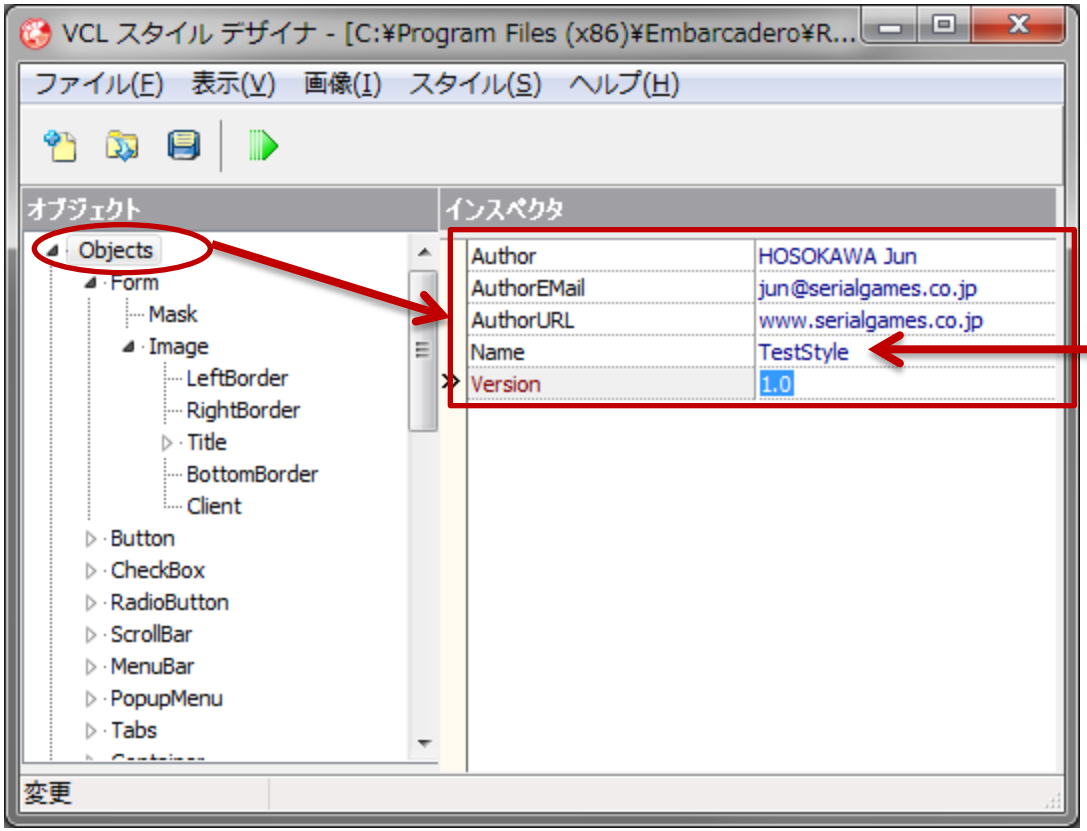
スタイルデザイナ上で確認できます



embarcadero
Developer Camp

VCL スタイル
スタイルデザイナの
デモ

VCL スタイル - 自作スタイルの使い方



まず Object のトップのプロパティを設定しておきます

Name プロパティがスタイルの名前になります

VCL スタイル - 自作スタイルの使い方

- TStyleManager.LoadFromFile
  - スタイルファイルを読み込むメソッド

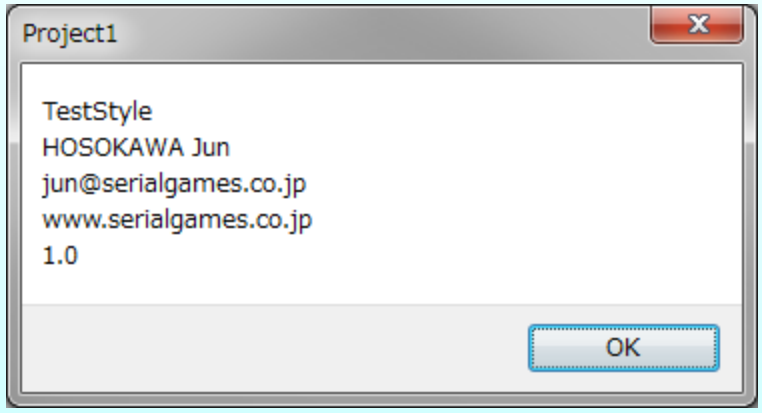
```
procedure TForm1.Button2Click(Sender: TObject);
begin
  TStyleManager.LoadFromFile('C:¥Temp¥TestStyle.vsf');
  TStyleManager.TrySetStyle('TestStyle'); //スタイルデザイナで設定した名前
end;
```

VCL スタイル - 自作スタイルの使い方

- TStyleManager.IsValidStyle
  - スタイルファイルの正当性チェック＆スタイル情報取得

```
procedure TForm1.Button2Click(Sender: TObject);
var
  StyleInfo: TStyleInfo;
begin
  if
    (TStyleManager.IsValidStyle('C:¥Temp¥TestStyle.vsf', StyleInfo))
  then
    with StyleInfo do
      ShowMessage(
        Name + sLineBreak +
        Author + sLineBreak +
        AuthorEMail + sLineBreak +
        AuthorURL + sLineBreak +
        Version
      );
end;
```

実行結果



embarcadero
Developer Camp

VCL スタイル
スタイルファイル
読み込み
デモ

embarcadero
Developer Camp

VCL スタイルを使用するコントロールの作成

VCL スタイル - コントロールの作成

一番単純な方法は Paint メソッドや WM_NCPAINT, WM_PAINT に応えて自分で実装します。

ノーマル状態のボタンを描画する例

```
procedure TForm1.PaintBox1Paint(Sender: TObject);
var
  DC: HDC;
  R: TRect;
begin
  DC := PaintBox1.Canvas.Handle;
  R := PaintBox1.ClientRect;

  StyleServices.DrawElement(
    DC,
    StyleServices.GetElementDetails(tbPushButtonNormal),
    R);

  StyleServices.DrawText(
    DC,
    StyleServices.GetElementDetails(tbPushButtonNormal),
    'TEST CAPTION',
    R,
    TTextFormatFlags(DT_CENTER or DT_VCENTER or DT_SINGLELINE),
    StyleServices.GetStyleFontColor(sfButtonTextNormal));
end;
```

VCL スタイル - コントロールの作成

- 一般的に、VCLスタイルの描画は TStyleHook を使います。
  - TStyleHook とは TStyleManager から呼び出されコントロールの描画を担います。
  - WM_PAINT や WM_NCPAINT などを「フック」して、それらの応答をコントロールに変わって実行します。
  - ただし、**OS 標準スタイル時の描画には使われません**
    - 標準のスタイル時は、従来と同様に、コンポーネントの Paint メソッドなどで処理されます。

VCL スタイル - TStyleHook の実例

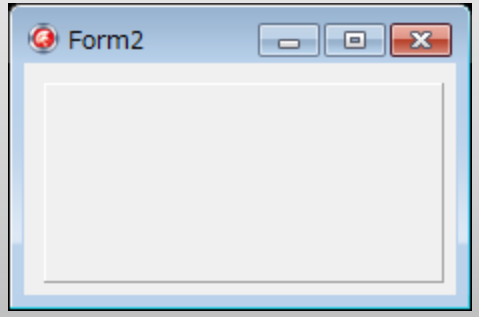
```
unit uTestControl;

interface

uses
  Winapi.Windows, Vcl.Controls, Vcl.ExtCtrls, Vcl.Graphics, Vcl.Themes;

type
  TTestControl = class(TPanel)
  private
    class constructor Create; // クラスコンストラクタは initialization に追加するのと同じ
  end;

implementation

type
  TTestControlStyleHook = class(TStyleHook)
  protected
    procedure Paint(Canvas: TCanvas); override;
  public
    constructor Create(iWinControl: TWinControl); override;
  end;
```

VCL スタイル - TStyleHook の実例

```
{ TTestControl }

class constructor TTestControl.Create;
begin
  // StyleHook とコントロールを結び付けます
  TCustomStyleEngine.RegisterStyleHook(TTestControl, TTestControlStyleHook);
end;

{ TTestControlStyleHook }

constructor TTestControlStyleHook.Create(iWinControl: TWinControl);
begin
  inherited;

  OverridePaint := True; // Paint メソッドをフックしますよ、という意味
end;

procedure TTestControlStyleHook.Paint(Canvas: TCanvas);
begin
  // TTestControl の Paint の代わりにこちらが呼ばれる
  (ノーマル状態のボタンを描画する例と同様のため中略)
end;

end.
```

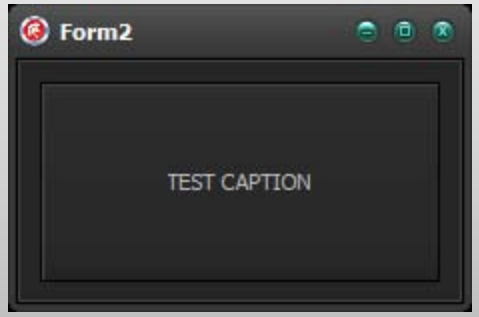
VCL スタイル - TStyleHook の実例

Windows 標準時



TTestControl が継承している TPanel の標準の見た目が描画される

VCL スタイル適用時



TTestControlStyleHook がスタイルに応じた見た目を描画する

embarcadero
Developer Camp

VCL スタイル
TStyleHook
デモ

embarcadero
Developer Camp

FireMonkey スタイル

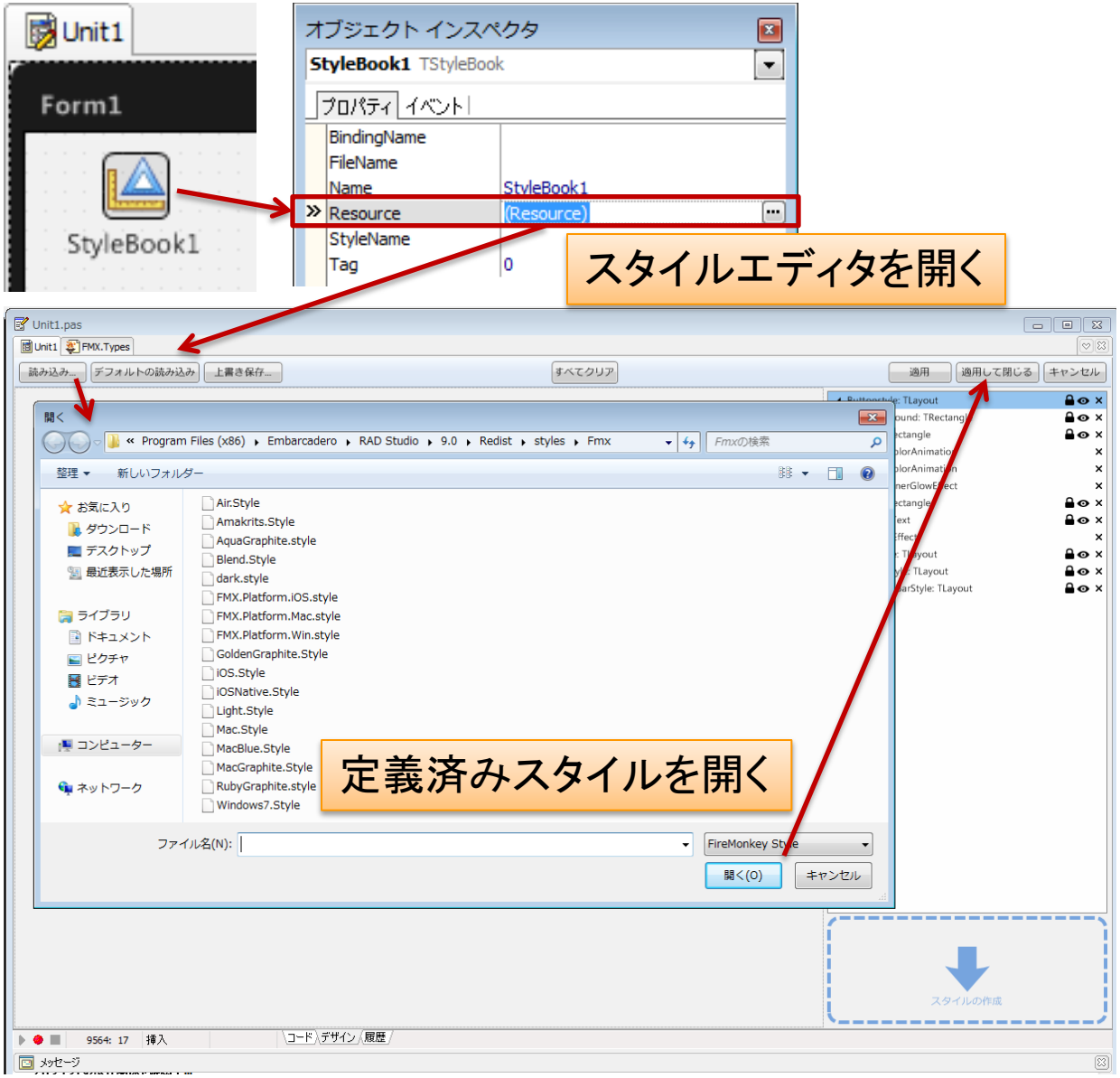
FireMonkey スタイル

- FireMonkey のスタイルは VCL スタイルとはまったく異なります。
  - 個々のコントロールがスタイルを持っている
    - TStyledControl を継承したコントロール
    - 個別に設定できる
  - スタイルは TStyleBook で管理される
    - TStyleBook で定義済みのスタイルリソースを読み込むとスタイルを一括で変更できる

FireMonkey スタイル

- 一番簡単な使用方法は…
  - Form に TStyleBook コンポーネントを貼る
  - TStyleBook コンポーネントの「Resource」プロパティをクリックしてスタイルエディタを開く
  - スタイルエディタから定義済みスタイルを読み込む
  - 「適用して終了」を押す

FireMonkey スタイル 簡単な設定方法

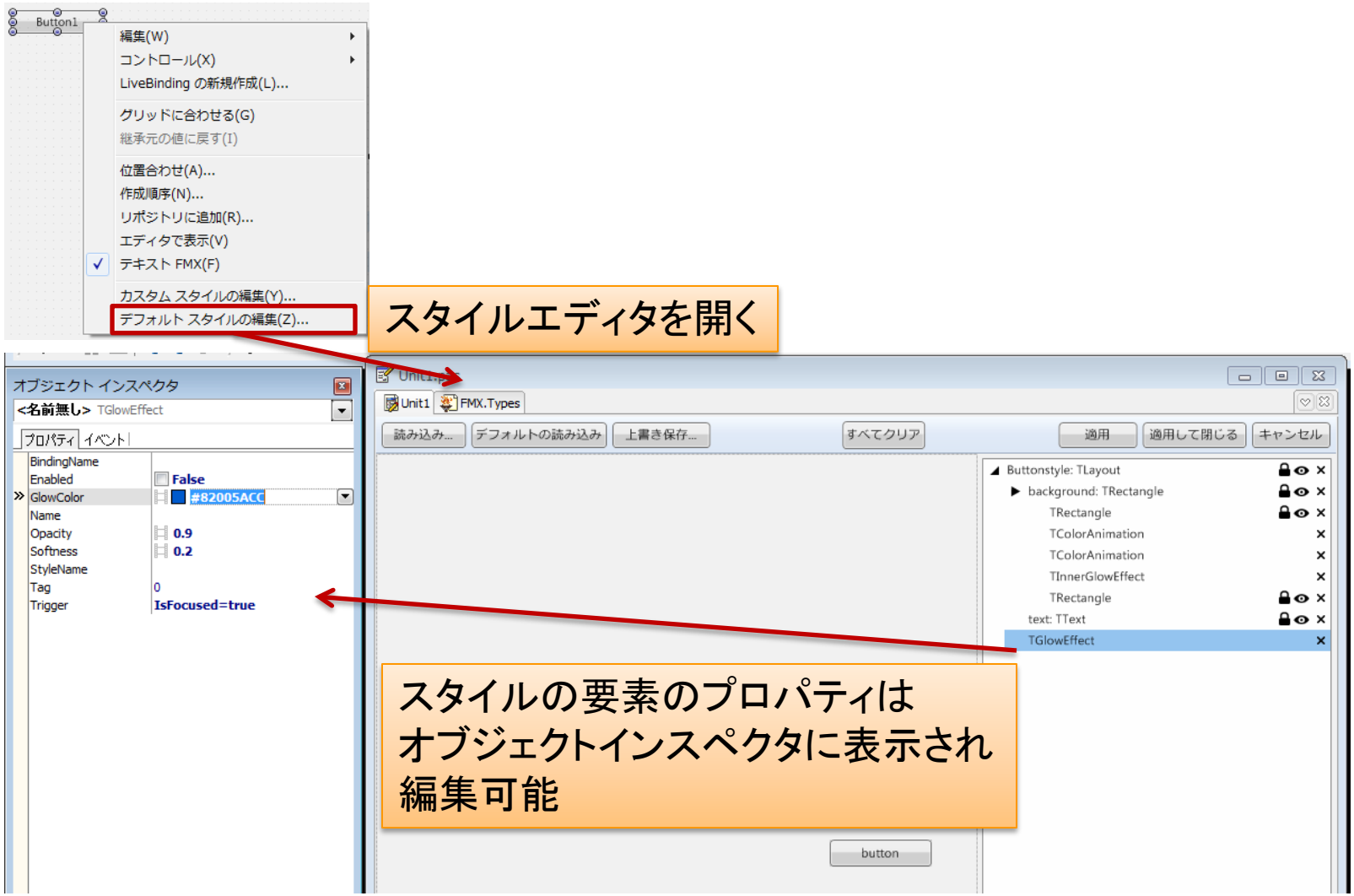


FireMonkey スタイル - コンポーネント毎

- 個々のコンポーネントごとに編集する方法は…
  - コンポーネントを右クリックして
  - 「カスタムスタイルの編集」「デフォルトスタイルの編集」のどちらかを選ぶ
  - スタイルエディタが開く
  - 「適用して終了」を押す

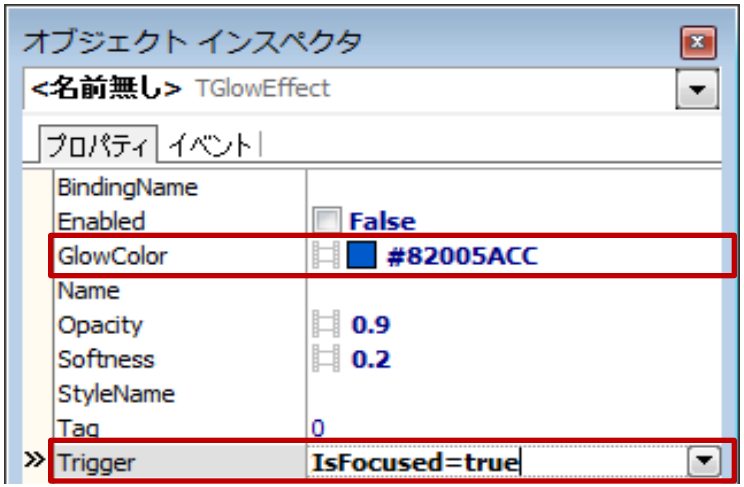
※個々のコンポーネントのスタイルを編集すると、自動的にTStyleBook コンポーネントのインスタンスが生成される

FireMonkey スタイル - コンポーネント毎



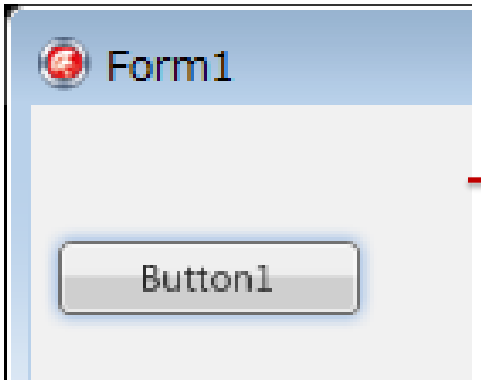
FireMonkey スタイル - コンポーネント毎の例

- フォーカス時に使われる TGlowEffect の色を変えてみる

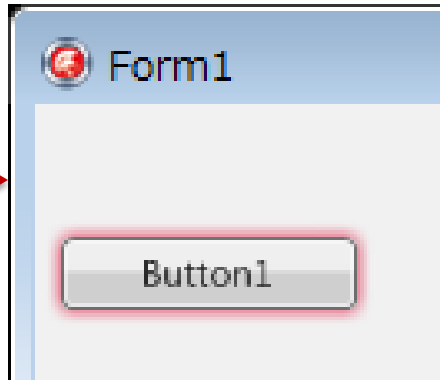


GlowEffect の色

GlowEffect が発動するトリガー
ほかにも IsMouseOver などがある



GlowColor を「Crimson」($DC143C)
に変更

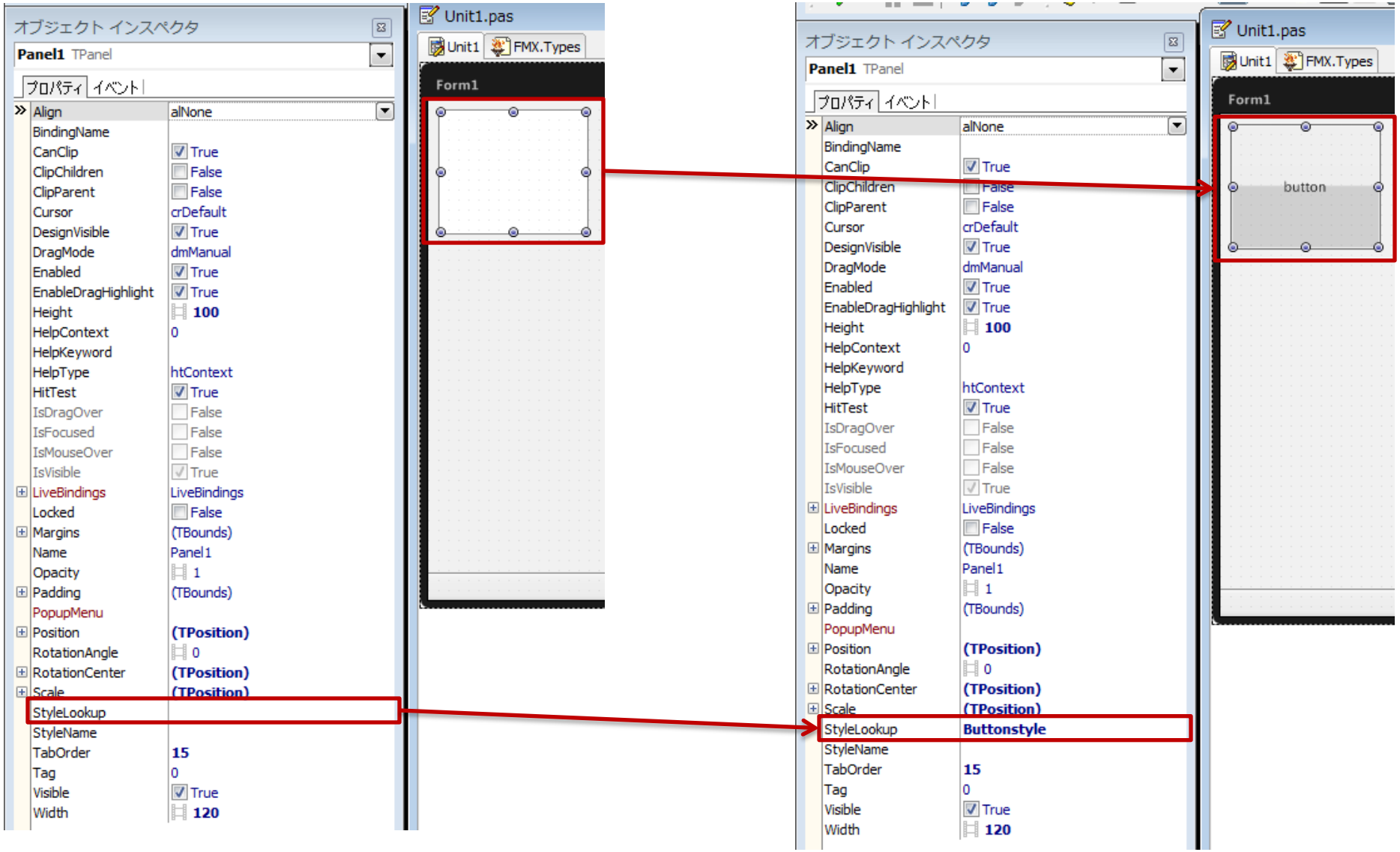


FireMonkey スタイル - StyleLookup

- コンポーネントの StyleLookup プロパティを設定するとデフォルトのスタイルを変更できます。
  - StyleLookup プロパティは、このコントロールが使うスタイルの名前です。
  - このプロパティが設定されると FindStyleResource メソッドによって適切なスタイルが設定されます。

FireMonkey スタイル - StyleLookup

- TPanel の StyleLookup を変更して見た目をボタンのようにしてみます。



embarcadero
Developer Camp

FireMonkey スタイル デモ

embarcadero
Developer Camp

FireMonkey スタイル そのほか

FireMonkey スタイル

- スタイルの個々の要素をプログラムから変更することができます（本当は）

- 現状では小細工しないと変更できません
  - Delphi XE2 Update4 で確認（QC100275 既に Closed）
  - これについては、EDN TeamJapan の高橋さんのブログ記事が詳しいです
    - http://blogs.embarcadero.com/teamj/2011/11/11/2589/

上記記事より引用

FireMonkeyのコンポーネントの**FindStyleResource**メソッドを呼び出して、スタイルにアクセスしようとしますが、なぜか**nil**(見つからなかったということ)が返ってきてしまいます。

この問題(現象)はQualityCentralにも報告されたのですが、トップレベルのスタイルの場合、親クラスである「TStyleControlクラスのprotectedなメンバであるFResourceLink」にアクセスする必要がある(As Designed, 仕様)という結果になりました。

embarcadero
Developer Camp

付録:UI 設計の指針

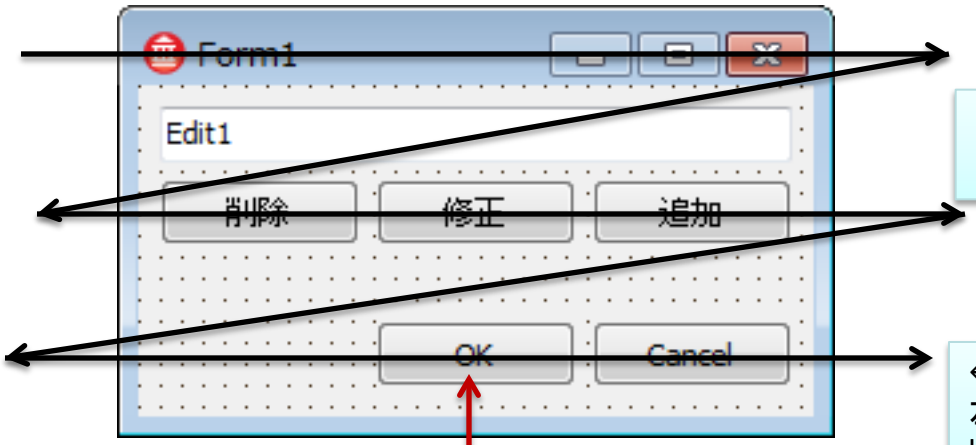
UI 設計の指針

- VCL / FireMonkey スタイルを利用することで OS の見た目から解放されます。
- しかし、**OS の UI から解放されるわけでありません**。そのため OS で提供されている一般的な操作が犠牲になる場合があります。
  - 例として
    - Flash (Air)
      - Win / Mac / Linux で動作するが、一般化のためにコンテキストメニューが廃止されている
    - Unitiy
      - Android / iOS などで動作するが、一般化のために「戻るボタン」による Activity 遷移がサポートされていない
- FireMonkey の場合は特に OS の提供する UX の犠牲を最小限にとどめる必要があります。
- VCL は Windows アプリケーションとなりますが、見た目が自由になったからといって、独自**すぎる** UI を提供すべきではありません。

UI 設計の指針 - 一般的論としての UI

- 視線は左から右、上から下に流れる

視線の流れ



このコンテキストで一般的な動作は右側に
イリーガルな操作は左側に。

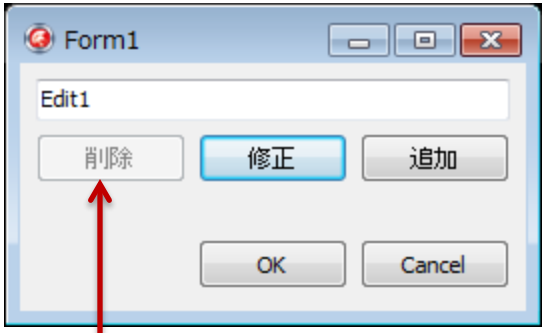
←ここには視線が最後にたどり着く
左上から見ていって、途中に問題がない事を確認していくので、
UI の右下に「OK」や「確認」といったボタンがある

上記原則に反するが、Windows は MacOS との類似を避けるためにOK ボタンを左にした
FireMonkey の場合は、MacOS に合わせるか、Windows に合わせるかはメインターゲットによる。
もしくは、プログラムで動的に変更する

参考として Android は 2.x までは左側、3.x からは右側に変更されている

UI 設計の指針 - 一般的論としての UI

- 視覚効果は感覚に合わせる
  - グレーアウトされていれば押せない
    - MouseOver で凹むボタンなど言語道断
      - 感覚としてボタンは押されると凹むもの
  - 曖昧なアイコンは使わない

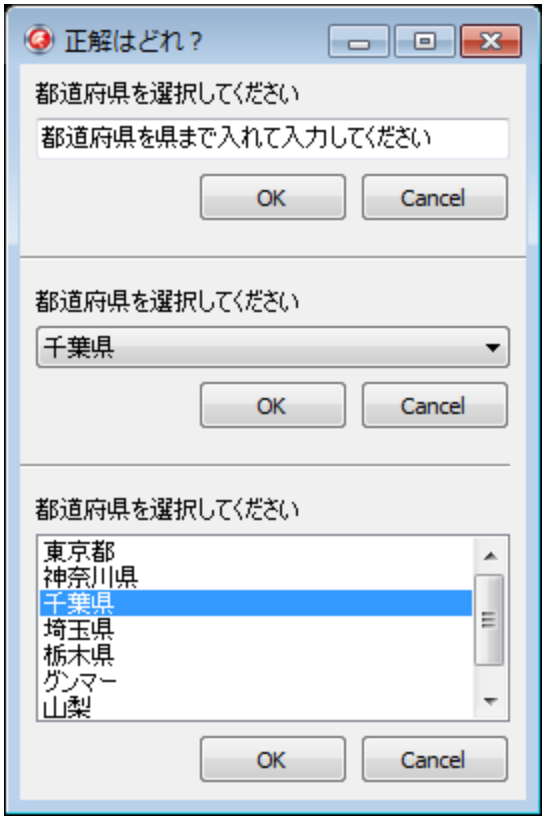


グレーアウトされているボタンは押せない。
マウスオーバーするとグレーが解除されたり、
ポップアップしたりして押せる事をアピールする
変な動作がウェブにあふれていますが、間違い。

このアイコンをクリックすると起きるアクションは？
・音がOFFになる
・音がONになる
どっち？

UI 設計の指針 - 一般的論としての UI

- その UI で正しいかどうか考慮する
  - TComboBox と ListBox
  - TLabel と TImage
  - TMainMenu と TPopupMenu
  - etc.

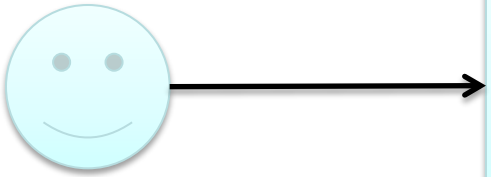


都道府県の選択の場合、
どの UI が正解だと思いますか？

UI 設計の指針 - その他の事

ターゲット UI はターゲット層で決まる

ターゲット層



ターゲットデバイス

ターゲット UI

色覚異常への対応

それぞれが判別可能か？

ユーザビリティテスト

独りよがりな UI になっていないか？

embarcadero
Developer Camp

質疑応答