「フォームアニメーションで満足度向上！ モバイルアプリ改善術」

第32回 エンバカデロ・デベロッパーキャンプ

RAD Studio 勉強会 大阪
毛利　春幸 (Haruyuki Mohri)

embarcadero DEVELOPER CAMP

はじめに

- モバイルアプリ制作でのユーザーからの要望を聞いていると、動きのある画面展開が必須になっています。作るだけで評価されていた時代とは違い、画面デザイン含めた動きの要望が数多くなったモバイル開発。
「エンドユーザーから見た商品力」を高めるためのテクニックが必要とされています。フォームをただShowしたりタブを配置するだけではない、こだわりのあるアプリを構築するためのDelphi/C++Builder固有のテクニックを、こうした要望を実現するために悪戦苦闘した経験をもとに解説します。

アジェンダ

- 画面切替について
  - 別画面切替時でのアニメーションについて
- 影付け
  - TGlowEffect, TShadowEffectを使って雰囲気を出す

画面切替について

```
void __fastcall TForm1::Button1Click(TObject *Sender)
{
    Form2->ShowModal();
}
```

Windowsアプリケーションでの画面切替の場合Show()などを使って切替を行っていた

同じようにiOSでビルド実行すると違和感ないでしょうか？

今までの画面切替(iOSで実装した場合)

```
void __fastcall TForm1::Button1Click(TObject *Sender)
{
    Form2->ShowModal();
}
```

動作としては正常でプログラムとしても間違いはありません

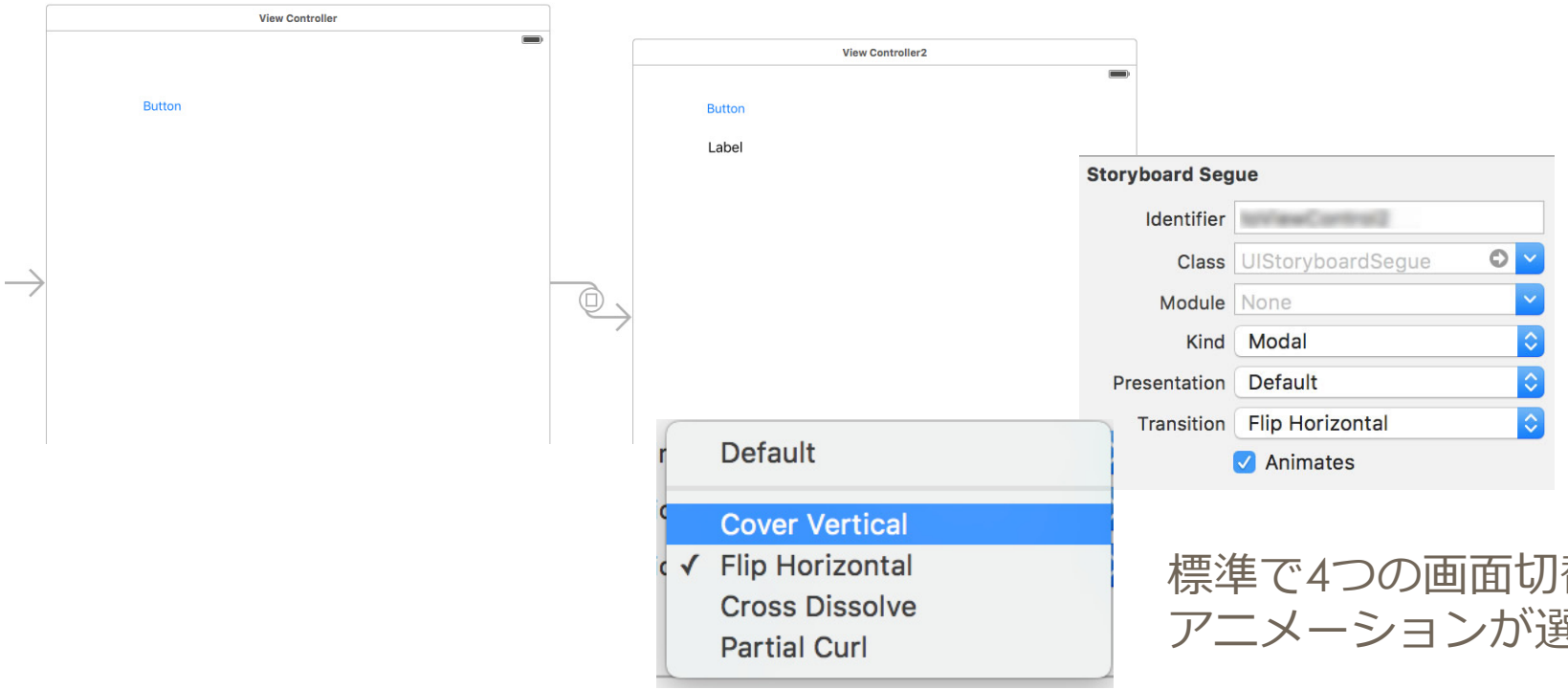
機能としては満足している

業務として使うのでこれで十分

(ex1)

Xcodeでの画面切替

Interface Builder



標準で4つの画面切替時の
アニメーションが選択できます

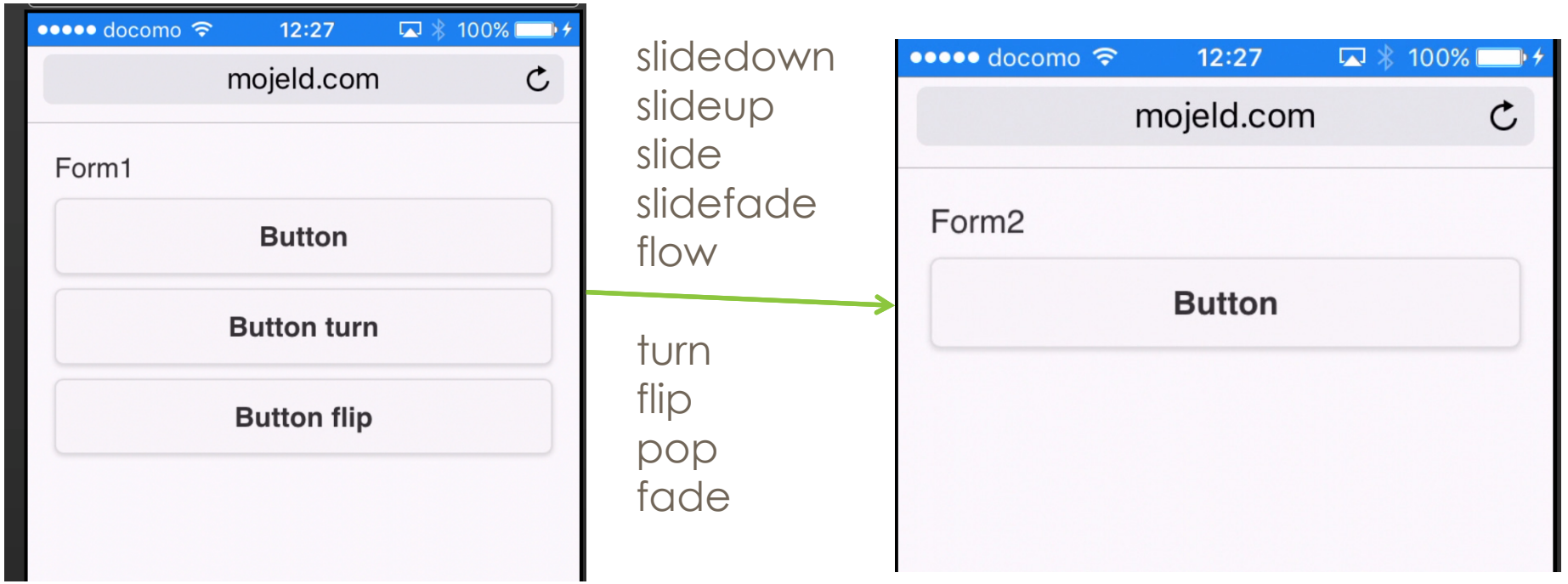
jQuery Mobileでの画面切替

```
<div id="Form1" data-role="page">
  <div role="main" class="ui-content">
    Form1<br />
  <a href="#Form2" data-role="button" data-transition="slide" direction="reverse">Button</a>
</div>
</div>

<div id="Form2" data-role="page">
  <div role="main" class="ui-content">
    Form2<br />
  <a href="#Form1" data-role="button" data-transition="slideup" direction="reverse">Button</a>
  </div>
</div>
```

data-transitionで設定します

(slidedown, slideup, slide, slidefade, flow, turn, flip, pop, fade)

jQuery Mobileでの画面切替2



画面切替時にスライドアニメーション

Xcodeと jQuery Mobileを例としてあげましたが 画面切替時にアニメーションを使う事が
当然のようになっていて、エンドユーザーからも同じような要望があります

画面切替時にアニメーションにする事で使う側の人に画面が切り替わった事を意識させているのでは？

項目が似ている画面などはアニメーション切替を使う事は有効でした

画面切替でShowを減らすとユーザーは見やすい?

エンドユーザーは画面切替時アニメーションでスライドするのを他のアプリで見慣れている

自身の画面切替時に対しての意識を変える事にしました。
(Form2.Showすればいいシーンと動きをつけないといけないシーンを分ける)

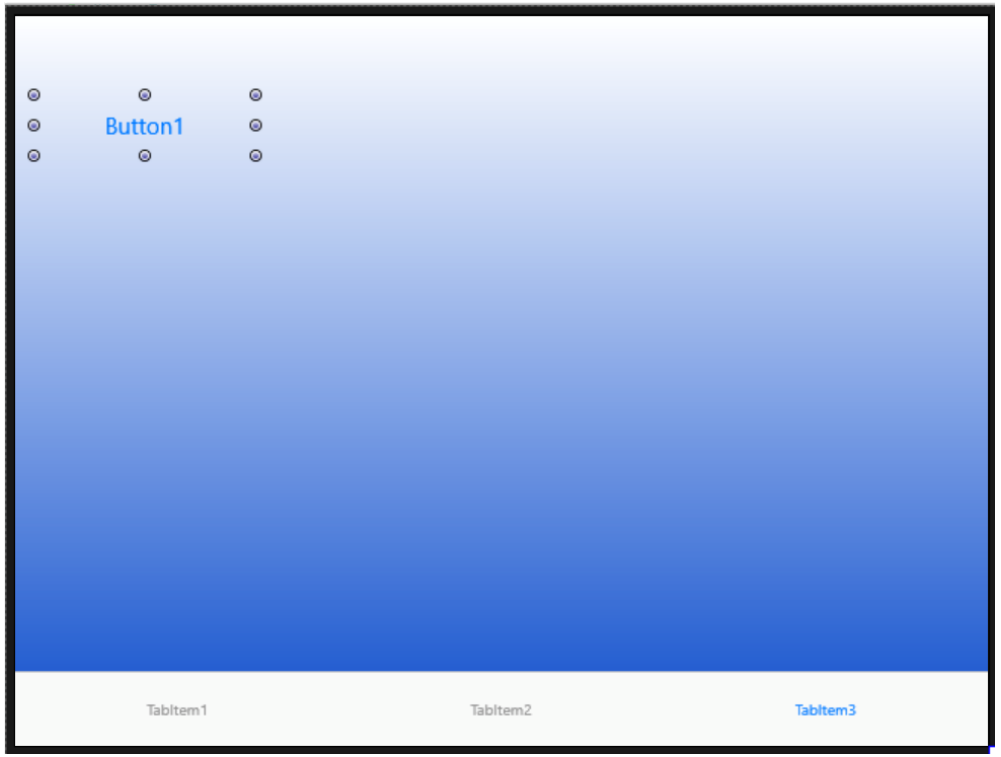
TFloatAnimationを 使って動きをつけていく

Delphi / C++BuilderでShow使わずに画面切替を行う

画面切替を行うにはたくさんの方法があり最終的に動けば全て正解だと思います
その中で、僕が行った方法をいくつかご紹介したいと思います

TTabControlで画面切替1

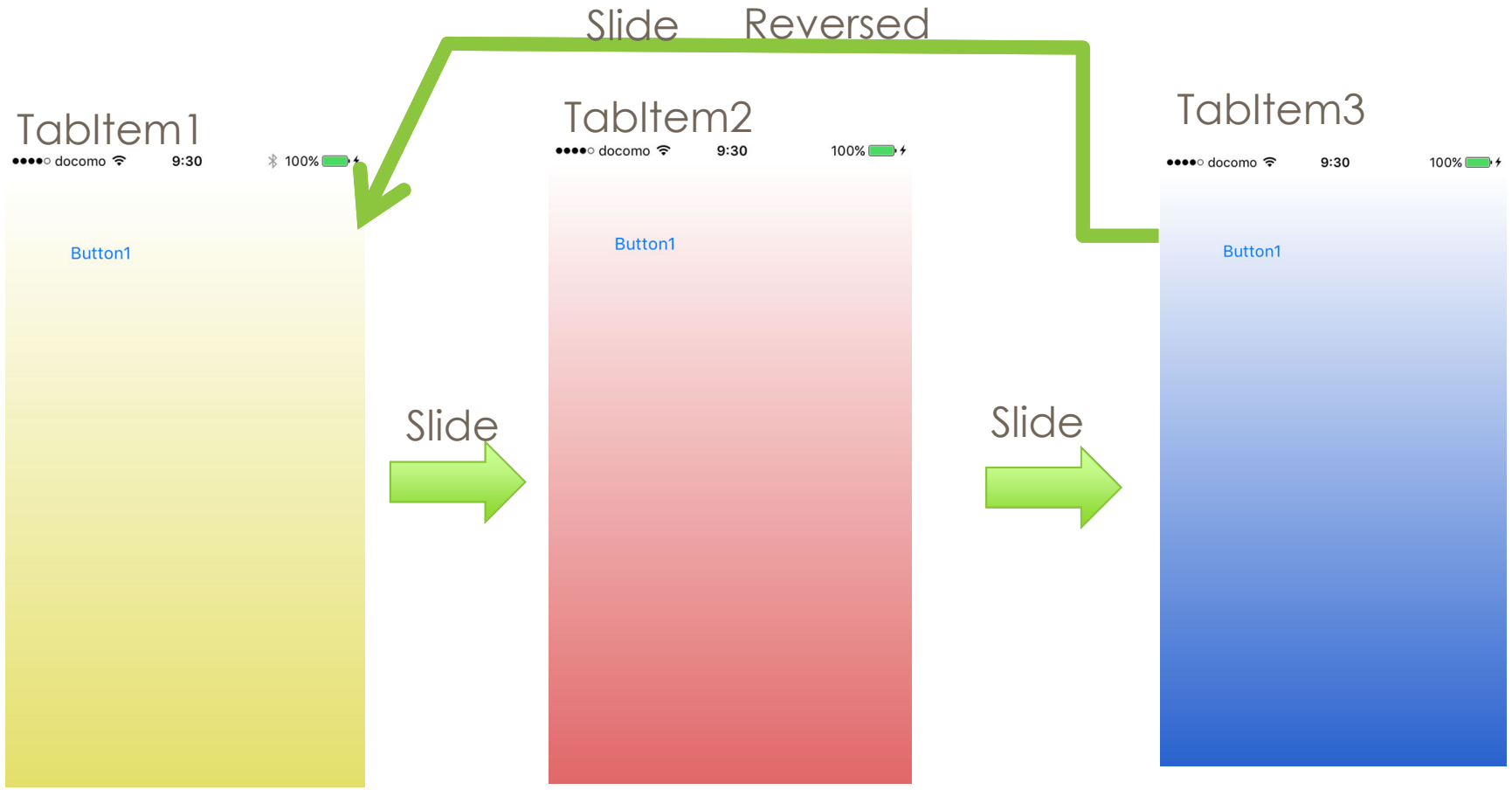
SetActiveTabWithTransition メソッドを使ったいちばんシンプルな方法です



下のタブボタンを見えなくすれば
画面が切り替わったように思える

(ex2)

TTabControlで画面切替1.1



(ex2)

TTabControlで画面切替2

```
void __fastcall TForm1::Button1Click(TObject *Sender)
{
    TabControl1->SetActiveTabWithTransition(TabItem2, TTabTransition::Slide, TTabTransitionDirection::Normal);
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Button2Click(TObject *Sender)
{
    TabControl1->SetActiveTabWithTransition(TabItem3, TTabTransition::Slide, TTabTransitionDirection::Normal);
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Button3Click(TObject *Sender)
{
    TabControl1->SetActiveTabWithTransition(TabItem1, TTabTransition::Slide,
TTabTransitionDirection::Reversed);

}
```

実装したのはこれだけです

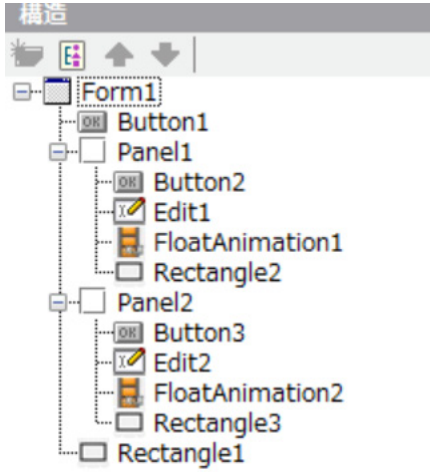
TTabControlで画面切替3

```
enum class DECLSPEC_DENUM TTabTransition : unsigned char { None, Slide, ttNone
_DEPRECATED_ATTRIBUTE3("Use TTabTransition.None")  = 0x0, ttSlide _DEPRECATED_ATTRIBUTE3("Use
TTabTransition.Slide")  = 0x1 };


TTabTransition = (None, Slide); //Delphi
TTabTransitionDirection = (Normal, Reversed);
```

TTabTransitionはスライドする/しないの２択なのですが今後増えれば嬉しいです

複数のパネルなどで画面切替1

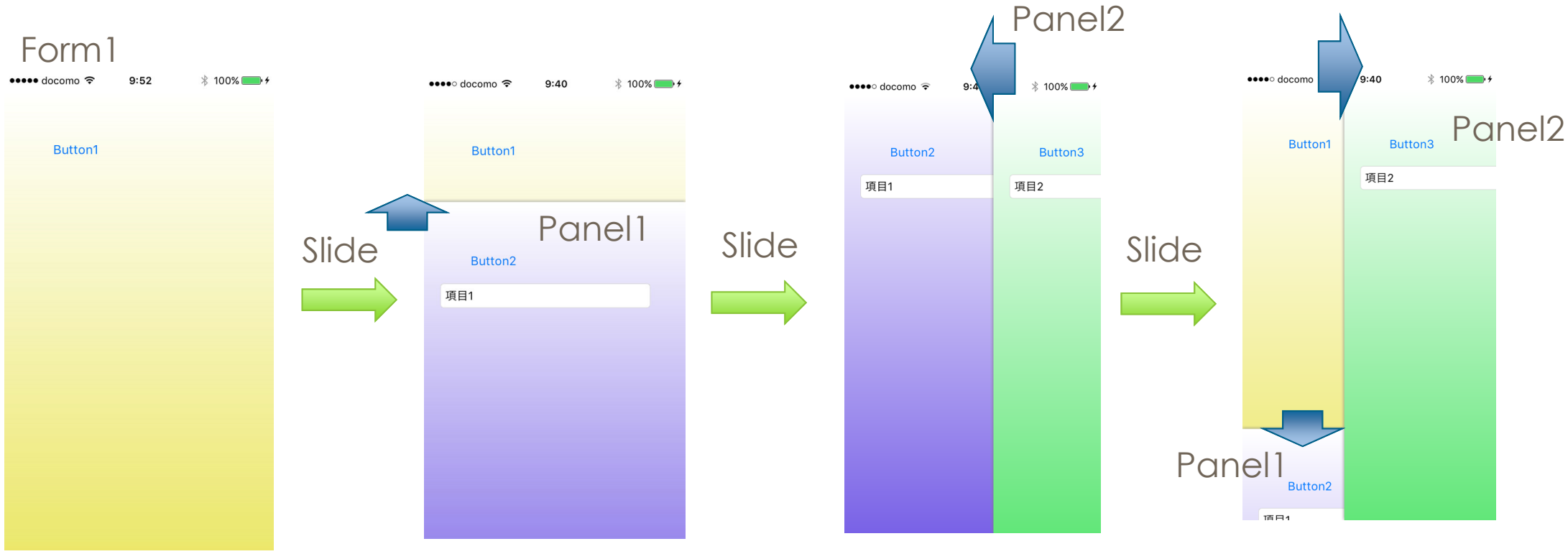
TPanelや TLayout を使って画面切替



TPanelや Tlayout をTFloatAnimationを使って切替に見せかける

(ex3)

複数のパネルなどで画面切替1.1



(ex3)

複数のパネルなどで画面切替2

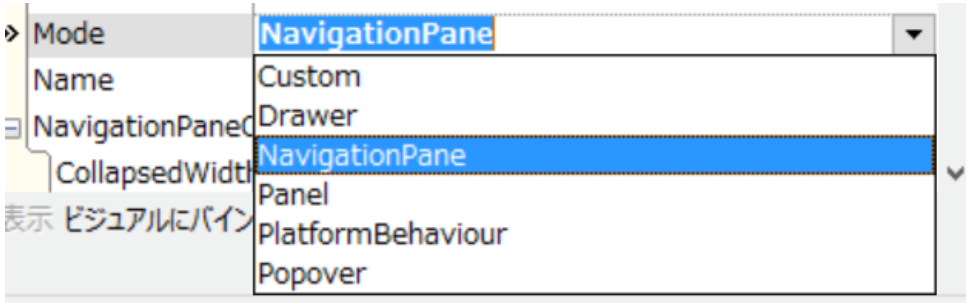
```
void __fastcall TForm1::Button1Click(TObject *Sender)
{
    FloatAnimation1->StartValue = Panel1->Position->Y;
    FloatAnimation1->StopValue  = 0;
    FloatAnimation1->Start();
}
void __fastcall TForm1::Button2Click(TObject *Sender)
{
    FloatAnimation2->StartValue = Panel2->Position->X;
    FloatAnimation2->StopValue  = 0;
    FloatAnimation2->Start();
}
void __fastcall TForm1::Button3Click(TObject *Sender)
{
    FloatAnimation1->StartValue = 0;
    FloatAnimation1->StopValue  = this->Height + 10;
    FloatAnimation1->Start();
    FloatAnimation2->StartValue = 0;
    FloatAnimation2->StopValue  = this->Width + 10;
    FloatAnimation2->Start();
}
```

コードはこれだけです

TMultiViewで画面切替1

TMultiViewコンポーネントを使っても同じように画面の切り替えできます

Modeプロパティをうまく使えば簡単に別Formのような扱いのできる画面が作れます




```
enum class DECLSPEC_DENUM TMultiViewMode : unsigned char { PlatformBehaviour, Panel, Popover, Drawer, Custom, NavigationPane };
```

PlatformBehaviour, Panel, Popover, Drawer, Custom, NavigationPane 6つモードがあります

(ex5)

TMultiViewで画面切替2

PlatformBehaviour, Panel, Popover, Drawer, Custom, NavigationPane

モード	表示
Drawer	ドローワ（プッシュ/オーバーラップ）
Panel	ドッキングされたパネル
PlatformBehaviour	プラットフォーム依存の動作
Popover	ポップアップ メニュー
NavigationPane	ナビゲーション ペイン
Custom	カスタム

Popover, Drawer, NavigationPaneは動きがある別画面を作れます

TMultiViewで画面切替3

PlatformBehaviour, Panel, Popover, Drawer, Custom, NavigationPane

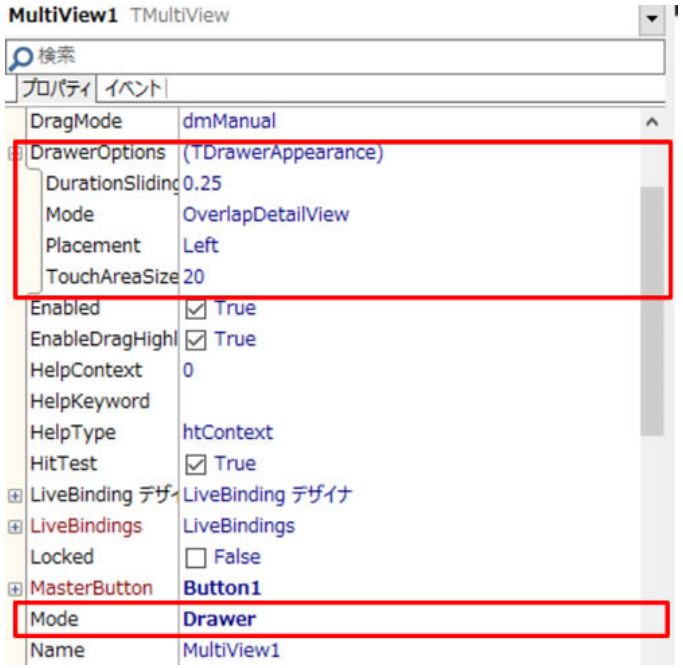
Mode = PlatformBehaviourの場合

デバイスの種類	デバイスの向き	マスタ ペインの表示
スマートフォン	横、縦	Drawer（プッシュ/オーバーラップ）
タブレット	横	ドッキングされたPanel
タブレット	縦	Drawer（プッシュ/オーバーラップ）
Windows 10		NavigationPane(ナビゲーションペイン)
Windows 8以前		ドッキングされたPanel
OS X		

TMultiViewで画面切替2.1

TMultiViewMode::Drawer を使って画面切替

iPhoneの場合、this->ClientWidth=320なのでMultiView1 Width=320にする



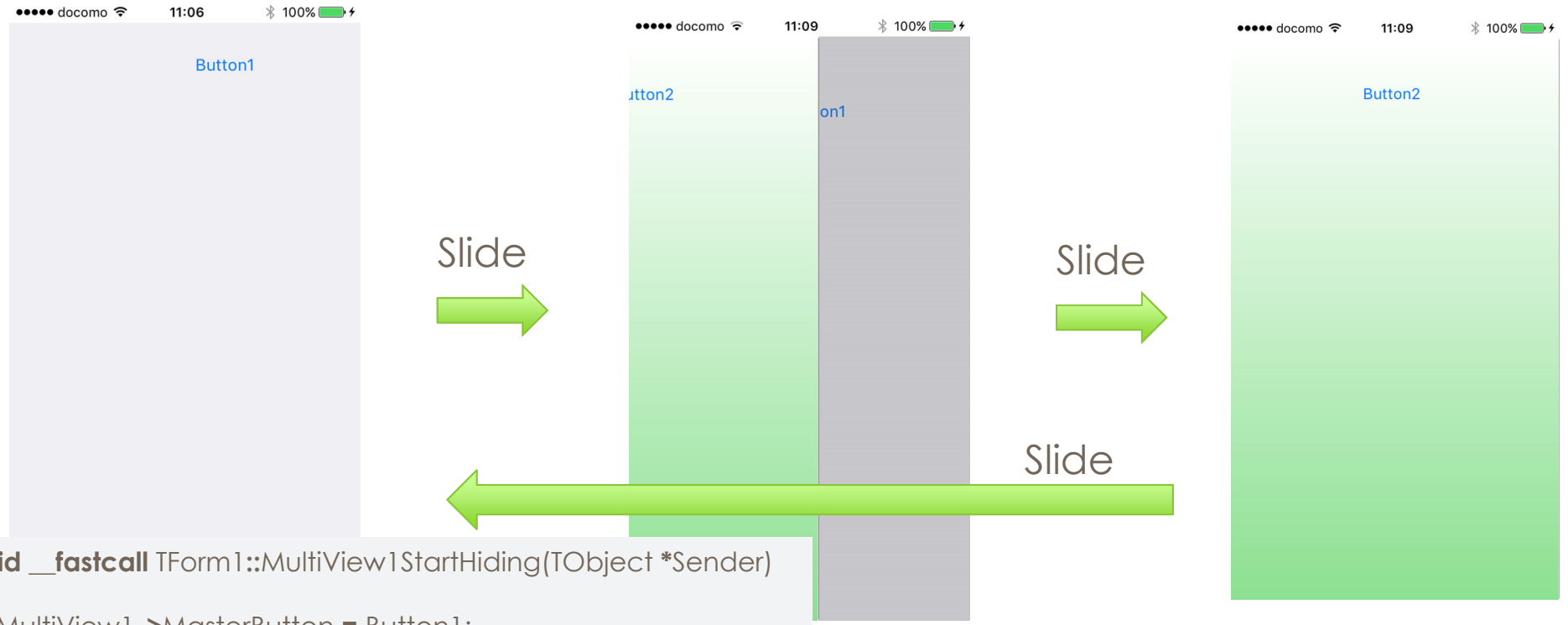

```
void __fastcall TForm1::MultiView1Shown(TObject *Sender)
{
    MultiView1->MasterButton = Button2;
}
//---------------------------------------------------------------------------

void __fastcall TForm1::MultiView1StartHiding(TObject *Sender)
{
    MultiView1->MasterButton = Button1;
}
```

(ex5)

TMultiViewで画面切替2.2

TMultiViewMode::Drawer を使って画面切替

```
void __fastcall TForm1::MultiView1Shown(TObject *Sender)
{
    MultiView1->MasterButton = Button2;
}
```




```
void __fastcall TForm1::MultiView1StartHiding(TObject *Sender)
{
    MultiView1->MasterButton = Button1;
}
```

TMultiViewで画面切替3

PlatformBehaviour, Panel, Popover, Drawer, Custom, NavigationPane

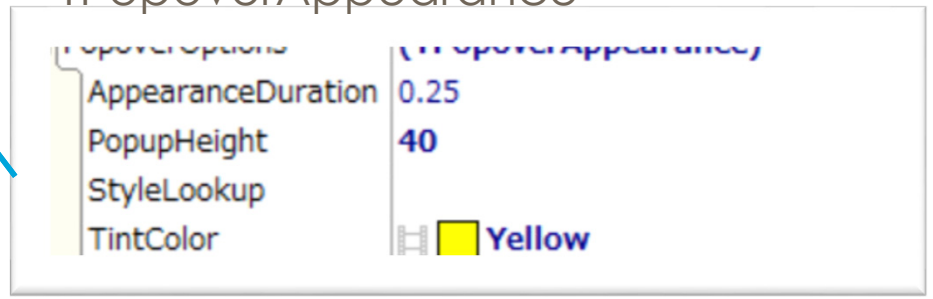
TSplitViewAppearance

TDrawerAppearance

TNavigationPaneAppearance

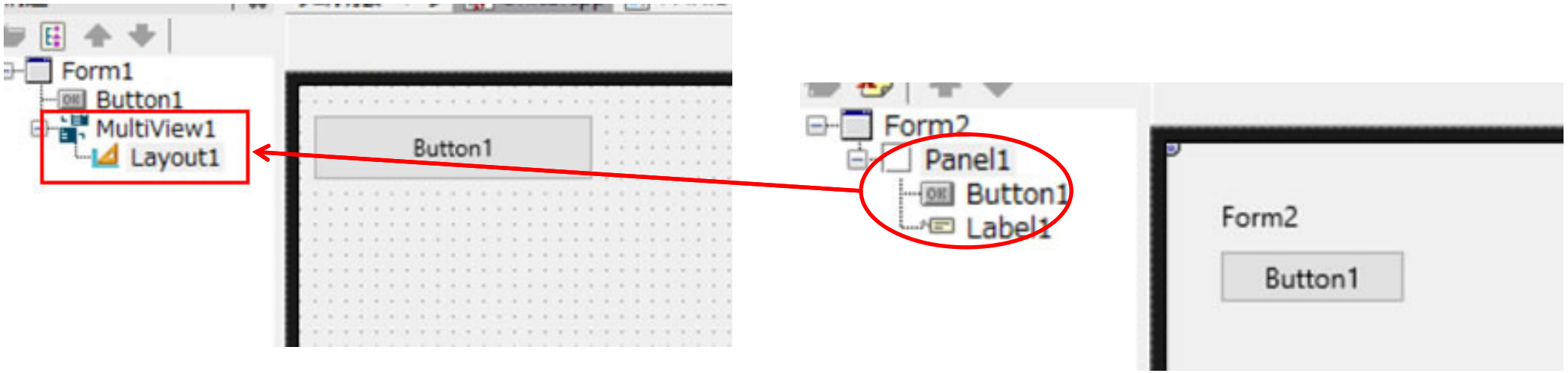
TShadowAppearance

TPopoverAppearance



設計上フォームを分割したい1

Form1, Form2を使ってスライドさせたい場合



Form2の部品をレイアウトに貼ればOK

(ex6)

設計上フォームを分割したい2

Form1, Form2を使ってスライドさせたい場合

```
extern "C" int FMXmain()
{
    try
    {
        Application->Initialize();
        Application->CreateForm(__classid(TForm1), &Form1);
//      Application->CreateForm(__classid(TForm2), &Form2);
        Application->Run();
    }
. . .
```

(ex6)

設計上フォームを分割したい3

MultiView1->MasterButton = Button1;なので

```
void __fastcall TForm1::MultiView1StartShowing(TObject *Sender)
{
    MultiView1->MasterButton = Button1;
    if (Form2 == nullptr )
    {
        Form2 = new TForm2(this);
    }
    Layout1->AddObject(Form2->Panel1);
    Form2->Panel1->Align = TAlignLayout::Client;
    MultiView1->MasterButton = Form2->Button1;
}
. .
```

Layout1->AddObjectに入ればForm2をスライドしたように見える

(ex6)

設計上フォームを分割したい4

MultiView1->MasterButton = Button1;なので

```
void __fastcall TForm1::MultiView1StartShowing(TObject *Sender)
{
    MultiView1->MasterButton = Button1;
    Form2 = new TForm2(this);
    Layout1->AddObject(Form2->Panel1);
    Form2->Panel1->Align = TAlignLayout::Client;
}
. .
```

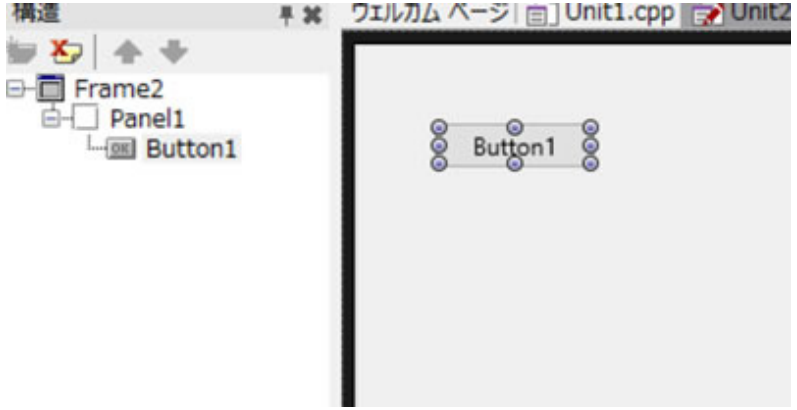
Layout1->AddObjectに入れればForm2をスライドしたように見える

(ex6)

設計上フォームを分割したいフレーム編1

Form1

Frame2(Delphi)



(ex7)

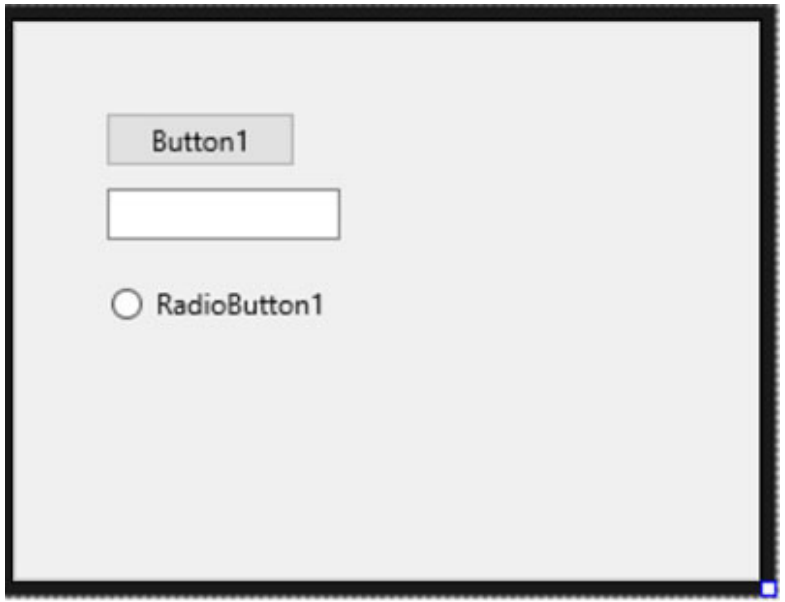
設計上フォームを分割したいフレーム編2

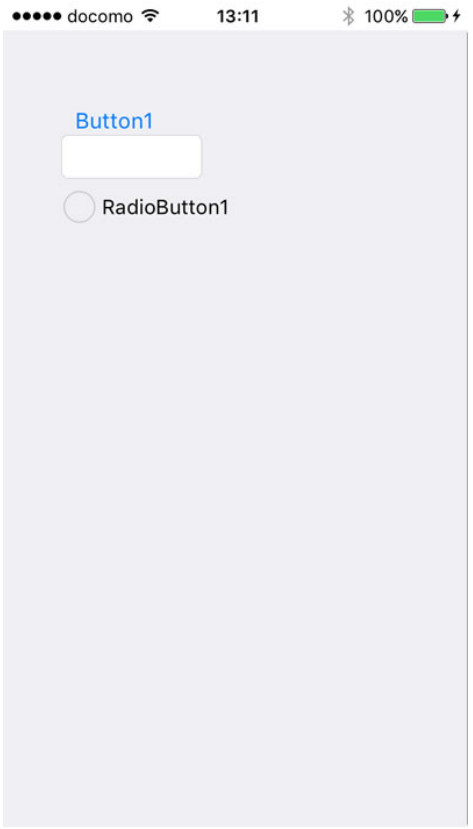
```
void __fastcall TForm1::MultiView1StartShowing(TObject *Sender)
{
    if (Frame2 == nullptr) {
        Frame2 = new TFrame2(this);
    }
    MultiView1->AddObject(Frame2);
    Frame2->Align = TAlignLayout::Client;
    MultiView1->MasterButton = Frame2->Button1;
}
//---------------------------------------------------------------------------
void __fastcall TForm1::MultiView1StartHiding(TObject *Sender)
{
    MultiView1->MasterButton = Button1;
}
```

(ex7)

設計上フォームを分割したいフレーム編3

Frameを使った場合 設計画面がWindowsデザインのままです





実行するとiOSの
スタイルで表示されます

(ex7)

影1

最近のスマホ画面フラットデザインが多いのであまり気にしなくてもよくなりました。

しかし、影で奥行きをつけて欲しいと言われる事もあります

影2

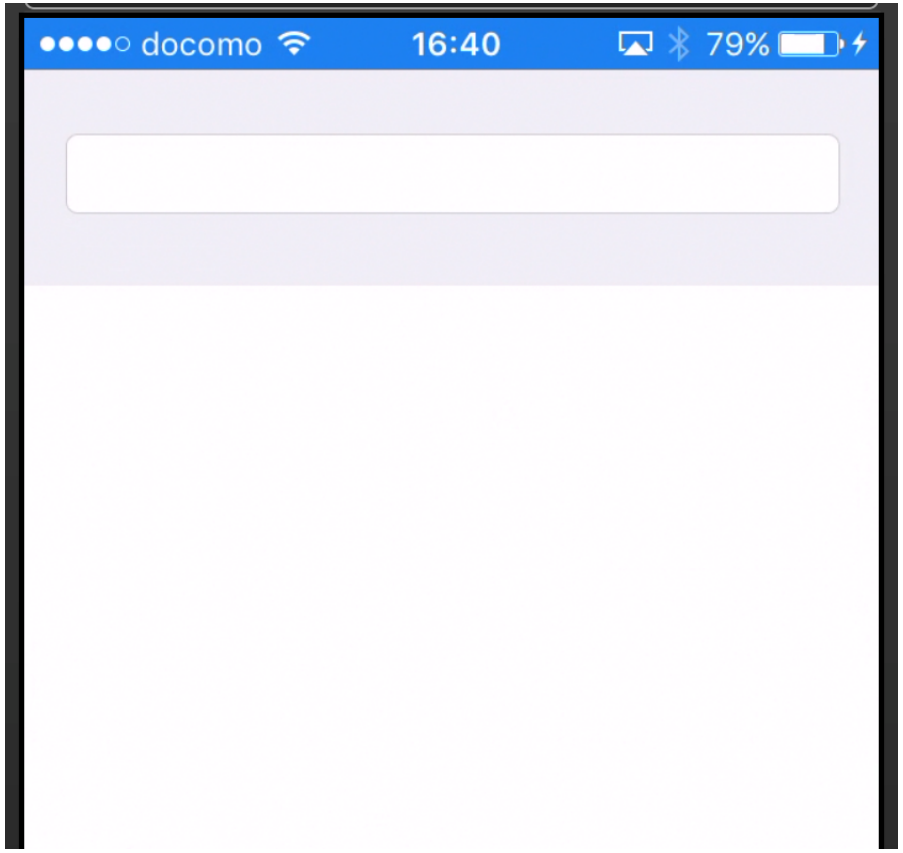
例えばウィンドウズなら

何気なくパネルTopに貼り
TMemo Align = TAlignLayout::Client

(ex4)

影3

このままiOSでビルドすると

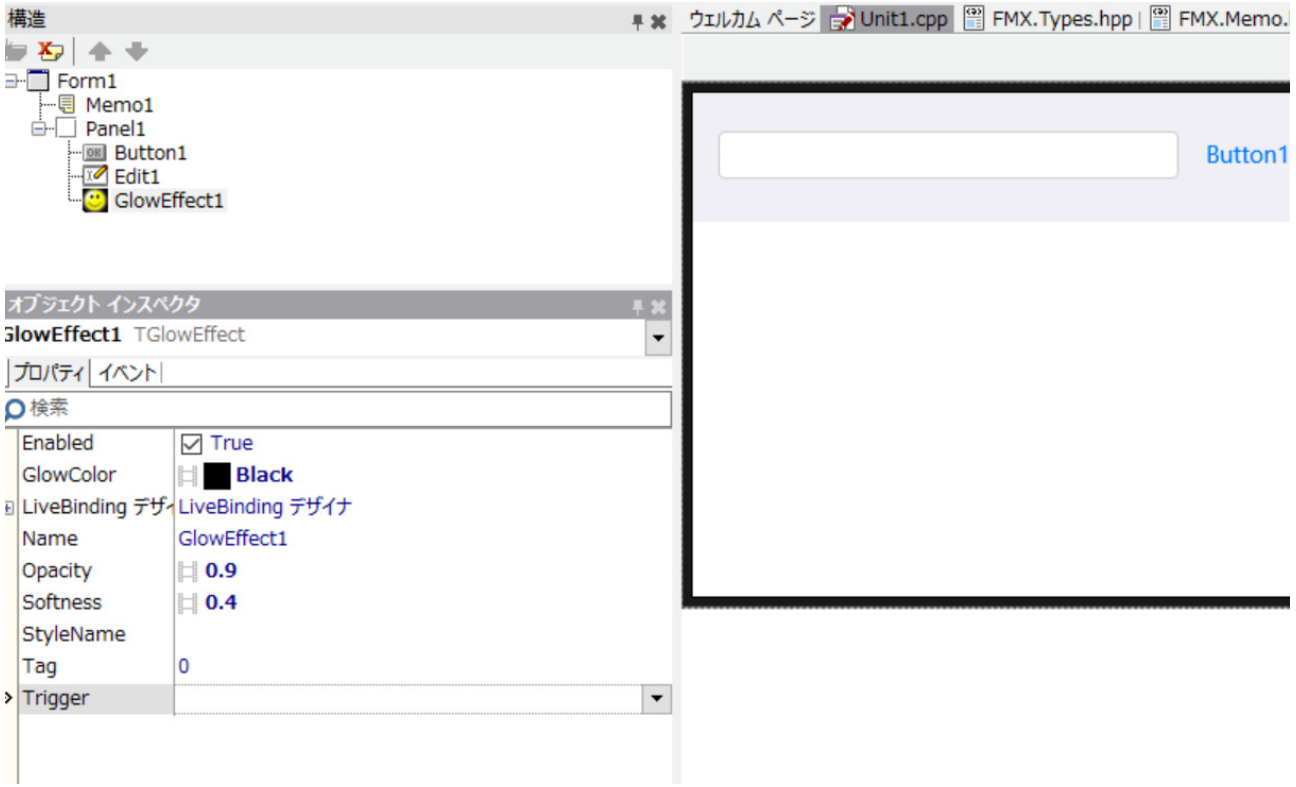


フラットデザインなので問題ないが
パネルとTMemoの境目が見づらい
同色パネルが2つある場合も同じ

ここにエフェクトを使って奥行きを出す

影4

このままiOSでビルドすると



TPanelにTGlowEffect貼った。

反映されていない

(ex4)

影5

FMXは*.fmx上から順番に描画している

```
object Form1: TForm1
  object Panel1: TPanel
    object Edit1: TEdit
    end
    object Button1: TButton
    end
    object GlowEffect1: TGlowEffect
      Softness = 0.400000005960464400
      GlowColor = claBlack
      Opacity = 0.899999976158142100
    end
  end
  object Memo1: TMemo
  end
end
```

↓パネルを書いて

↓エフェクトを書いて

↓メモを書いている

影6

FMXは*.fmx上から順番に描画している

```
object Form1: TForm1
  object Memo1: TMemo
  end
  object Panel1: TPanel
    object Edit1: TEdit
    end
    object Button1: TButton
    end
    object GlowEffect1: TGlowEffect
      Softness = 0.400000005960464400
      GlowColor = claBlack
      Opacity = 0.899999976158142100
    end
  end
end
```

↓メモを書いている

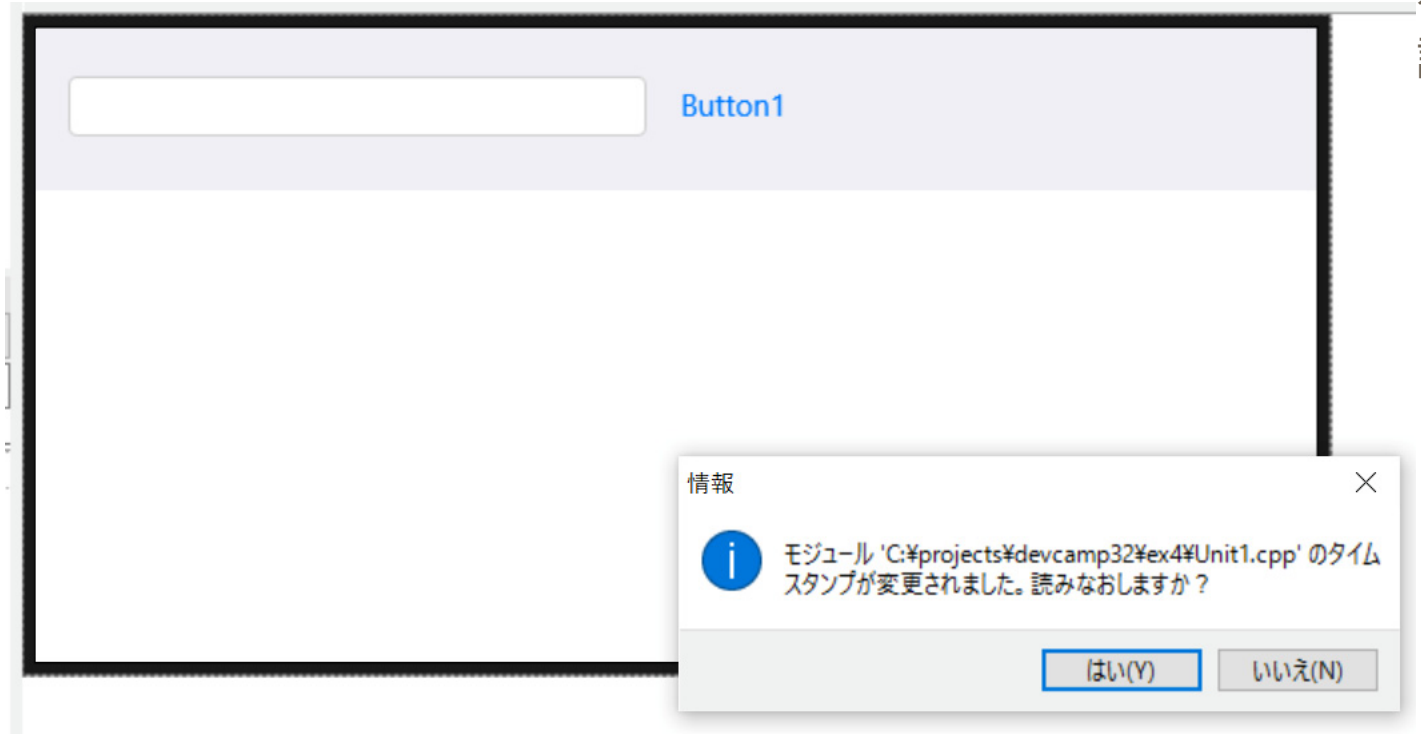
↓パネルを書いて

↓エフェクトを書く

これでパネルが全面に移動される

影7

エディタでfmxファイルを変更すると



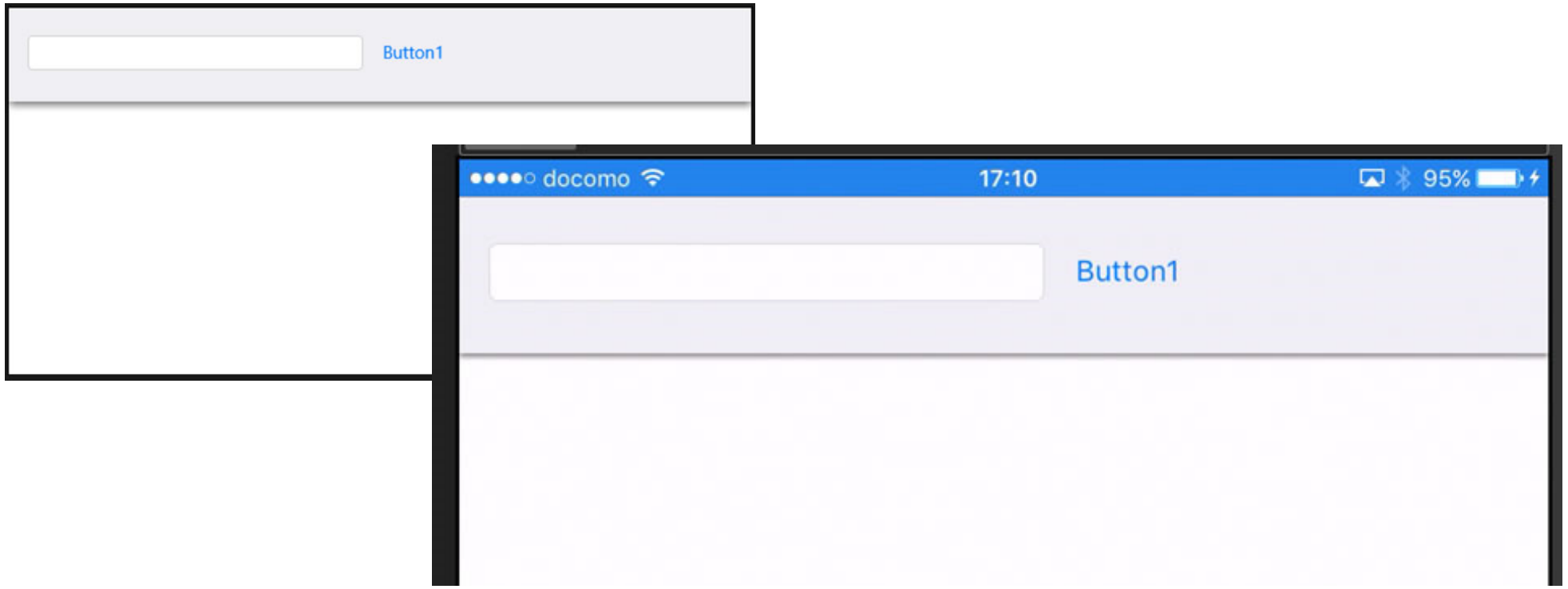
モジュール 'Unit1.cpp' の
タイムスタンプが変更されました。
読みなおしますか？

[はい(Y)] [いいえ(N)]

(ex4)

影8

エディタでfmxファイルを読み直すとTGlowEffect を貼った影が出ます



(ex4)

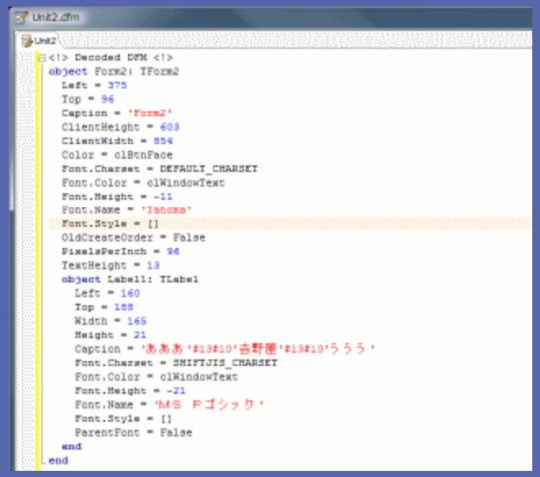
影9

エディタでfmxファイルの書き換え面倒です
http://ht-deko.com/junkbox.html#DFMCONVEXP
ここに 簡単にfmxを入れ替えるChangeCreateOrderツールがあります

・DFMコンバータエキスパート ver1.61 [DownLoad: dfmconv_expert_162.zip (2014/07/25 更新)]

(ガリレオ IDE の Delphi)

ソース表示したフォームのコントロール文字列をデコードしたり、逆にコントロール文字列へエンコードする IDE機能拡張です。

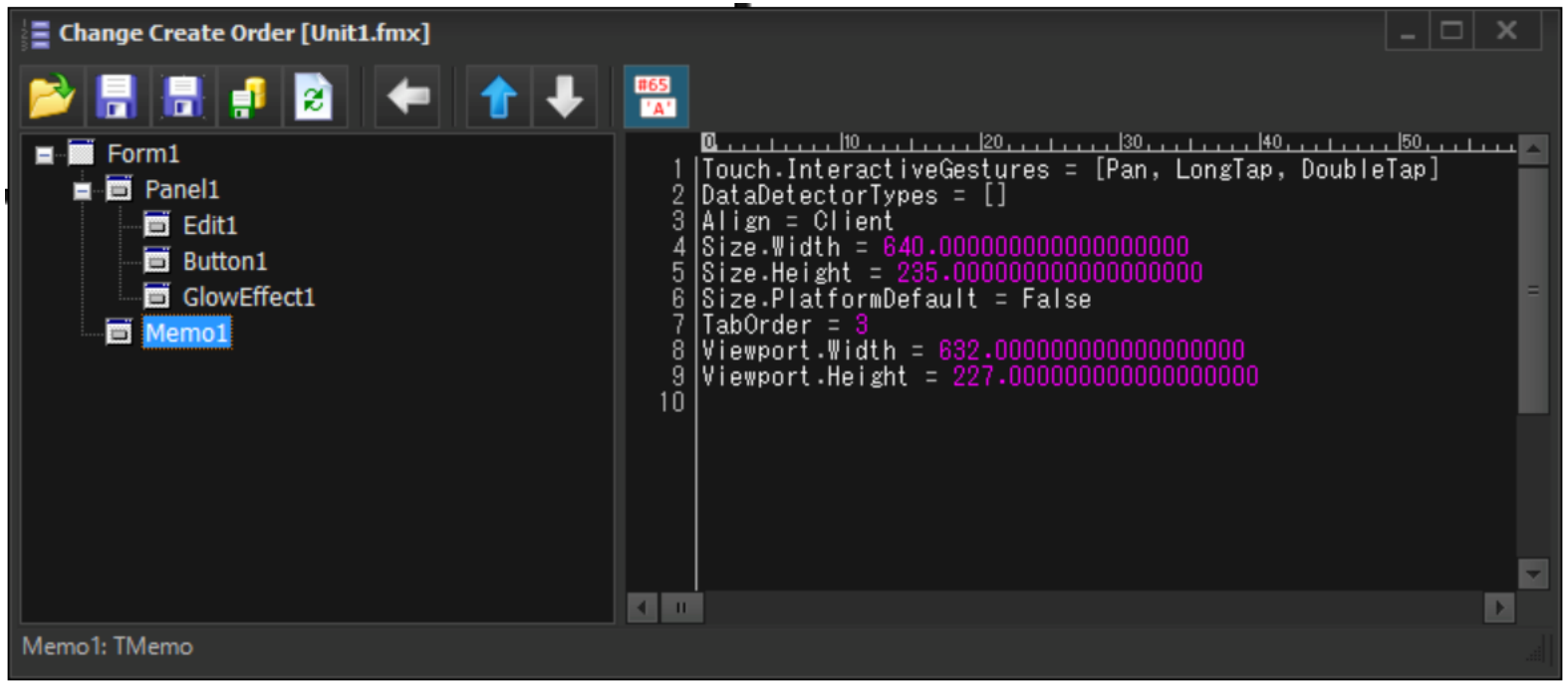


D2007 / D2009～XE4 に対応しています。2005～2006 (or Turbo Delphi), XE5～XE6 は未確認ですが動作すると思います。

添付されているドキュメント(readme.txt) をよく読んでからお使い下さい。エキスパート形式でない DFMコンバータ はこちらにあり

影10 ChangeCreateOrder

ChangeCreateOrder.exe 画面



ドラッグ＆ドロップで任意の位置へ移動する事ができます

IBLite

InterBase XE7 のデータベースファイルがそのまま利用可能です

InterBase XE7をLinuxにインストールしてデータを作成したファイルをiOSで使いたいと思います

InterBase_XE7_JPインストール

InterBase_XE7_JP インストール

InterBase_XE7_JP/Disk1/InstData/Linux/VM この中に

ib64_install.bin ib_install.binがあります

コマンドラインインストールの場合は次のように実行します

```
sudo ./ib64_install.bin -i console
```

これでインストールが開始されまます

インストール時に幾つかの質問で止まるのでyes

InterBase_XE7_JPライセンス登録

/opt/interbase/binにインストールされたプログラムが入ります

```
sudo ./LicenseManagerLauncher -i Console
```

ここで InterBase XE7 Server エディション、
または InterBase XE7 Developer エディションのライセンスを入れるとInterbase XE7が使えます

InterBase_XE7_JP テーブル作成

isqlで テーブルを作成できます

```
./isql
SQL> CREATE DATABASE '/home/***/DBファイル名.ib';
SQL>
SQL> quit; --quitで抜けます
```

DBファイルがある場合

```
./isql
SQL> connect '/home/****/DBファイル名.ib';
```

IBLite をiOSで動かしてみる

先ほどLinuxにInterBase_XE7_JPをインストールしてCreate DBしたファイルを使います

```
./isql
SQL> CREATE DATABASE '/home/***/a1234.ib';
```

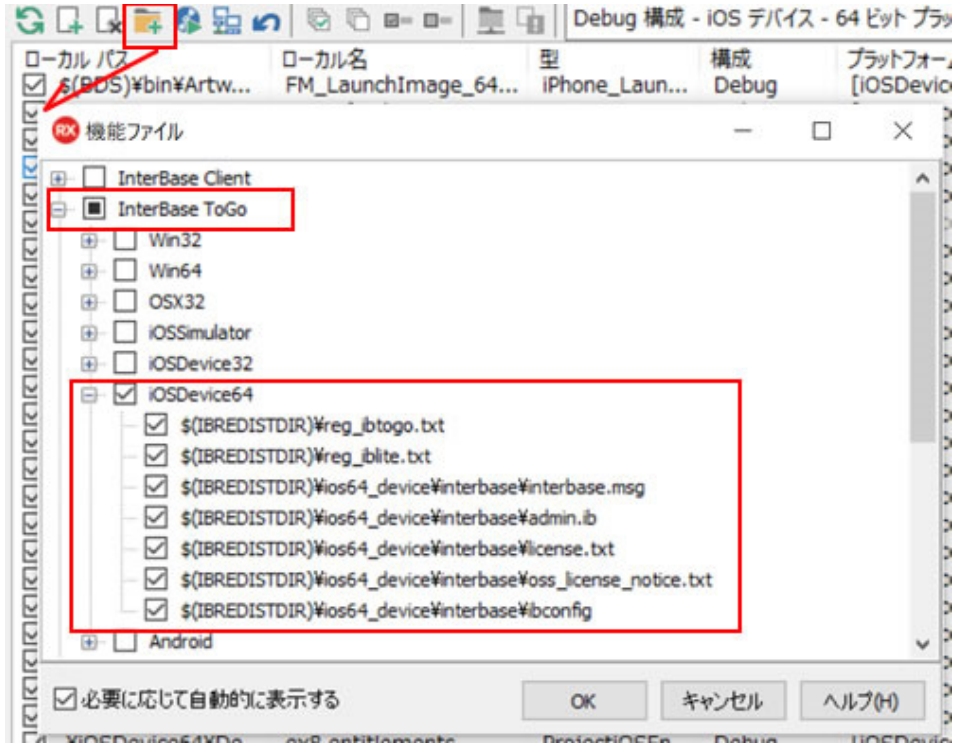
-rw---S--- 1 root root 925696 5月 30 15:03 a1234.ib

↑925KByteほどのファイルでできます

作成したDBファイル(a1234.ib)をRAD Studioインストールしている環境にコピーします

IBLite をiOSで動かしてみる1

iOS プロジェクトを新規作成して 配置の設定

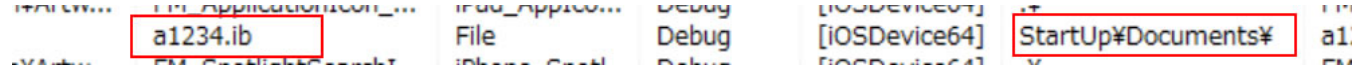


配置一覧から「機能ファイルの追加」

[InterBase ToGo][iOSDevice] を選択

配置一覧から「ファイルの追加」

InterBaseで作成したDBファイルを追加する



「StartUp¥Documents¥」ここ

IBLite をiOSで動かしてみる2

TBL_TEST1と言うテーブルを作成します CREATE TABLE TBL_TEST1(c1 int, c2 varchar(100));

```
void __fastcall TForm1::Button1Click(TObject *Sender)
{
    using TPath_Ioutils = System::Ioutils::TPath;
    TFileName stDBName = TPath_Ioutils::Combine(TPath_Ioutils::TPath::GetDocumentsPath(), "a1234.ib");
    if (FileExists(stDBName) )
    {
        FDConnection1->Params->Database = stDBName;
        FDConnection1->Connected = True;
        FDQuery1->SQL->Text = L"CREATE TABLE TBL_TEST1(c1 int, c2 varchar(100) );";
        FDQuery1->ExecSQL();
        FDConnection1->Connected = False;
    }
}
```

IBLite をiOSで動かしてみる3

TBL_TEST1テーブルインサート

```
void __fastcall TForm1::Button2Click(TObject *Sender)
{
   if (FileExists(stDBName) )
   {
      TDateTime dtTemp = Now();
      FDConnection1->Params->Database = stDBName;
      FDConnection1->Connected = True;
      for (int i = 0; i < 100; i++)
      {
         FDQuery1->SQL->Text = "INSERT INTO TBL_TEST1 values (" +IntToStr(i)+ ", '" + Edit1->Text +"') ";
         FDQuery1->ExecSQL();
      }
      FDConnection1->Connected = False;
      dtTemp  = Now() - dtTemp;
      Memo1->Lines->Append(L"Time " + FormatDateTime("hh:nn:ss", dtTemp));
   }
}
```

IBLite をiOSで動かしてみる4

TBL_TEST1テーブル参照

```
void __fastcall TForm1::Button3Click(TObject *Sender)
{
   if (FileExists(stDBName) )
   {
      TDateTime dtTemp = Now();
      FDConnection1->Params->Database = stDBName;
      FDConnection1->Connected = True;
      FDQuery1->SQL->Text = "select * from TBL_TEST1";
      FDQuery1->Active = True;
      while (!(FDQuery1->Eof))
      {
         Memo1->Lines->Append(
            FDQuery1->FieldByName("c2")->AsWideString);
         FDQuery1->Next();
      }
      FDQuery1->Active = False;
      FDConnection1->Connected = False;
      dtTemp  = Now() - dtTemp;
      Memo1->Lines->Append(L"Time " + FormatDateTime("hh:nn:ss", dtTemp));
   }
}
```

まとめ

- 画面切替
  - 画面切替時なぜ動きが必要かは利用者に切り替った事をお知らせする大事な役割
  - 様々な方法で動きのある画面遷移が作れる

- 影付け
  - 雰囲気を出すのと利用者側から見て境界線メリハリが認識できる
  - *.fmxファイルの順番は重要

THANKS!

www.embarcadero.com/jp

第32回 エンバカデロ・デベロッパーキャンプ