尾崎 浩司

株式会社ミガロ.

RAD事業部 営業・営業推進課

[Delphi/400] デスクトップアプリケーション開発でも役立つFireMonkey活用入門

略歴
1973年8月16日生まれ
1996年 三重大学工学部卒業
1999年10月 株式会社ミガロ. 入社
1999年10月 システム事業部配属
2013年4月 RAD事業部配属

現在の仕事内容
ミガロ. 製品の営業を担当している。これまでのシステム開発経験を活かして、IBM iをご利用のお客様に対して、GUI化、Web化、モバイル化などをご提案している。

1.はじめに

Delphi/400のバージョンXE3以降では、VCL（Visual Component Library）フレームワークとは別に、FireMonkeyという新しいフレームワークが搭載されるようになった。

Delphi/400は、VCLを使用したWindowsクライアント上で実行できるアプリケーション開発に長年利用されているが、この新しいFireMonkeyフレームワークを使用すれば、iOSやAndroid用のモバイル開発も可能である。

そのためFireMonkeyはモバイル開発専用として使われることが多いが、実はWindowsやMacといったデスクトップOS向けの開発にも対応しており、iOS/Androidと合わせて合計4つのプラットフォームに対応するマルチデバイス開発機能を持っている。

本稿では、これまで主にVCLを使用してアプリケーションを開発している方を対象に、Windowsアプリケーション開発で役立つFireMonkeyのポイントを解説する。

2.FireMonkeyとは

2-1 FireMonkeyの概要

FireMonkeyを説明する前に、まず従来のVCLの特徴を確認する。VCLは、Windowsアプリケーション開発専用のフレームワークであり、Windows APIをラッピングしたものである。【図1】

開発者は、VCLコンポーネントとRTL（ランタイムライブラリ）と呼ばれるモジュールを使用して開発する。VCLコンポーネントは、Windows APIをラッピングしているので、Windowsに用意されたすべての機能を活用できる。

これに対してFireMonkeyフレームワークは、マルチデバイス向けの開発を目的としたフレームワークである。VCLのようにWindowsに機能特化していないものの、複数のOS向けに汎用的なアプリケーションを開発できる。

またさまざまなOS上で表示可能にするため、グラフィック処理装置（GPU）を使用している。Windowsの場合はDirectX、Macの場合はOpenGL、iOSやAndoridの場合はOpenGL ESといったGPU用APIをラッピングしており、各OSで共通的な機能を活用できる。【図2】

各OSに用意されたすべての機能を標準で活用できるわけではないが、特定のOSに特化しない仕組みが、マルチデバイス対応を実現する根幹となっている。

FireMonkeyの場合、GPUを活用するフレームワークなので、とくに3Dグラフィック処理が優れている。また2Dでも、VCLでは表現できないようなビジュアルやアニメーションなどの表示も可能である。なお、VCLと同様にRTLを使えるので、たとえばIntToStr関数等Delphi/400で利用している関数や手続きは、VCLと同じように使用できる。

2-2 FireMonkeyに最適なアプリケーション

PCアプリケーション開発での、従来

図1

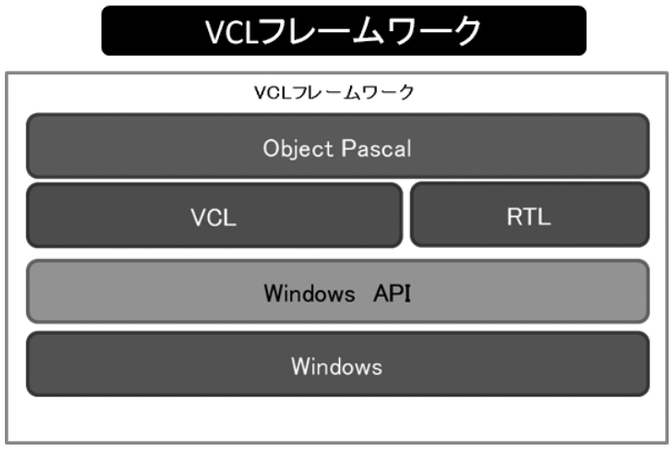


図2

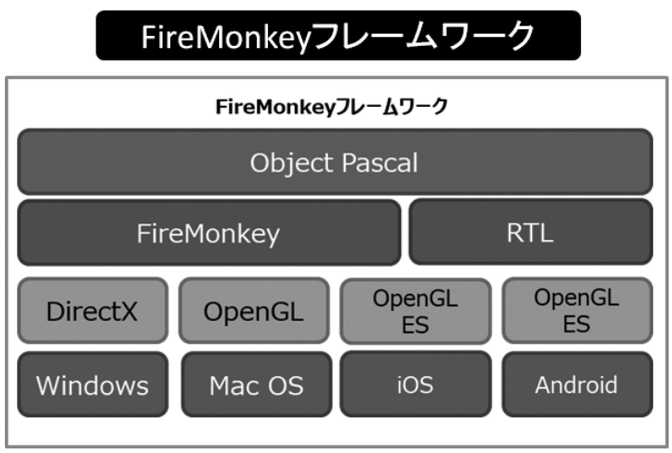
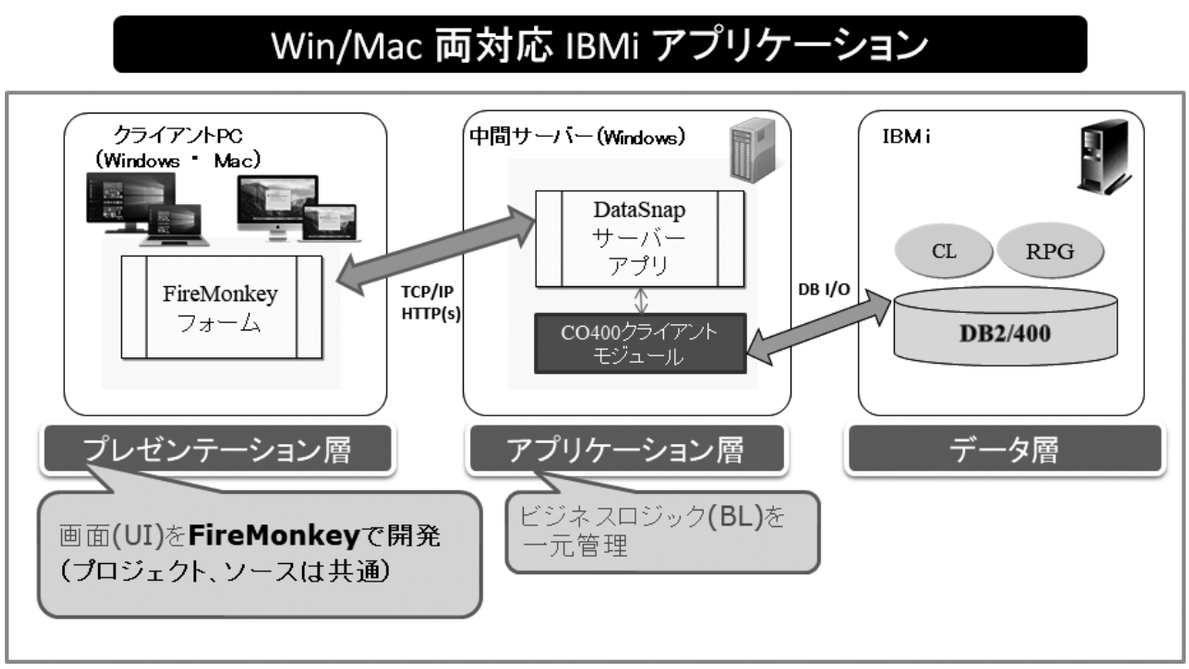


図3



のVCLとFireMonkeyとの使い分けを考えてみる。もしクライアントPCの対象がWindowsだけでなく、Macが含まれるのであれば、FireMonkeyが必要となる。ただしdbExpressやFireDACといったデータベースエンジンのIBM i用ドライバーは、Windows専用となるので、中間サーバー（DataSnapサーバー）を使用した3階層形式での実装となる。【図3】

もちろんクライアントがWindowsだけであれば、FireMonkeyでもVCLと同様のC/S形式で開発できる。

FireMonkeyが最適なアプリケーションとしては、キーボードを持たないタブレットPC用アプリケーションや、画像等のビューア系アプリケーション、デジタル・サイネージに代表される情報系アプリケーションなどが挙げられる。【図4】

これらのアプリケーションは、キーボードでの情報入力ではなく、情報の参照がメインとなる。こうしたアプリケーションの開発は、視覚的な効果が活かせるグラフィック処理に長けたFireMonkeyが最適といえる。

3.VCLとFireMonkeyの違い

3-1　コンポーネントの違い

FireMonkeyのプログラム開発手順はVCL同様、次の4つの手順で実施する。

①プロジェクトの新規作成
②フォームにコンポーネントを配置
③配置したコンポーネントにプロパティを定義
④イベントハンドラにロジック作成

言語としてはどちらもObjectPascalを使用しているので、Delphi/400開発者はVCL同様のプログラミングスキルで開発できる。

VCLとFireMonkeyとの違いの1つは、使用するコンポーネントの種類が異なる点である。FireMonkeyには、VCLと同じ名前のコンポーネントも多数あるが、完全に同じではなく、設定や使用方法が若干異なる。以下に代表的な違いを挙げる。

まずコンポーネント全般の違いとして、VCLでは表示文字列をCaption、入力文字列をTextプロパティで扱うが、FireMonkeyではどちらもTextプロパティで扱う。またコンポーネントの位置を指定するLeft、Topプロパティは、FireMonkeyではPosition.X,Yプロパティに相当する。

このように同じ機能でも、プロパティの異なる場合があるので注意が必要である。【図5】

次に、画面作成に多用するTEditについて詳しく比較してみる。これもいくつかの機能や使用方法が異なっている。【図6】にいくつかの違いを挙げているが、たとえばVCLにあるCharCaseプロパティには、FireMonkeyで相当するプロパティが存在しないので、個々の文字が入力された時に発生するイベント（OnChangeTraking）でロジックを記述する必要がある。

またVCLにおけるColorプロパティは、R（赤）、G（緑）、B（青）の3原色で表されるが、FireMonkeyのColorプロパティは、RGBに加え、A（透過度）という要素を持つので、半透明な色も表現できる。

最後に、Fontプロパティの違いを説明する。VCLの場合はフォントのサイズがポイント単位（1ポイントが1/72インチ）であるが、FireMonkeyの場合はデバイス非依存ピクセル（DIP）単位（1DIPが1/96インチ）である。したがって、同じFont.Size値を持つ場合、FireMonkeyのほうが文字が小さくなる。

3-2　FireMonkey独自のコンポーネント機能

次に、FireMonkey独自の機能について説明する。VCLにもコンポーネントの親子関係があるが、フォーム以外で親となりえるコンポーネントは、TPanelやTGroupBoxなどのコンテナコンポーネントに限定されている。

しかしFireMonkeyの場合は、あらゆるコンポーネントを親子関係にできる。たとえば、TButtonの子としてTImageを使用することで、TBitBtnのような機能を実現したり、TEditの子にTLabelを使用することで、TLabeledEditのような機能を実現できる。【図7】

このようにコンポーネントの親子関係を使用して独自の組み合わせが可能なので、よく使用する組み合わせはコンポーネントテンプレートに登録しておけば、ツールパレットからいつでも利用できる。

またFireMonkeyのコンポーネントはすべてグラフィック描画により実現しているので、表示のカスタマイズ機能に長けている。FireMonkey独自のRotationAngleプロパティは、コンポーネントを表示する角度を自由に指定できる。【図8】

データベースアプリ開発では、VCLの場合、TDBEditやTDBGridなどデータベース連結コンポーネントを使用することが多い。しかし、FireMonkeyにはデータベース連結コンポーネントが存在しない。そのため、TEditやTStringGridなどを使用することになる。

ただしDelphi/400には、LiveBindingという仕組みが用意されているので、この機能を使用するとデータベース連結コンポーネントと同様の実装が可能である。【図9】

4.FireMonkey画面作成のポイント

4-1　フォームレイアウトとコンポーネントの配置

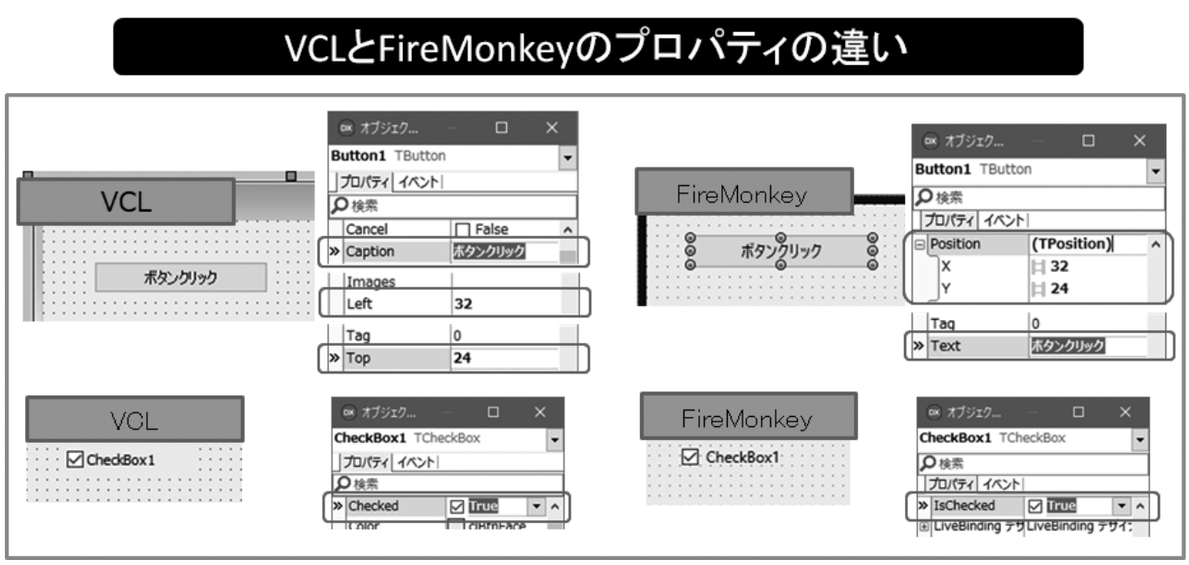
VCLでフォームレイアウトを作成する場合、TForm上に直接コンポーネントを配置して作成する。この時に配置されたコンポーネントは、Left,Topプロパティにより固定位置に配置される。もちろんFireMonkeyでも、Position.X,Yプロパティを同様に定義することで、同じような固定配置による作成が可能である。

しかし従来の解像度を基準に作成した固定のフォームレイアウトは、近年の高解像度ディスプレイやMacなどで使用されるRetinaディスプレイ上では見えづらいことも多い。【図10】

ビジュアル機能に優れたFireMonkeyでは、LayoutsカテゴリにあるTLayoutというレイアウトコンポーネントを使用することで、解像度によって画面を自動調整するようなアプリケーションも開発できる。【図11】

図4

図5



TLayoutは、TPanelのようなコンテナコンポーネントであるが、TLayout自体は表示されない。またTLayout自体のVisibleを変更すると、配下のコンポーネントの表示／非表示が一括設定できる点も便利である。

レイアウトコンポーネントで一番わかりやすいのが、TScaledLayoutである。TScaledLayoutは縮小／拡大が可能なフォームレイアウトで、この配下に配置したコンポーネントは、フォームのサイズ変更に合わせて縮尺が自動的に調整される。

TScaledLayoutを使用すると、固定配置したコンポーネントの場合でも、高解像度ディスプレイの対応が可能となる。【図12】

次に、TFlowLayoutを説明する。これは文章の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フォームレイアウトである。

左端から順に貼り付けたコンポーネントは、フォームの幅を超えると自動的に改行される。またフォームの幅が変更された場合、その変更に応じて改行位置が自動的に調整されるのが特徴である。【図13】

なお文章の場合、段落ごとに改行を入れるが、それと同じように、並べられたコンポーネントの途中で改行するのが、TFlowLayoutBreakである。

配置したコンポーネントのうしろにTFlowLayoutBreakを挿入すると、任意の位置でコンポーネントを改行できる。

このTFlowLayoutは、たとえばタブレット用アプリケーションで、縦画面と横画面を切り替えた時に項目の配置を自動調整する場合に便利である。

3つ目に説明するのは、TGridLayoutである。これは等間隔のセルにコンポーネントを配置するようなフォームレイアウトである。

1つのセルの高さと幅をItemHeight, ItemWidthプロパティで設定し、その中で順番にコンポーネントを配置して利用する。碁盤目のようなフォームレイアウトを作る時に重宝する。【図14】

最後は、TGridPanalLayoutである。これは、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フォームレイアウトである。

グリッドの列情報を保持するColumnCollection、行情報を保持するRowCollection、グリッドに配置されるコンポーネント情報を保持するControlCollectionを使用してフォームレイアウトを作成する。エクセルのように列ごとの幅や行ごとの高さを指定して、セル結合などを行いながら表を調整できる。【図15】

フォームレイアウトが決まったら、次はその中に配置する各コンポーネントを検討する。コンポーネントを固定配置する場合、位置をPosition.X,Yプロパティで指定し、大きさをWidth,Heightプロパティで指定する。

この場合、フォームの大きさが変わっても、コンポーネントの位置やサイズは変わらない。フォームの大きさが変わった時に、コンポーネントの配置を調整するには、配置を定義するプロパティを使用すればよい。具体的には、Alignプロパティ、Marginsプロパティ、Paddingプロパティの3つである。

Alignプロパティは、親コンポーネントに対し整列するためのもので、VCLにも同名のプロパティが存在する。FireMonkeyのAlignプロパティも同様の機能だが、VCLよりも多くのオプション値が用意されている。【図16】

またMarginsプロパティはコンポーネント間の余白をピクセル単位で指定し、Paddingプロパティは親コンポーネントから子コンポーネントまでの距離をピクセル単位で指定する。【図17】

これらを設定すると、画面比率の異なる環境で実行した時にも、コンポーネントの表示が最適に調整できる。

4-2　Effectとアニメーション効果

FireMonkeyはグラフィック処理に長けたフレームワークであり、画面の表示効果にさまざまな機能が用意されている。その代表が、Effect効果とアニメーション効果である。以下に、それぞれのポイントを説明する。

Effect効果とは、コンポーネントに対する画像効果のことである。たとえば、TEditの子にTGlowEffectを配置すると、入力欄にグロー（発光）効果を適用できる。【図18】

このエフェクト効果だが、効果を有効にする条件（トリガー）も指定できる。Effectに用意されたTriggerプロパティを設定すればよい。【図19】

TGlowEffect以外にも、影の効果を生むTShadowEffectや、反射効果を生むTReflectionEffect、波模様効果を生むTWaveEffectなど多彩な効果がある。

次にアニメーション効果だが、これはプロパティの値を連続的に変化させる仕組みのことである。時間の経過に合わせてプロパティの値を変化させることで、動きをつけられる。

アニメーション効果は、ソースコードを使用して任意のタイミングで開始／終了したり、Effectと同様、トリガーにより実行できる。

アニメーション効果の代表的なコンポーネントは、TFloatAnimationである。これは数値型プロパティに関連付けることで、数値を連続的に変化させられるコンポーネントである。【図20】

シンプルなアニメーション効果の例として、フォーム上に貼り付けたコンポーネントの移動を説明する。

フォーム上にボタンを配置し、ボタンのPositon.Xプロパティに対し、「TFloatAnimationの新規作成」を選択する。するとFloatAnimationに対するプロパティ設定ができるので、StartFromCurrentプロパティをTrueに、StopValueプロパティを300に、Durationプロパティを2に、最後にTriggerプロパティを”IsPress = True”に設定する。

プログラムを実行してボタンをクリックすると、ボタン自体が右側にアニメーションで移動できる。【図21】

このようにいろいろなプロパティ値の操作で、プログラムからアニメーション効果を実現できる。

Effect効果とアニメーション効果をそれぞれ説明したが、もちろんこれらの組み合わせも可能である。最後に、組み合わせた例を説明する。

発光効果であるTGrowEffectの発光度合いは、Softnessプロパティで設定できるが、このSoftnessプロパティに対し、TFloatAnimationを設定できる。【図22】

ボタンをクリックした時、エラーチェックとして項目がブランクだったら発光するアニメーションを実行するよう

図6

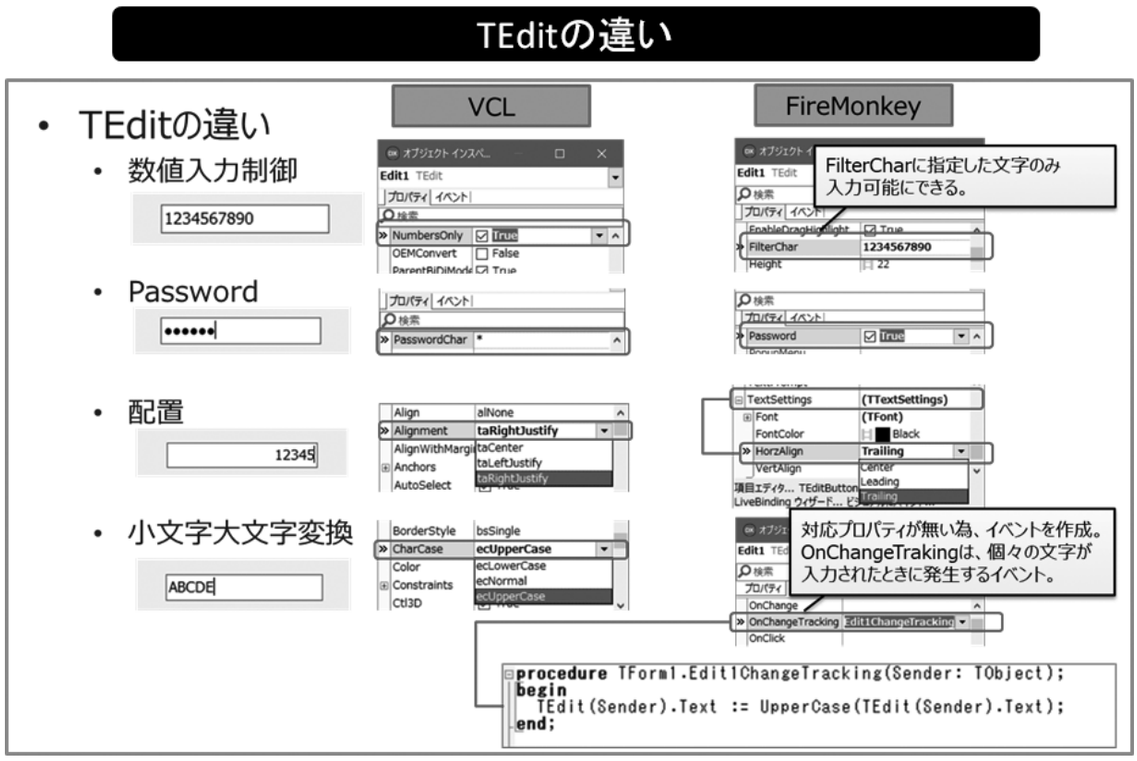


図7

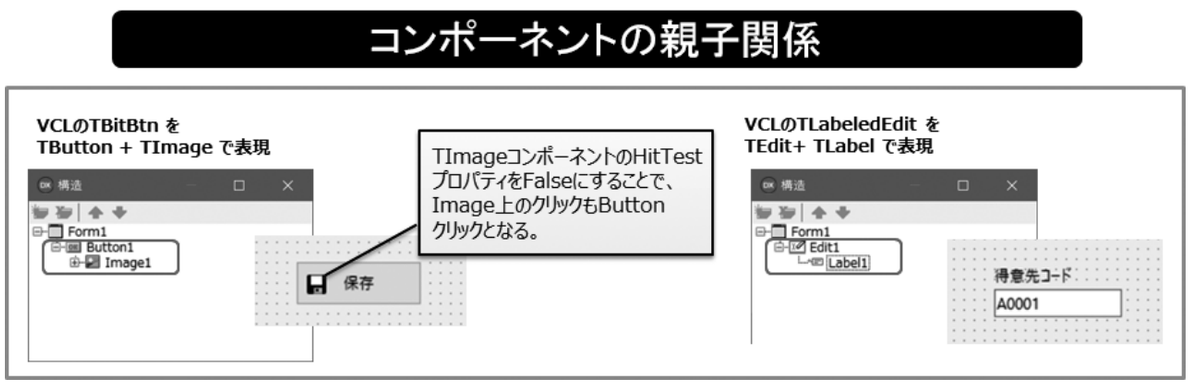
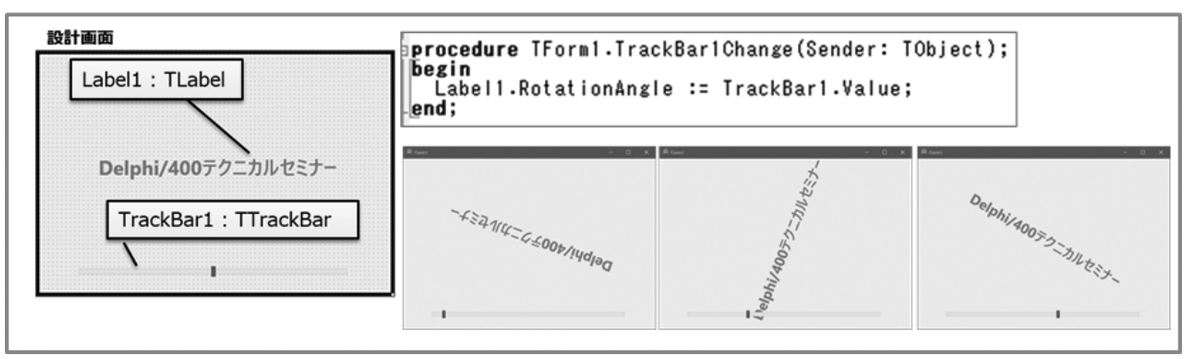


図8



な場合は、【ソース 1】で実現できる。たったこれだけの処理で、エラー発生時にエラー項目に対してのブリンク（点滅）処理を実現できる。

5. おわりに

本稿では、PC アプリケーション開発に役立つ FireMonkey フレームワークの基本を説明してきた。Windows 機能をフル活用したい場合には、VCL 開発が向いているが、視覚的効果が必要なビジュアルアプリケーションを開発する場合は、FireMonkey が適切である。

目的や用途によって VCL と FireMonkey を使い分けることで、これまで以上に Delphi/400 で開発できるアプリケーションの幅を広げられるはずである。

M

図9

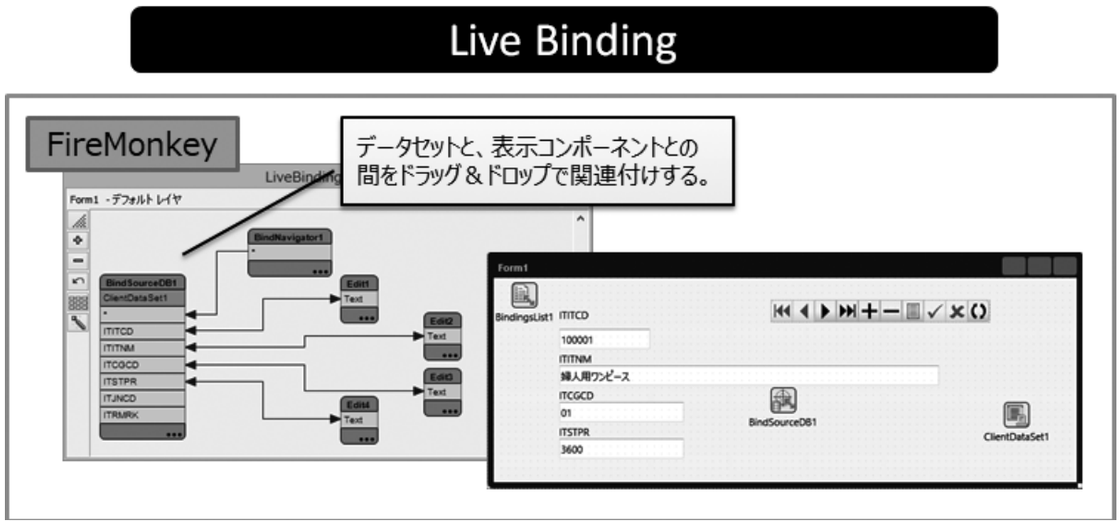


図10

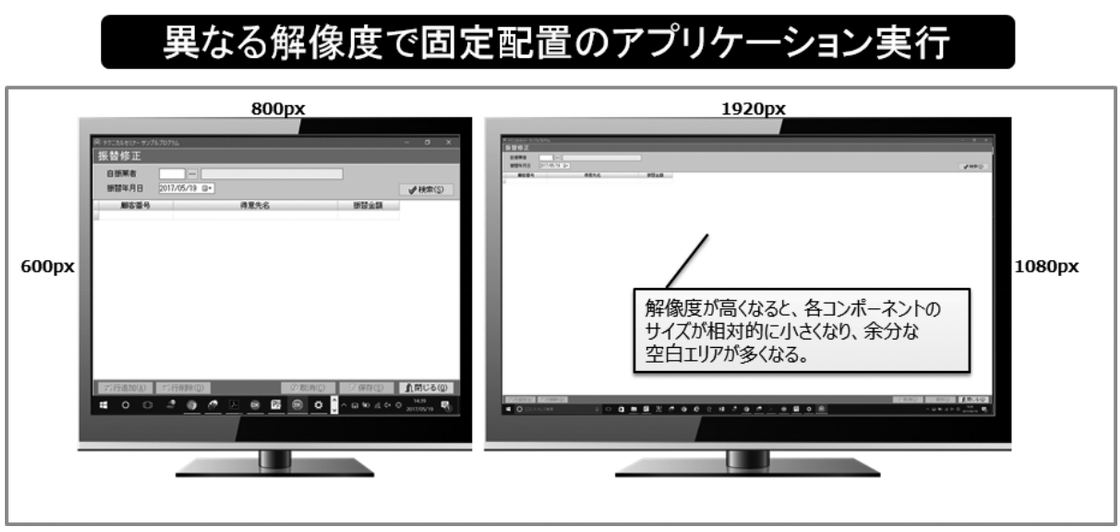


図11

図12

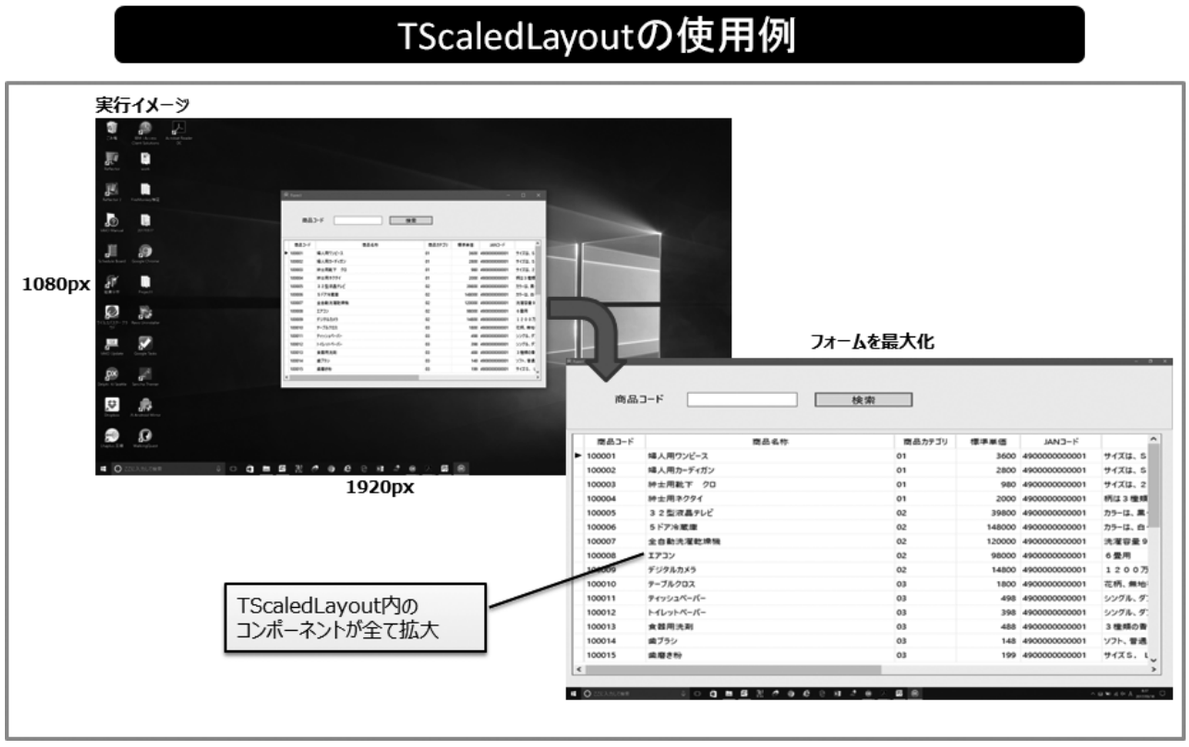


図13

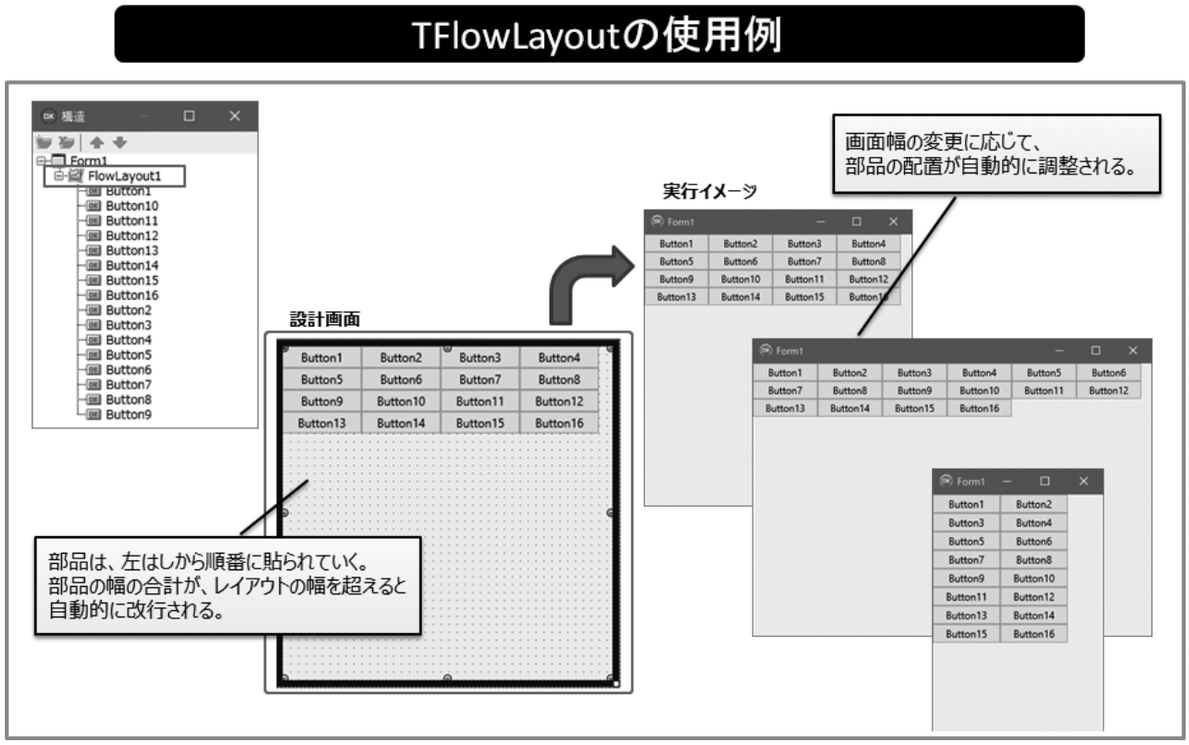


図14

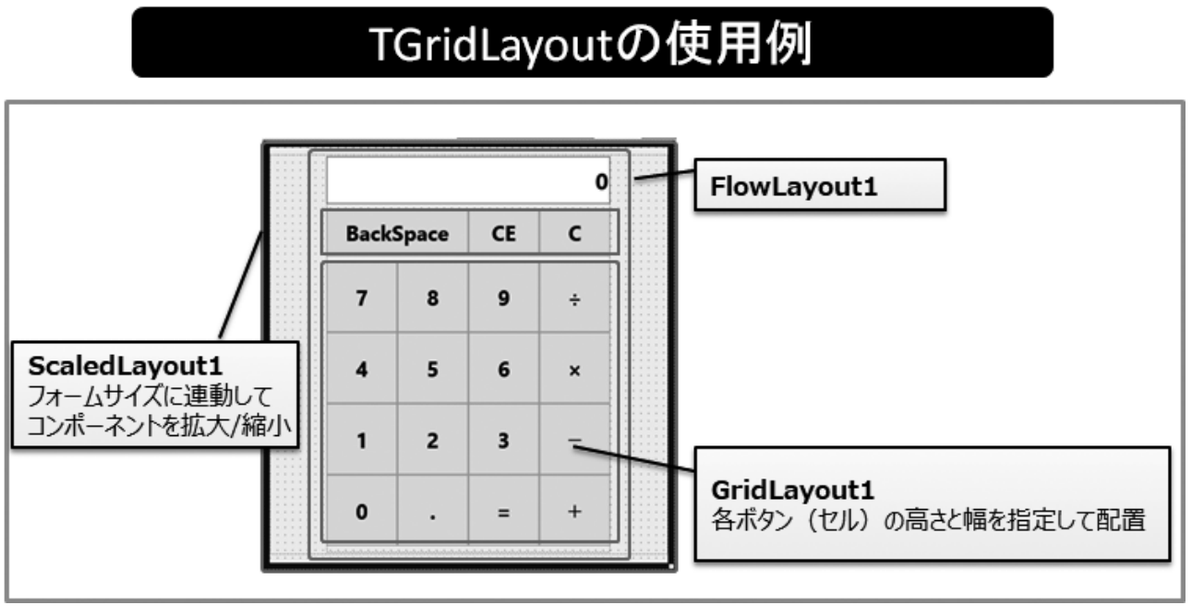


図15

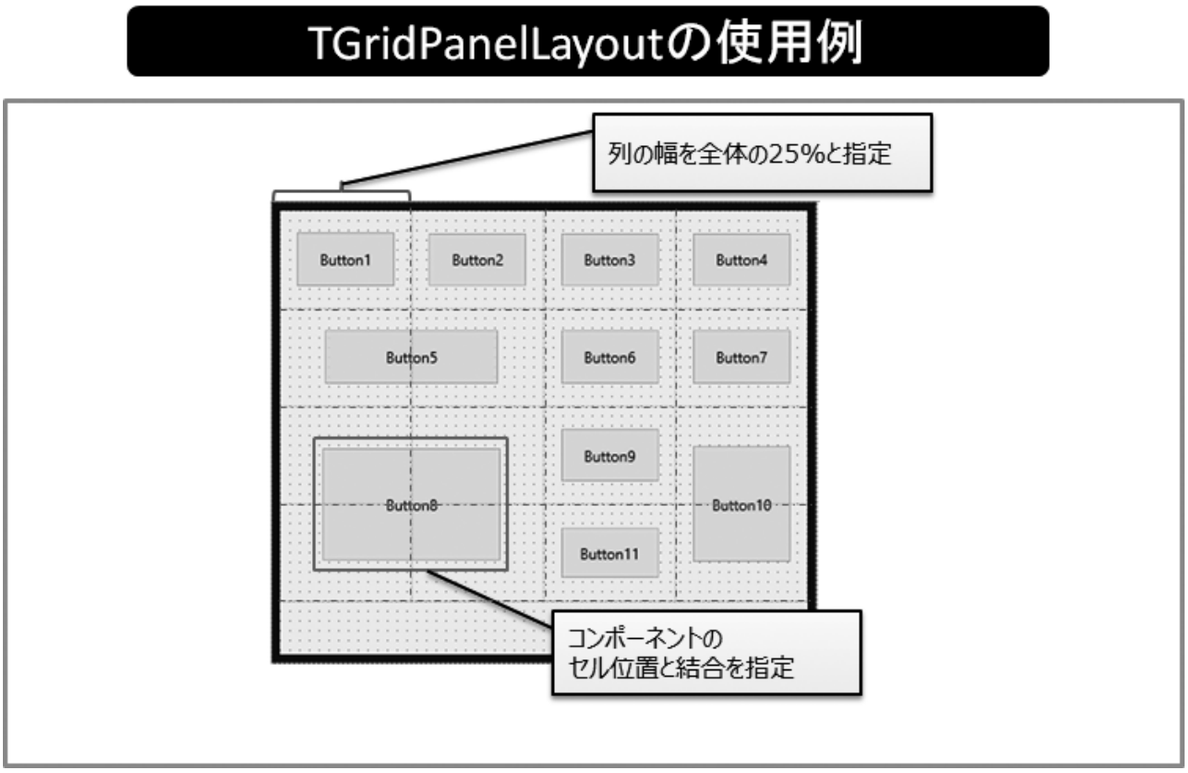


図16

Alignプロパティ

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）
Client	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
Contents	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

図17

MarginsとPadding

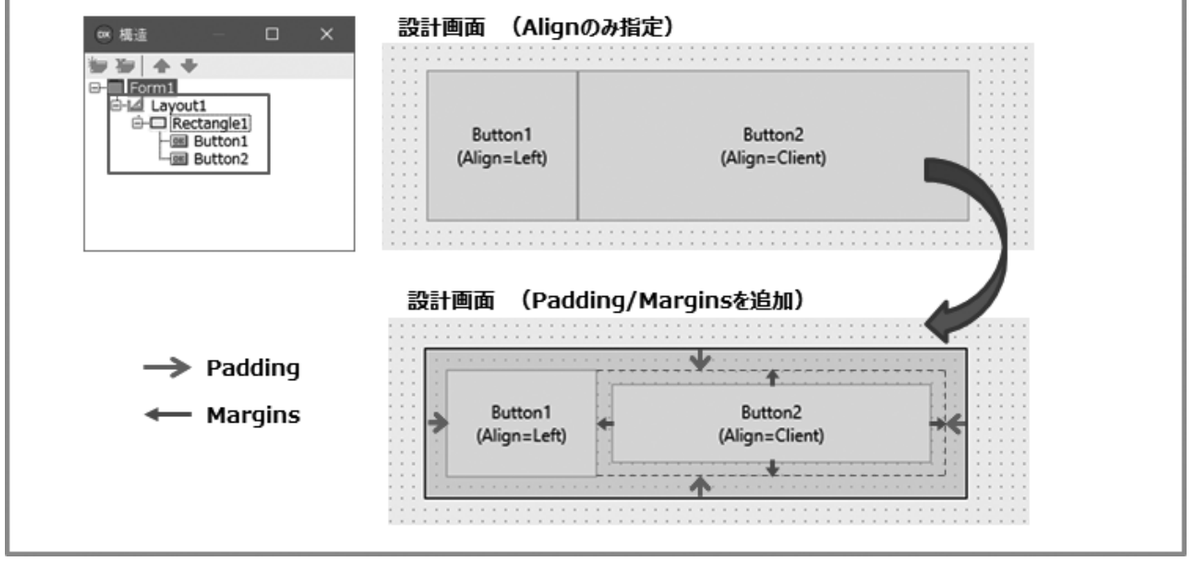


図18

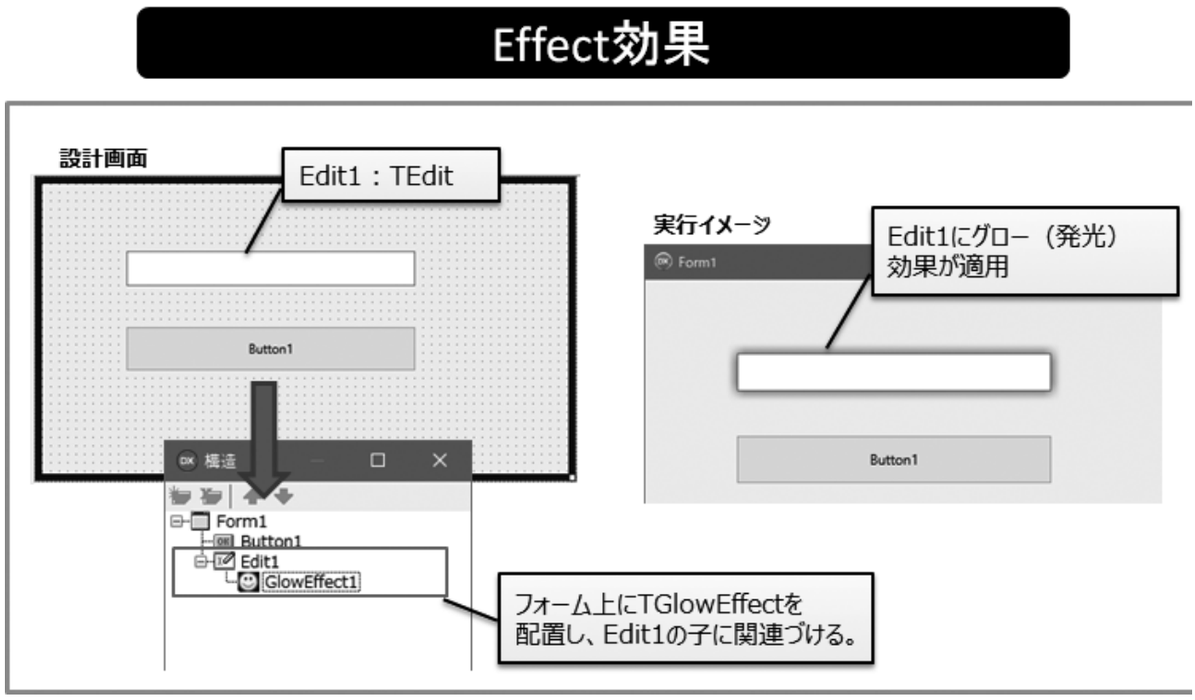


図19

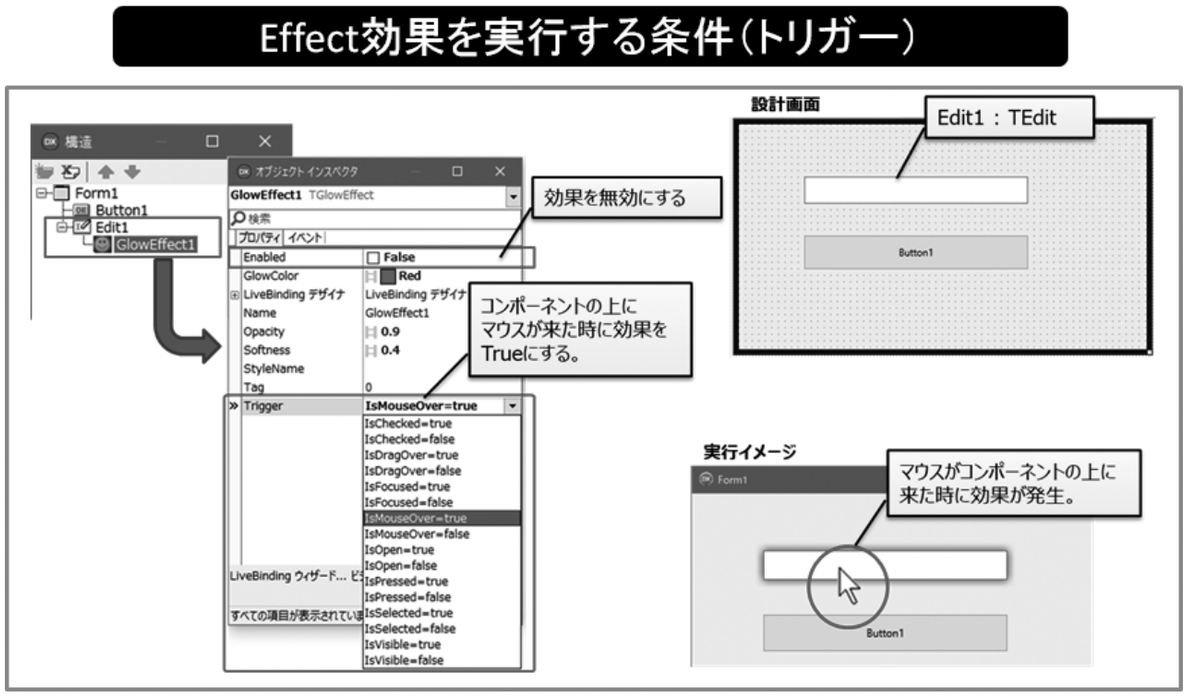


図20

TFloatAnimation

TFloatAnimation

主なプロパティ

プロパティ	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

図21

アニメーション効果　実行例

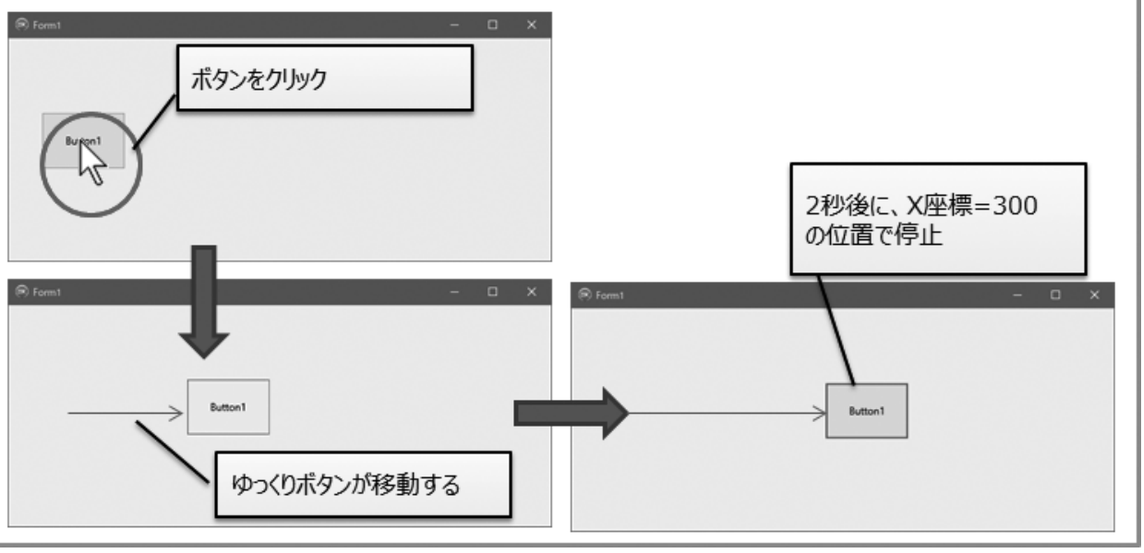
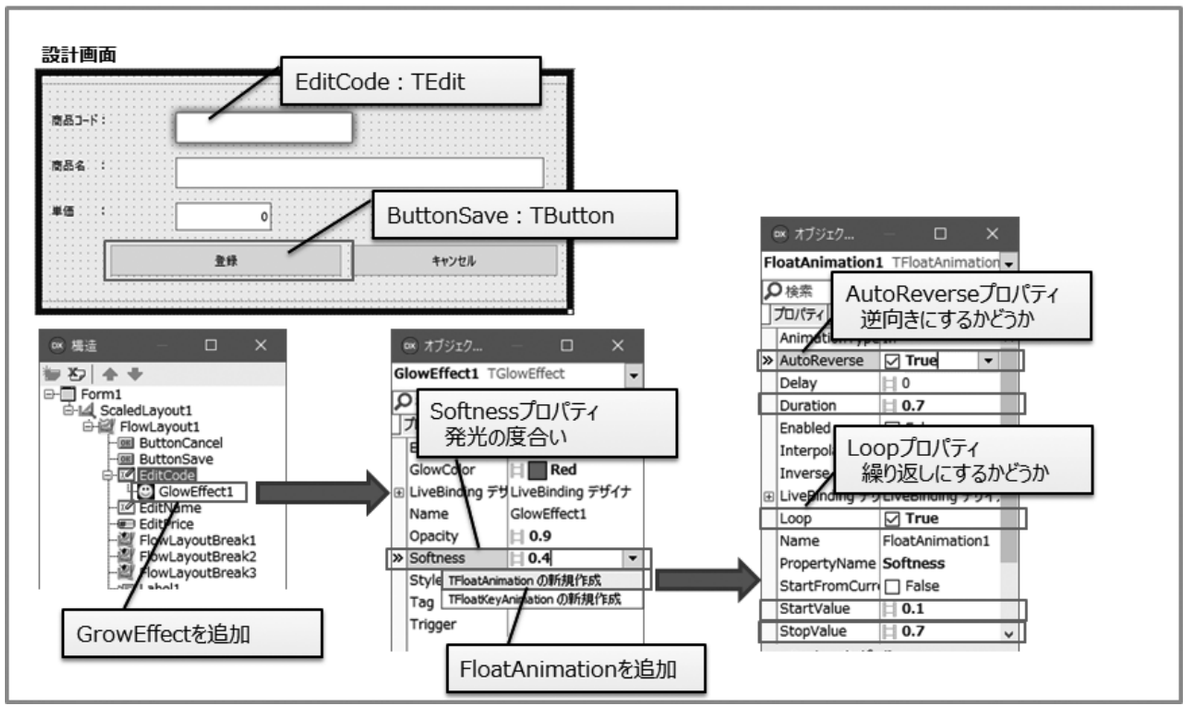


図22

Effect効果とアニメーション効果の組み合わせ



ソース1

登録ボタンのOnClick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;
```