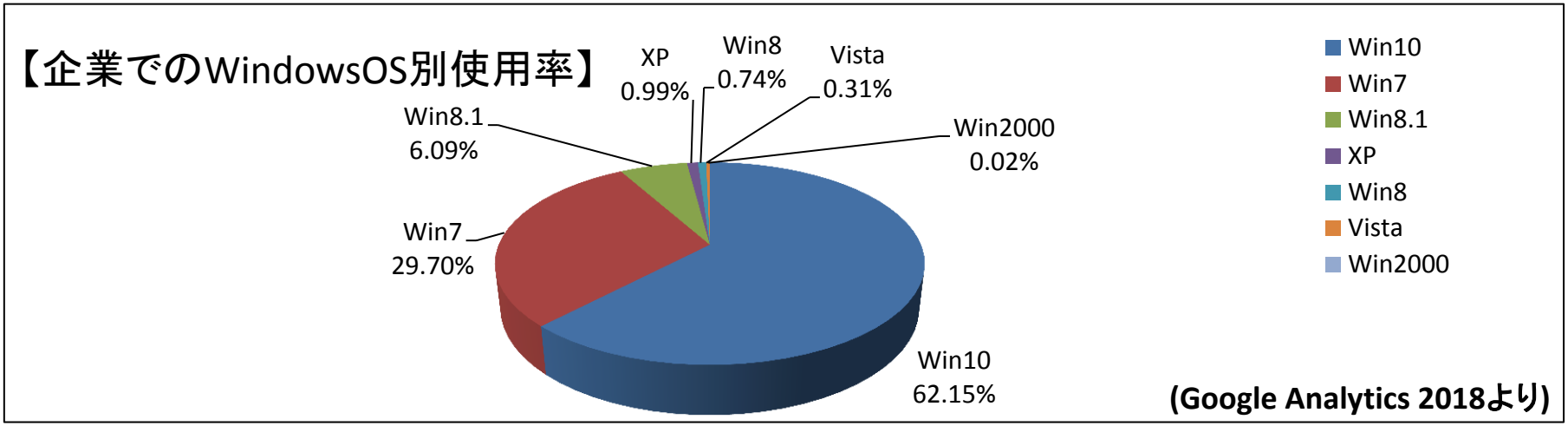
【セッションNo．2】

開発者が知りたい
実践プログラミングテクニック！

株式会社ミガロ.
RAD事業部　技術支援課
鶴澤　佳樹

■ はじめに

企業でWindows10が使われる割合は、2018年で既に6割を超えており、
最も多く使われているWindows OSとなりました。



Windows10がこれまでのWindows OSと大きく異なる特徴はデスクトップPCだけでなく、Windowsタブレットも使用対象としている点です。その為、Windows10向けのアプリケーション開発では、これらの環境を考慮したレスポンシブデザインやタッチ操作対応のユーザーインターフェースが重要になってきています。

本セッションでは、**Delphi/400のコンポーネントで、このようなWindows10のユーザーインターフェースに対応することをテーマ**とした実践プログラミングテクニックをご紹介いたします。

【アジェンダ】

1. レスポンシブデザインに対応するコンポーネント
   - TStackPanel、TGridPanel、TFlowPanel
2. 画面領域を有効に活用するコンポーネント
   - TSplitView、TCardPanel
3. タッチ操作に対応するコンポーネント
   - TToggleSwitch、TCalendarPicker、TDatePicker 、TTimePicker
4. ジェスチャに対応するコンポーネント
   - TGestureManager
5. まとめ

1.レスポンシブデザインに対応するコンポーネント

■ レスポンシブデザインとは？

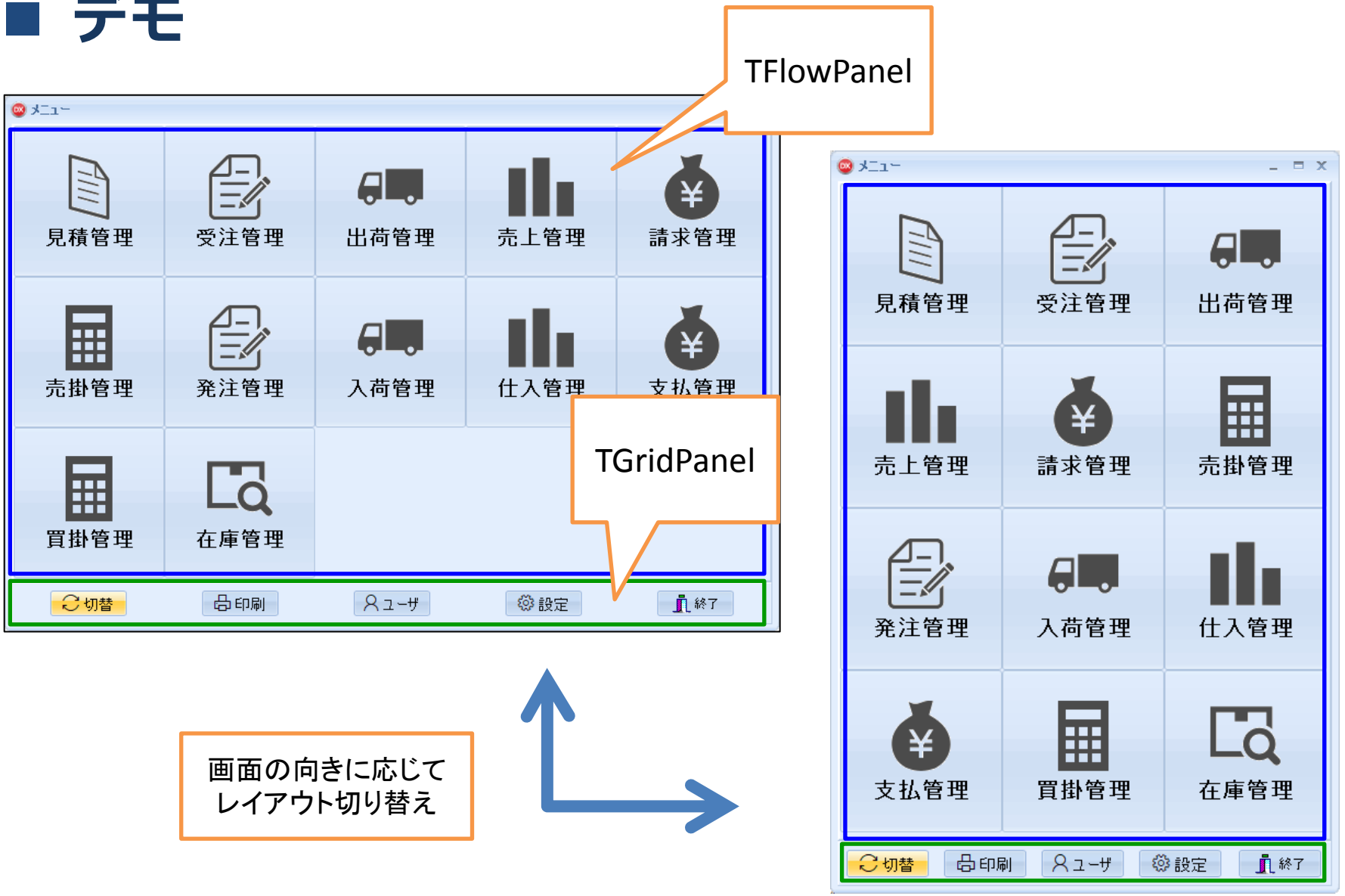
- レスポンシブデザイン
  Webページでよく用いられる手法で、どんな画面サイズであっても適切に表示されるように、文字サイズやレイアウトを変更するデザイン技術。

例）ミガロ.ホームページ

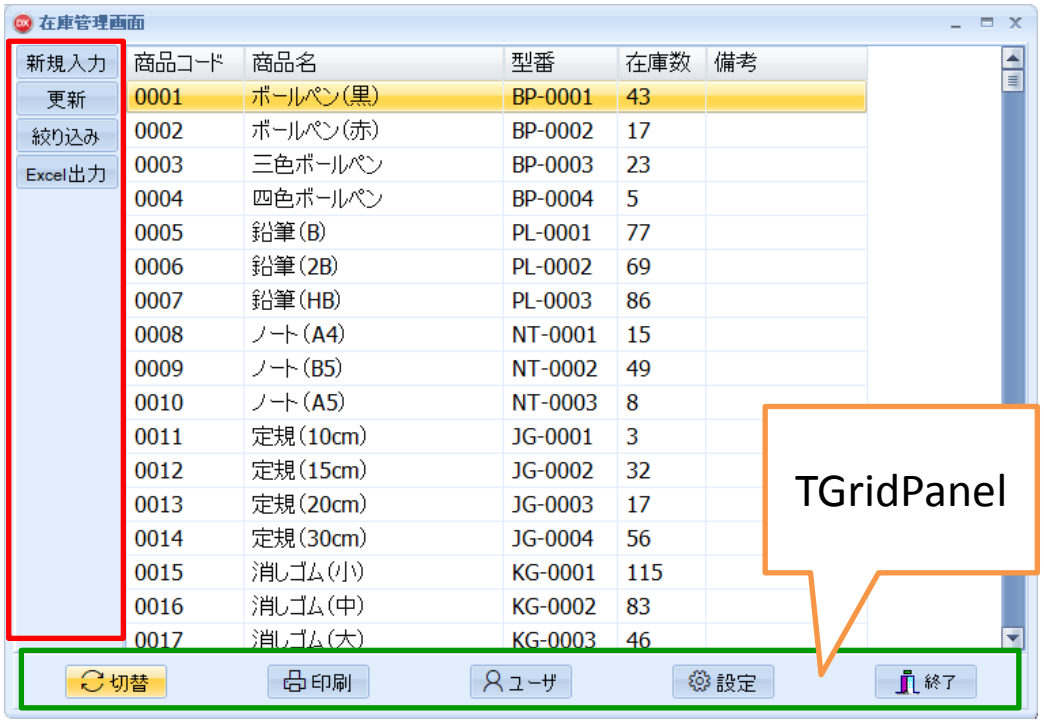
デスクトップPCでの表示

モバイル端末での表示

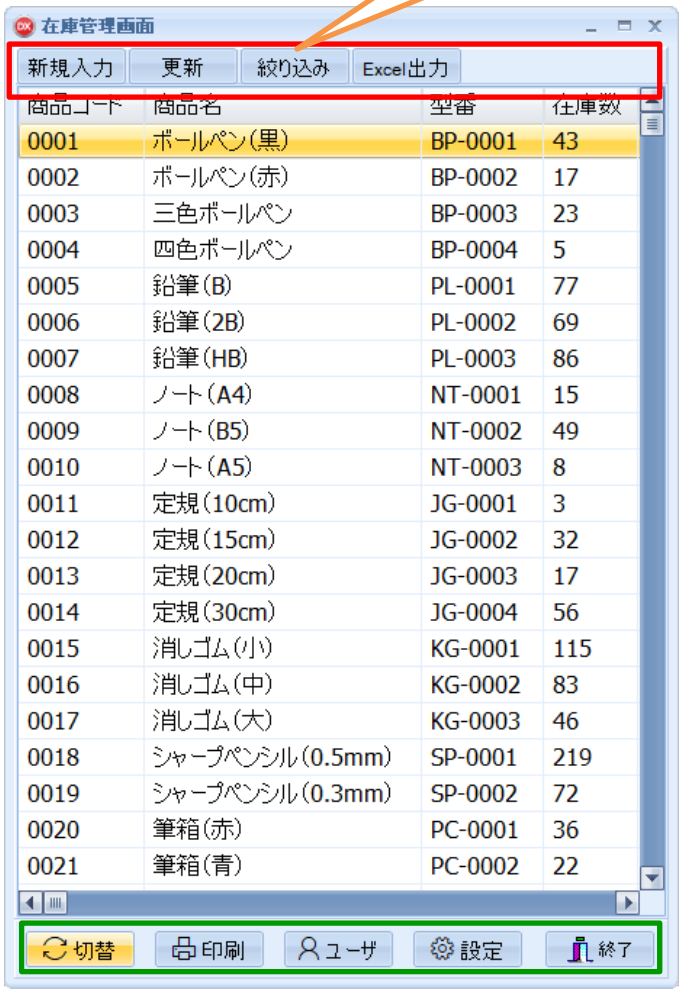
■ デモ


TFlowPanel
TGridPanel
メニュー
見積管理
受注管理
出荷管理
売上管理
請求管理
売掛管理
発注管理
入荷管理
仕入管理
支払管理
買掛管理
在庫管理
切替
印刷
ユーザ
設定
終了
画面の向きに応じて
レイアウト切り替え

■ デモ



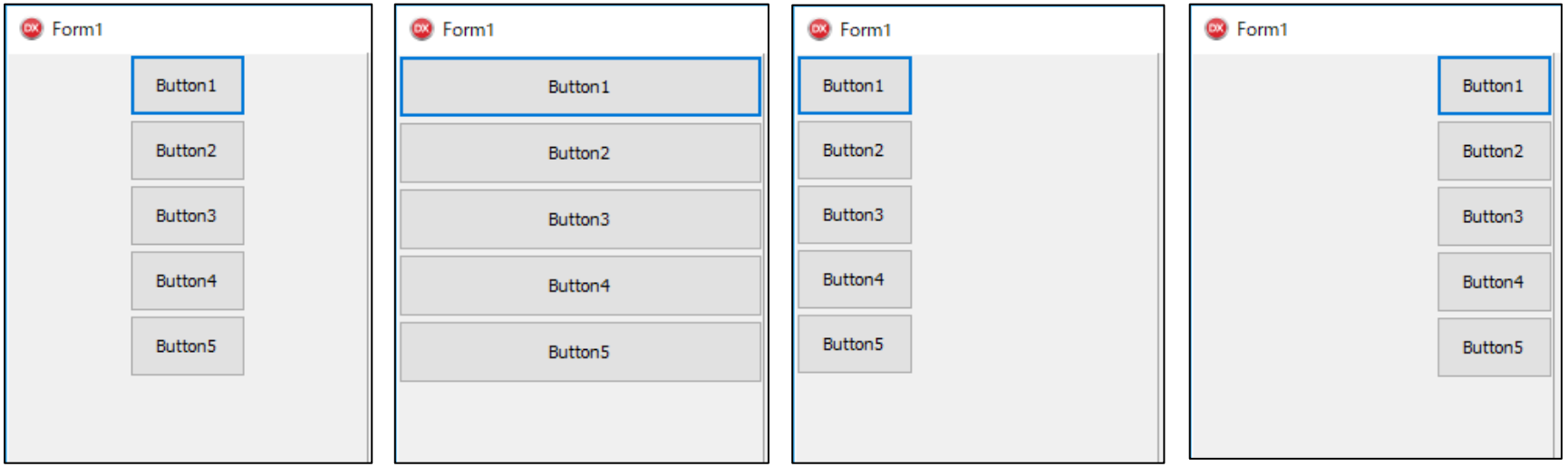
画面の向きに応じて
レイアウト切り替え



TStackPanelコンポーネント(Delphi/400 10.2 Tokyo～)

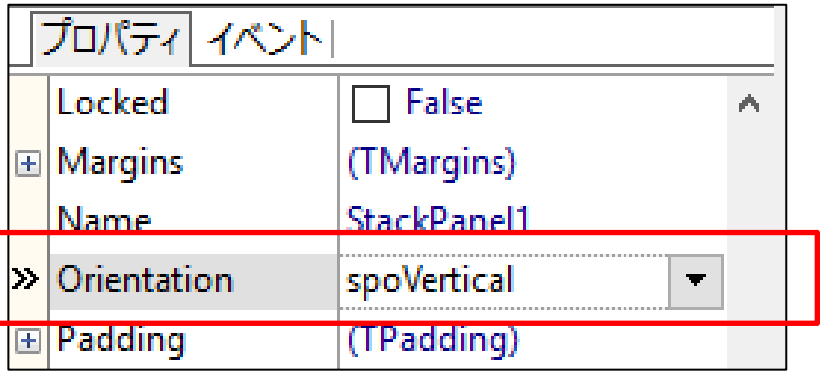
- 機能概要
  コンテナ内のすべてのコンポーネントを、それぞれ独自の高さや幅の設定を維持しつつ、垂直または水平に自動的に位置揃えして配置することができるコンポーネント。

  その際、マージンやパディングの設定もすべてのコンポーネントに適用させることができる。



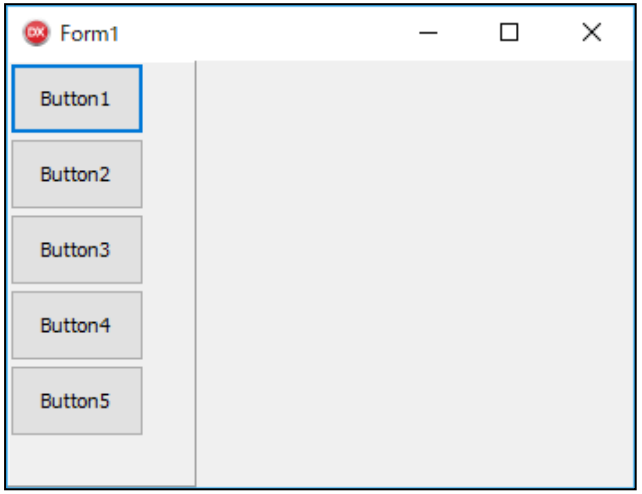
TStackPanelコンポーネントの使い方

- 配置方向に関するプロパティ

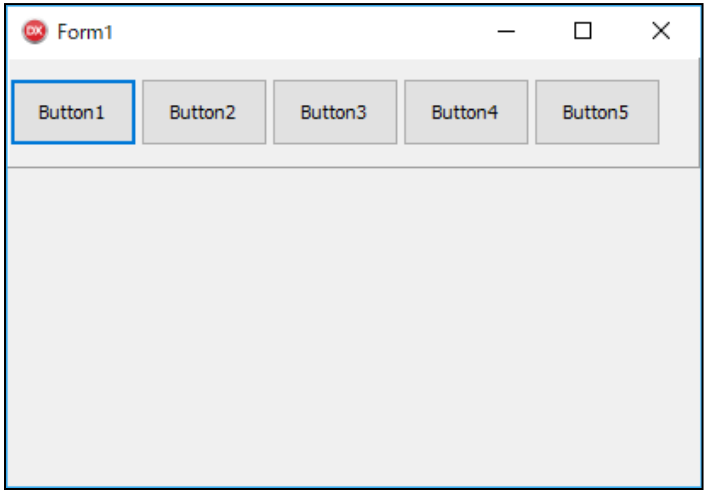


Orientationプロパティ ：
　コンポーネントを並べる方向
spoVertical ： 垂直方向
spoHorizontal ： 水平方向

spoVertical

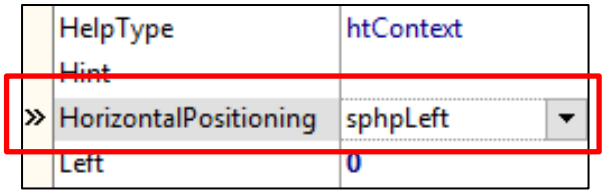


spoHorizontal

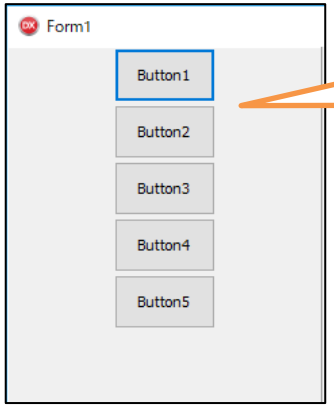


TStackPanelコンポーネントの使い方

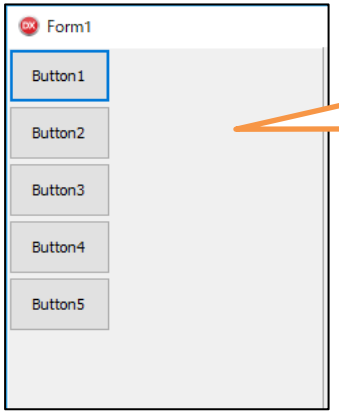
- 位置揃えに関するプロパティ



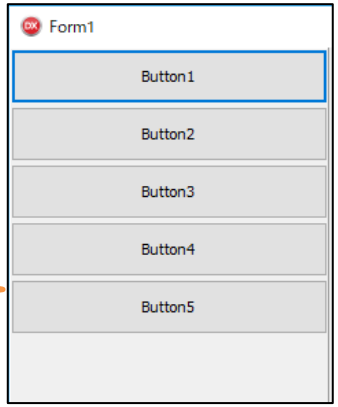
HorizontalPositioningプロパティ　：
コンポーネントを垂直方向に並べる際の、水平方向の位置揃え



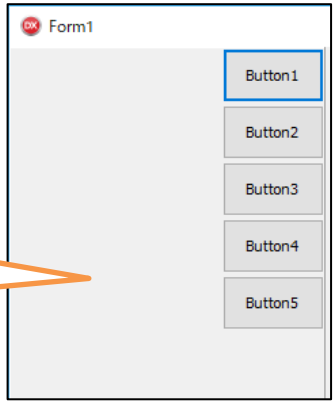
sphpCenter　：
中央揃えで配置



sphpLeft　：
左揃えで配置



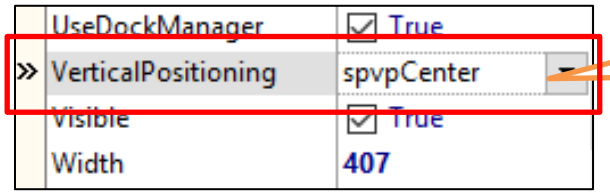
sphpFill　：
左右の端に揃えて配置



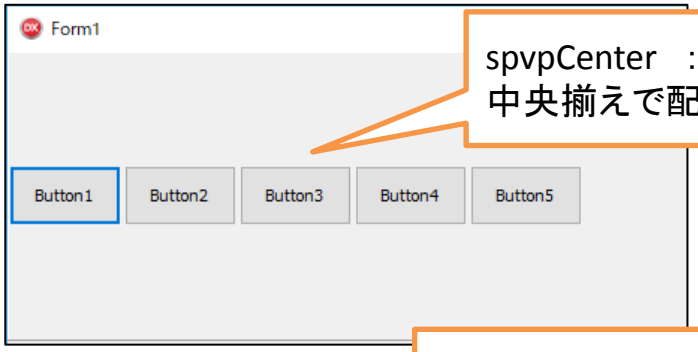
sphpRight　：
右揃えで配置

TStackPanelコンポーネントの使い方

- 位置揃えに関するプロパティ

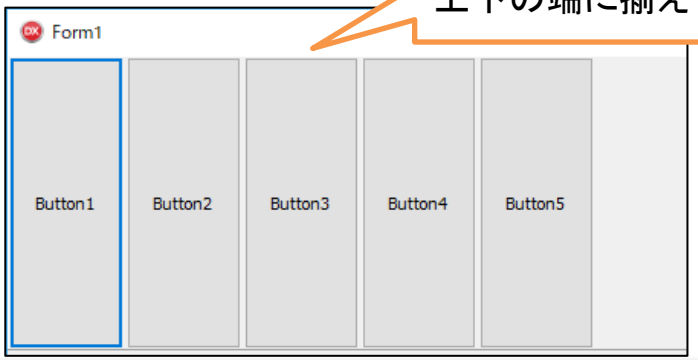


VerticalPositioningプロパティ　：
コンポーネントを水平方向に並べる際の、垂直方向の位置揃え



spvpCenter　：
中央揃えで配置

spvpBottom　：
下揃えで配置



spvpFill　：
上下の端に揃えて配置

spvpTop　：
上揃えで配置

TStackPanelコンポーネントの使い方

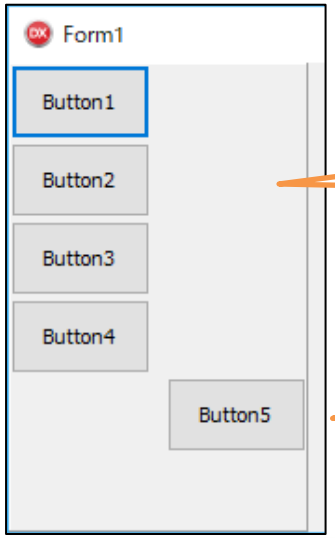
- 配置コンポーネントへの追加プロパティ
  TStackPanel上に配置したコンポーネントに対しては、元々のプロパティの下に、TStackPanel用のプロパティが追加される。

追加されるプロパティ

ControlIndex	0
HorizontalPositioning	sphpDefault
VerticalPositioning	spvpDefault

ControlIndex　：　TStackPanel内での、コンポーネントの並び順

HorizontalPositioning、VerticalPositioning　：
TStackPanel自体にも設定できるプロパティ
配置したそれぞれのコンポーネントに対して、個別に設定も可能
デフォルト値はsphpDefault、spvpDefault
（TStackPanelでの設定を引き継ぐ）



TStackPanel自体にはsphpLeftを設定

このコンポーネントのみsphpRightを設定

TStackPanelコンポーネントの使い方

- 画面作成時の処理

コーディング例

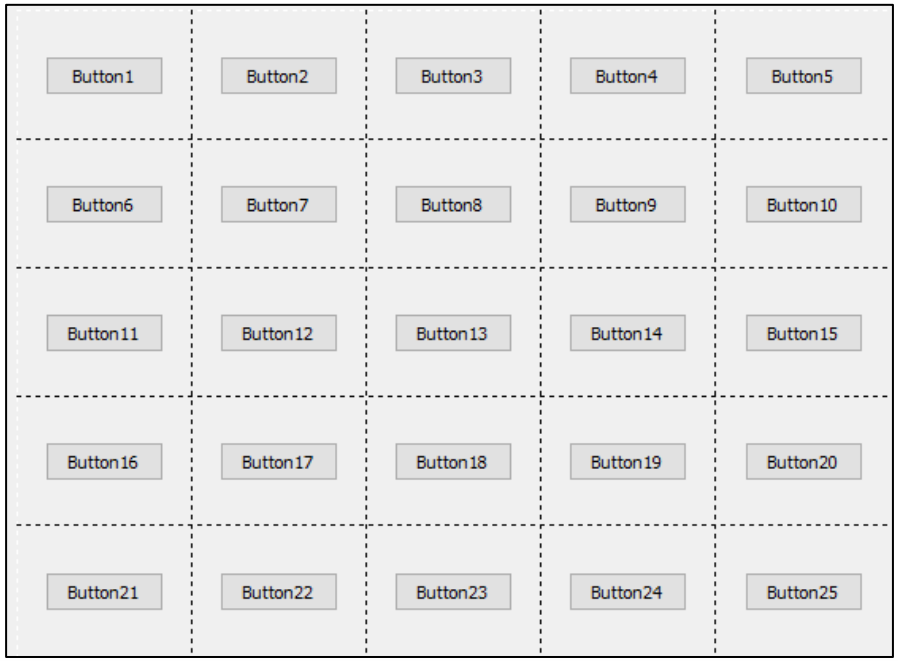
```
procedure TForm1.FormCreate(Sender: TObject);
begin

  //画面の向きに応じてTStackPanelの方向を切り替える
  if Height < Width then
  begin
    StackPanel1.Align := alLeft;
    StackPanel1.Orientation := spoVertical;
    StackPanel1.Width := 75;
  end
  else
  begin
    StackPanel1.Align := alTop;
    StackPanel1.Orientation := spoHorizontal;
    StackPanel1.Height := 25;
  end;

end;
```

デモプログラムでは、同様の処理をOnResizeイベントにも記載

TGridPanelコンポーネント(Delphi/400 V2006～)

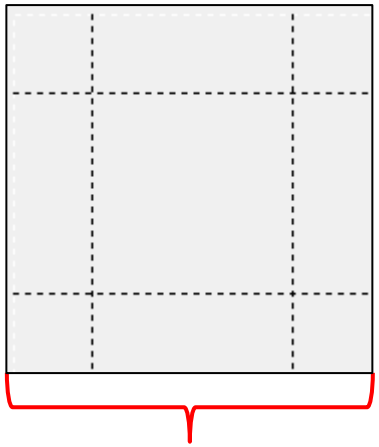
- 機能概要
  パネルを指定した列数および行数でグリッド状に分割し
  各コンポーネントをセル内に配置できるコンポーネント。

  各セルにはそれぞれ1つずつコンポーネントを配置可能となっており、
  各コンポーネントは自動的に整列される。



TGridPanelコンポーネントの使い方

- 列の設定に関するプロパティ



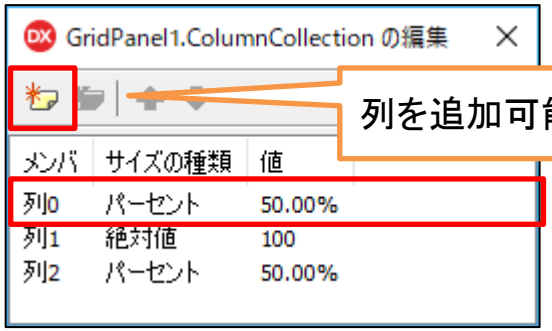
ColumnCollectionプロパティ　：
分割数や各列のサイズを設定

Color	clBtnFace
ColumnCollection	(ColumnCollection)
Constraints	(TSizeConstraints)

SizeStyleプロパティ　：　サイズの種類
ssAbsolute：絶対値、ssAuto：自動、ssPercent：比率

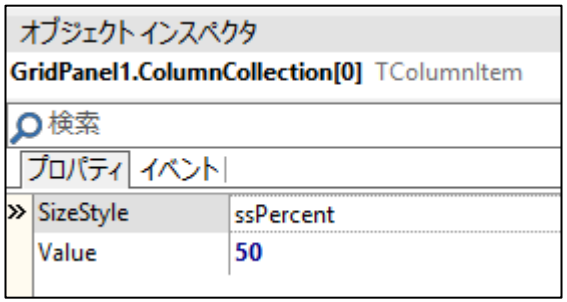
Valueプロパティ　：
SizeStyleがssAbsolute（絶対値）の時は実際のサイズ（列の幅）
SizeStyleがssPercent（比率）の時はTGridPanelに対する割合

列のプロパティ



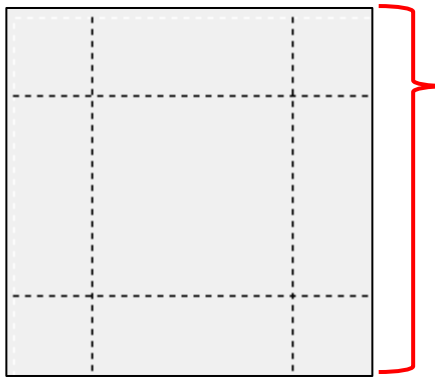
列を追加可能

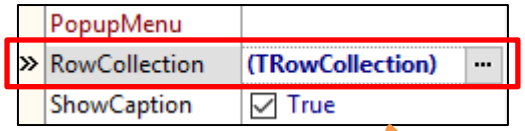
列を選択



TGridPanelコンポーネントの使い方

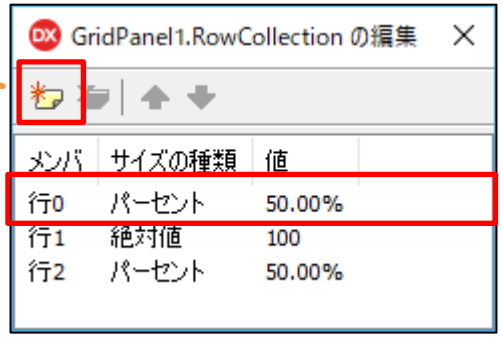
- 行の設定に関するプロパティ





RowCollectionプロパティ　：
分割数や各行のサイズを設定

行を追加可能

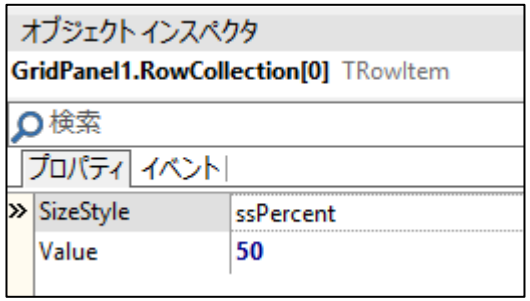


行を選択

行のプロパティ

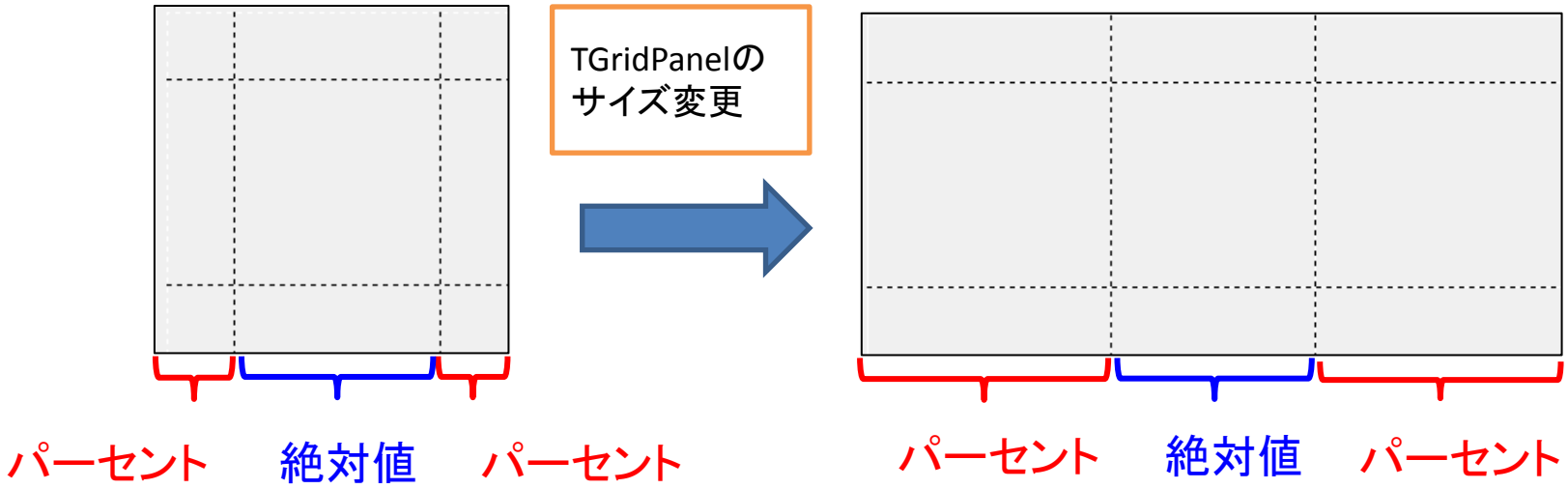
SizeStyleプロパティ　：　サイズの種類
ssAbsolute：絶対値、ssAuto：自動、ssPercent：比率

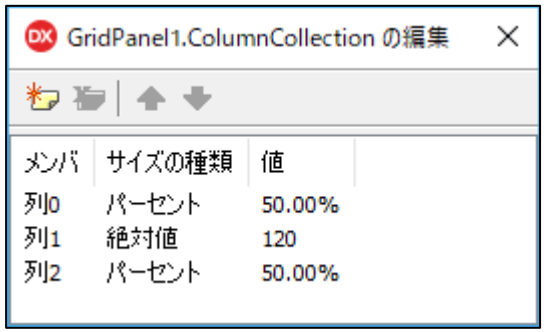
Valueプロパティ　：
SizeStyleがssAbsolute（絶対値）の時は実際のサイズ（行の高さ）
SizeStyleがssPercent（比率）の時はTGridPanelに対する割合



TGridPanelコンポーネントの使い方

- サイズの指定方法による違い





SizeStyleプロパティがssPercent（比率）の場合：
セルのサイズはTGridPanelのうち絶対値を除いたサイズと
Valueプロパティの設定値（パーセント）より算出

SizeStyleプロパティがssAbsolute（絶対値）の場合：
セルのサイズはValueプロパティの設定値で固定

TGridPanelコンポーネントの使い方

- 配置コンポーネントへの追加プロパティ
  TGridPanel上に配置したコンポーネントに対しては、元々存在しているプロパティの下に、TGridPanel用のプロパティが追加される。

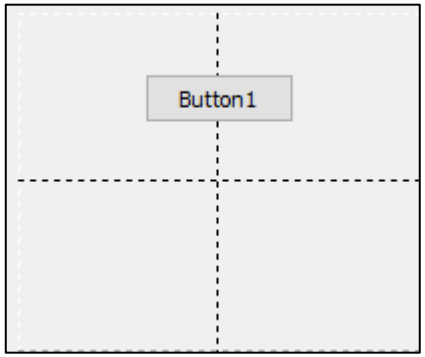
追加されるプロパティ

Column	0
Row	0
ColumnSpan	1
RowSpan	1

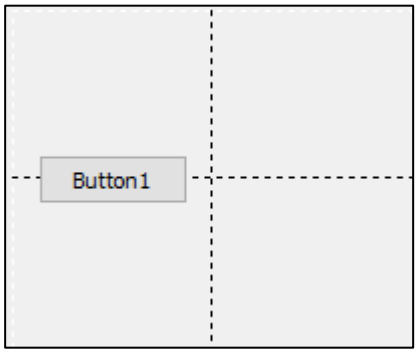
Column、Rowプロパティ　：
　コンポーネントが配置されている列、行の位置

ColumnSpan、RowSpanプロパティ　：
　配置されているコンポーネントがいくつの列、行にまたがるか
　2以上の場合、複数のセルにまたがってコンポーネントを配置可能

ColumnSpanに
2を設定



RowSpanに
2を設定



TFlowPanelコンポーネント(Delphi/400 V2006～)

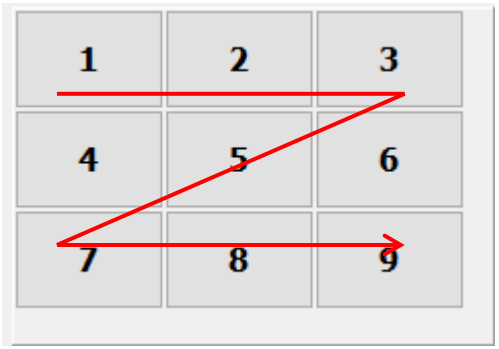
- 機能概要
  コンテナ内のコンポーネントを、あらかじめ決められた並びに従って、自動的に位置を合わせて配置できるコンポーネント。
  配置されたコンポーネントは、TFlowPanelの大きさに従って自動的に再配置される。

TFlowPanelコンポーネントの使い方

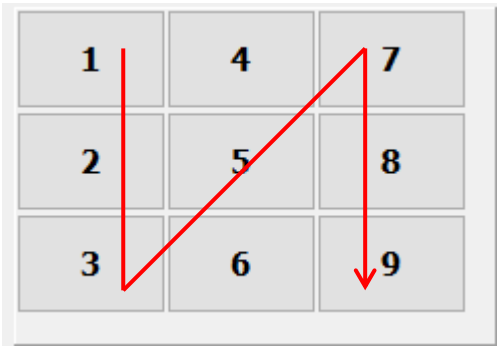
- コンポーネントの配置方向に関するプロパティ

FlowStyleプロパティ　：
コンポーネントを配置する方向
配置する方向によって、8種類の値を設定可能

FlowStyleに
fsLeftRightTopBottomを設定



FlowStyleに
fsTopBottomLeftRightを設定



2.画面領域を有効に活用するコンポーネント

■ 画面領域の有効活用について

- 画面領域の有効活用
  情報量の多いアプリケーションを画面の小さいWindowsタブレット等から表示すると、コンテンツが画面内に収まらない場合がある。
  様々な端末からの利用に対応するには、画面領域を有効に活用し、どのような画面サイズの端末からでも効率良く情報を参照できるアプリケーションを考慮する必要がある。

デスクトップPCでの表示例

Windowsタブレットでの表示例

デモ

ボタンをクリックすると
検索用のメニューが開く

TSplitView

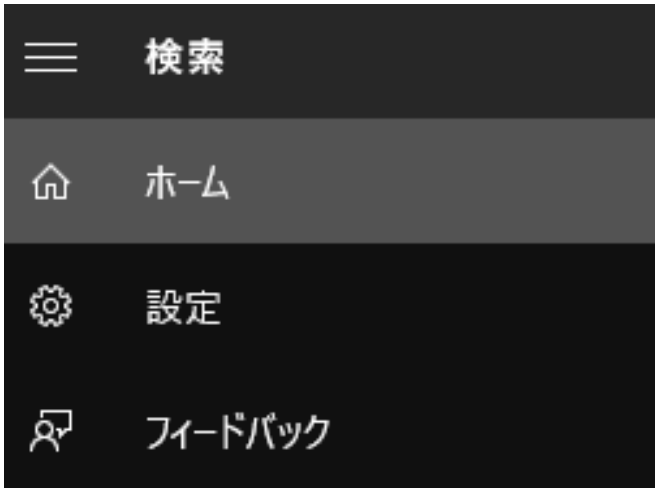
ジェスチャで物件の画像を
切り替えることができる

TCardPanel

TSplitViewコンポーネント（Delphi/400 10Seattle～）

- 機能概要
  Windows10のスタートメニューのように、画面の左右の端から
  開いたり閉じたりすることのできるコンポーネント。
  オープン時、クローズ時の挙動をプロパティから設定することも可能。

Windows10のスタートメニュー（例）



TSplitView

TSplitViewコンポーネントの使い方

- オープン時の動きに関するプロパティ

DisplayModeプロパティ ： オープン時の挙動を設定
svmDocked ： TSplitViewが開いた分だけ、クライアント領域が小さくなる
svmOverlay ： クライアント領域に覆い被さるようにして、最前面にTSplitViewを表示

TSplitViewコンポーネントの使い方

- クローズ時の動きに関するプロパティ

CloseStyleプロパティ　：　クローズ時の挙動を設定
　svcCollapse　：　クローズ時、TSplitViewが完全に非表示
　svcCompact　：　クローズ時、TSplitViewが一部残る

オープン時

クローズ時
（CloseStyle：svcCollapse）

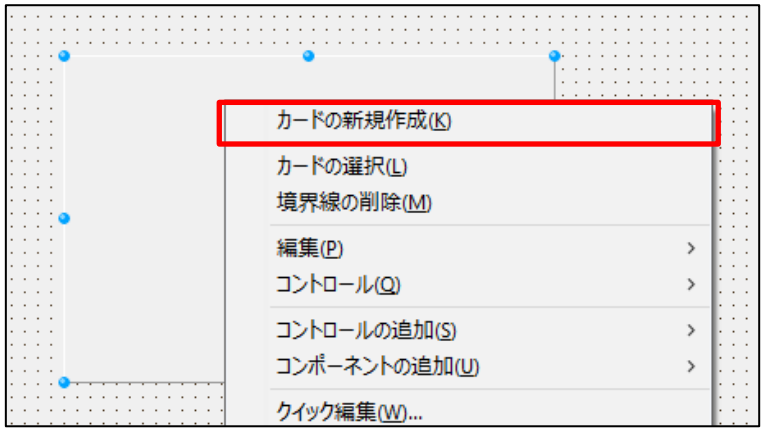
クローズ時
（CloseStyle：svcCompact）

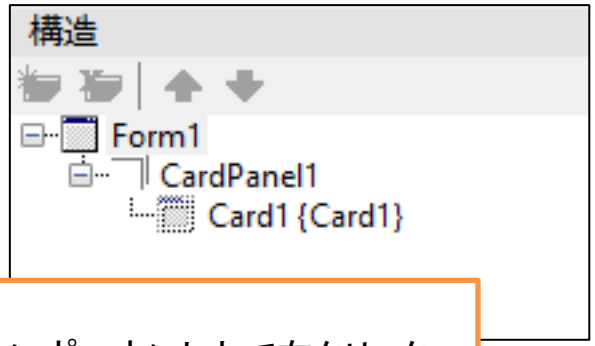
TCardPanelコンポーネント（Delphi/400 10.2 Tokyo～）

- 機能概要
  カードパネルのコレクションを表示することができるコンポーネント。
  コレクション内のカードはループしてつながっており、ジェスチャとの親和性も高い。

TCardPanelコンポーネントの使い方

- カードの追加方法





TCardPanelのコンポーネント上で右クリック
⇒「カードの新規作成」でカードを追加可能

それぞれのカードは異なるコンテナコンポーネントとして
使用でき、別々にデザイン可能

TCardPanelコンポーネントの使い方

- カードの切り替え方法

①NextCardメソッド

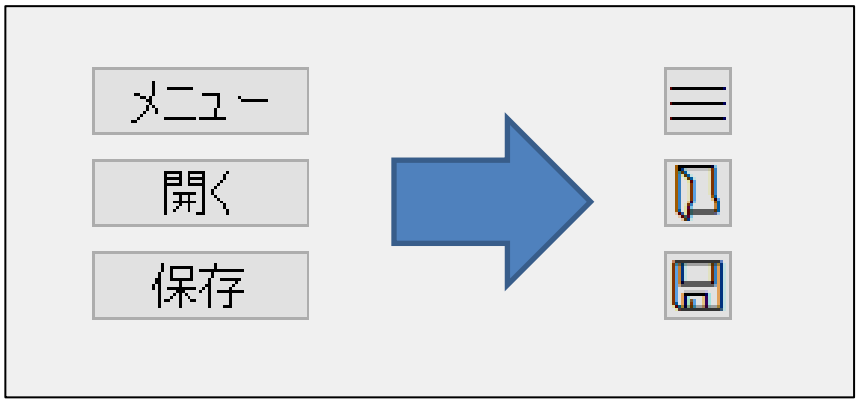
次のカードを参照するためのメソッド。
最後のカードを参照している時に実行すると、最初のカードに戻る。

②PreviousCardメソッド

前のカードを参照するためのメソッド。
最初のカードを参照している時に実行すると、最後のカードに戻る。

■ 補足：Segoe MDL2 Assets フォントとは？

- Segoe MDL2 Assets フォント
  Windows10からは「Segoe MDL2 Assets」というフォントが追加になっており、使用することで様々な記号を表現することが可能になっている。このフォントを使えば、例えばボタンのアイコンイメージを画像ではなく文字で表示できる利点があり、狭いスペースで効率よく情報を参照することができる。

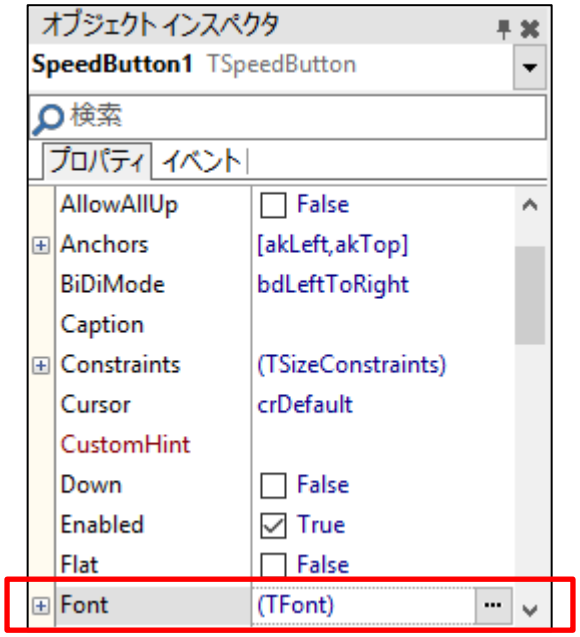


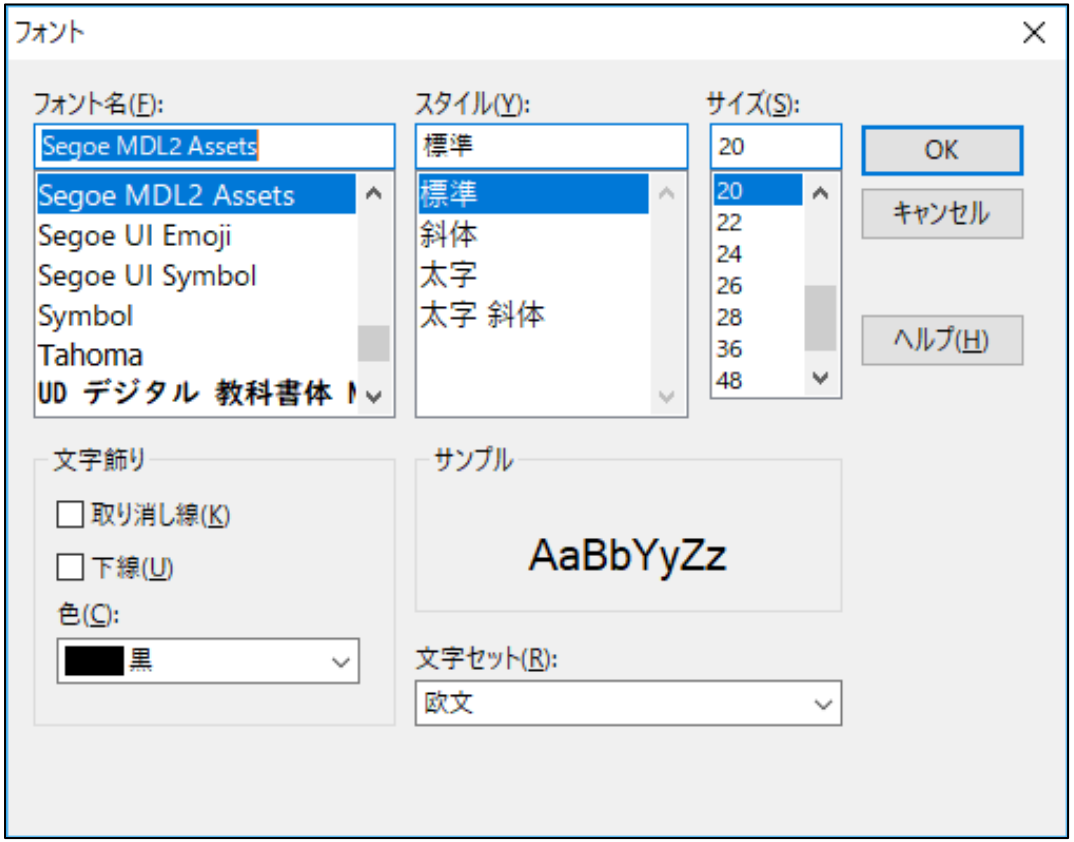
■ 補足：Segoe MDL2 Assets フォントの使い方

- 使用手順①

例）TSpeedButtonのCaptionに使用する場合

フォントの設定で
「Segoe MDL2 Assets」を選択





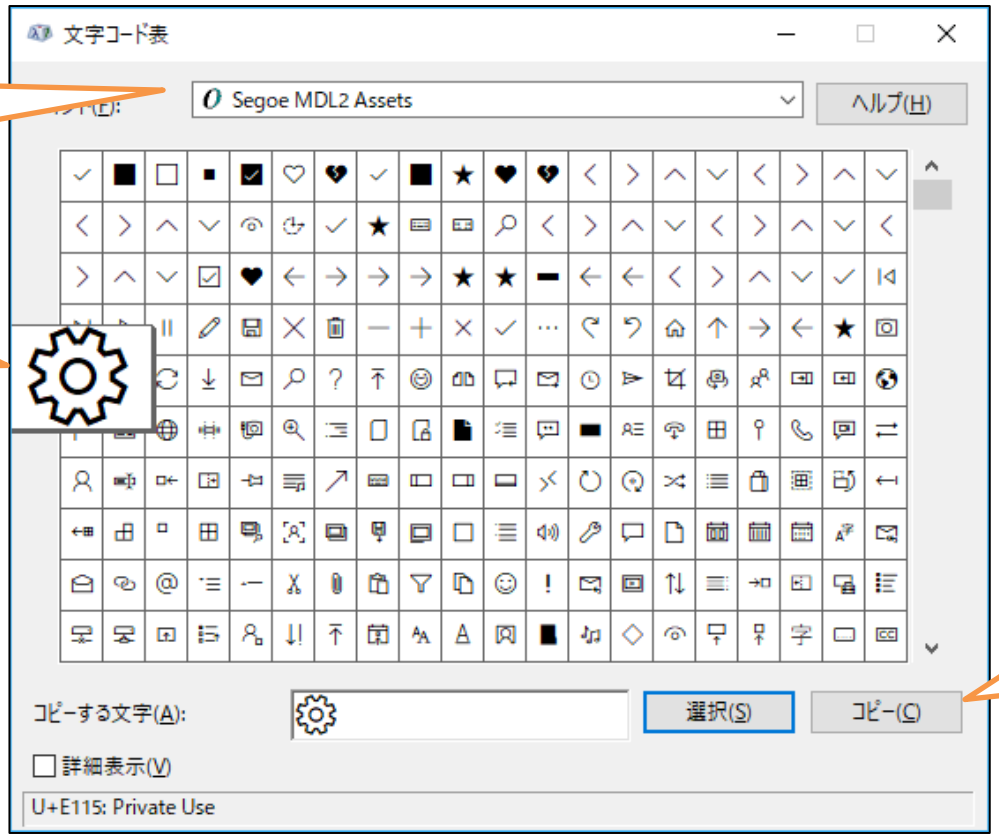
■ 補足：Segoe MDL2 Assets フォントの使い方

- 使用手順②

Windowsの文字コード表を起動する。

①「Segoe MDL2 Assets」を選択

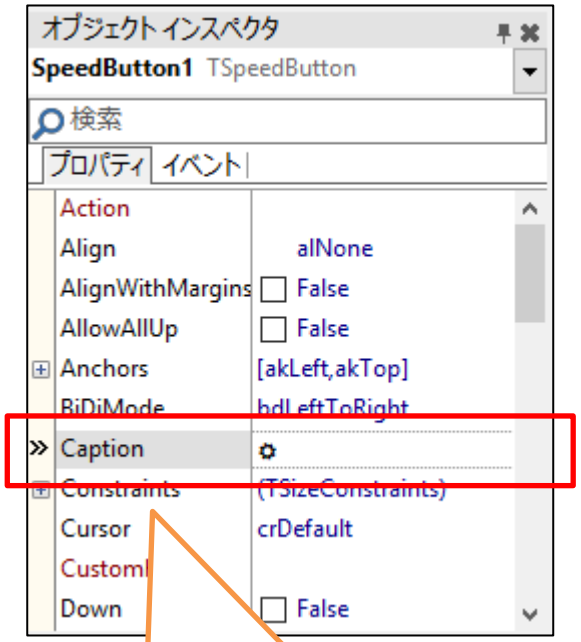
②記号を選択

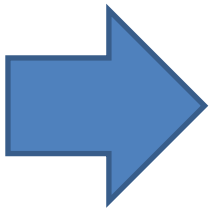


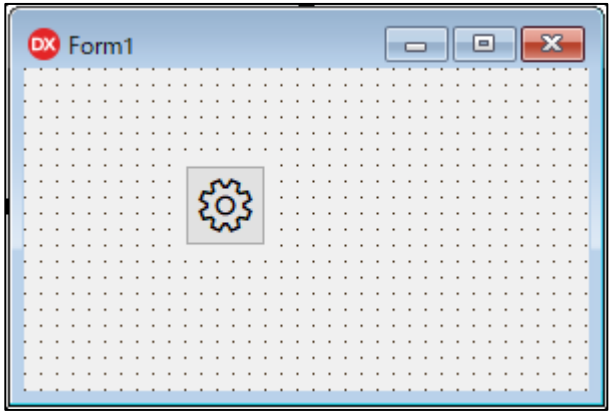
③コピー

■ 補足：Segoe MDL2 Assets フォントの使い方

- 使用手順③



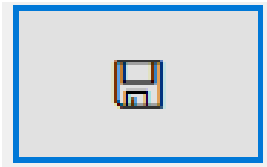




記号のみのボタンを簡単に作成することができ
狭いスペースで情報を表現することが可能

コンポーネントのCaptionプロパティに
コピーした文字をそのまま貼り付ける
（記号によっては･と表示されるものもある）

Windows10



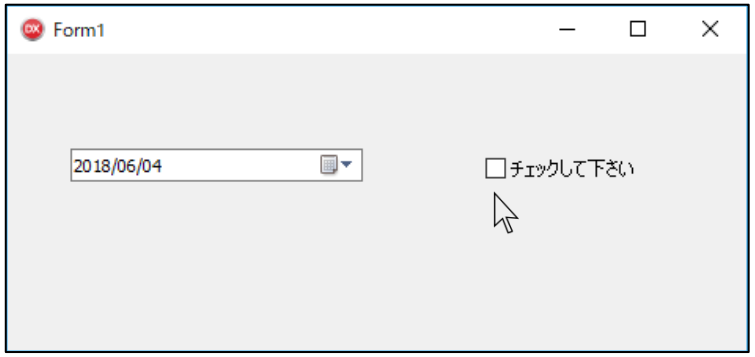
Windows7

※注意点※
同じ文字でも「Segoe MDL2 Assets」フォントがない環境
（例えばWindows7）では正しく表示できない。

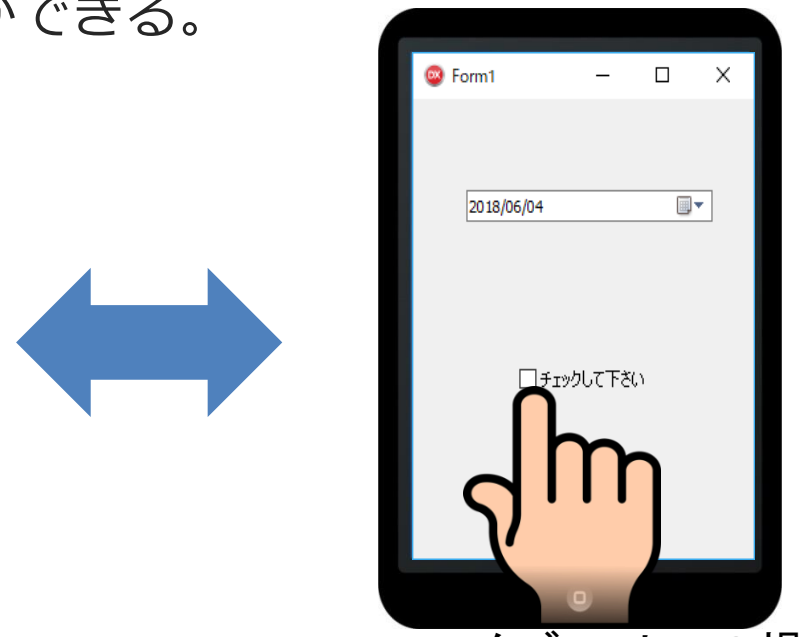
3.タッチ操作に対応するコンポーネント

■ タッチ操作への対応について

- タッチ操作への対応
  Windowsタブレット等、タッチ操作が行える端末でアプリケーションを利用する場合、従来のマウスでのクリック操作に対応したアプリでは操作できる領域が狭く、タッチ操作が行いづらい場合がある。操作領域の広いコンポーネントを活用することで、タッチの操作性が良いアプリケーションを作成することができる。



デスクトップPCでの操作
（マウス操作）

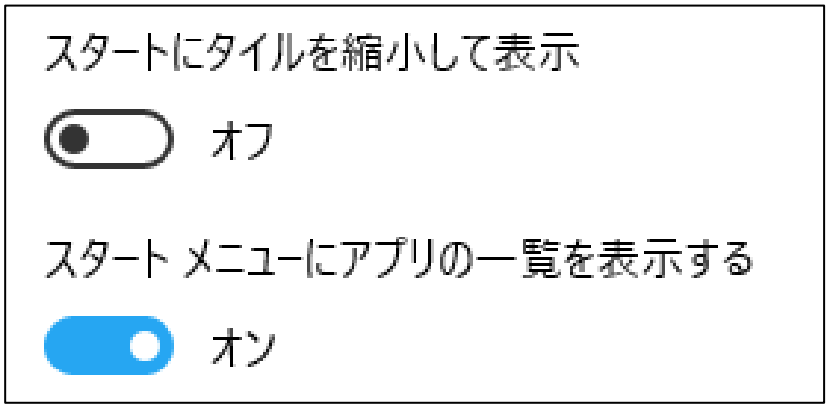


Windowsタブレットでの操作
（タッチ操作）

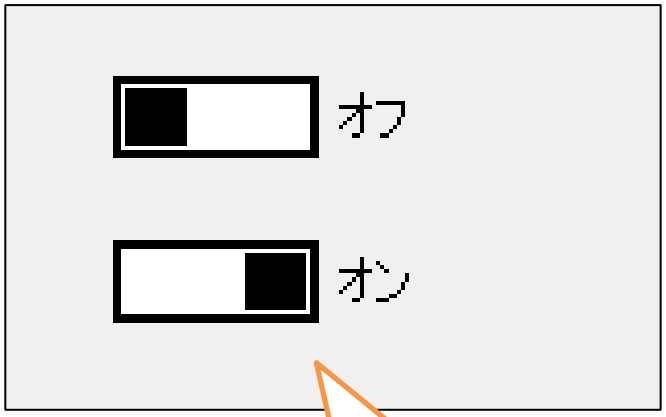
TToggleSwitchコンポーネント(Delphi/400 10 Seattle～)

- 機能概要
  Windows10の設定画面で使われている、スイッチ型のコンポーネント。
  タッチ操作でオン/オフを切り替えやすく設計されている。

Windows10の設定画面



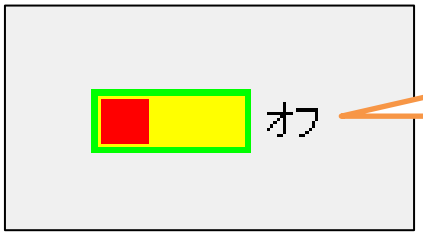
TToggleSwitchコンポーネント



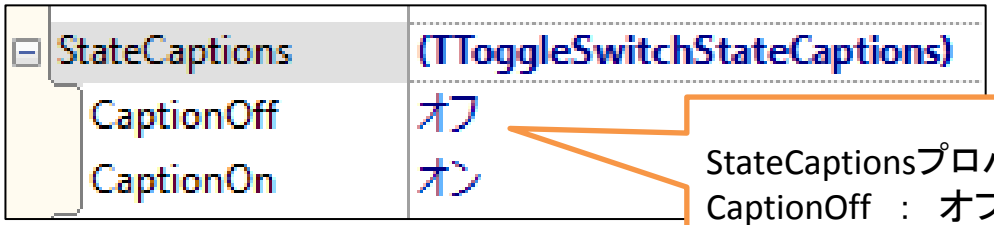
Stateプロパティ ： オン/オフの状態
tssOff ： オフ
tssOn ： オン

TToggleSwitchコンポーネントの使い方

- 表示に関するプロパティ
  各部分の色の他、オン/オフの状態で表示される文字列を設定することが可能。



Colorプロパティ　：　コンポーネントの背景色
FrameColorプロパティ　：　コンポーネントの枠の色
ThumbColorプロパティ　：　スイッチのつまみ部分の色

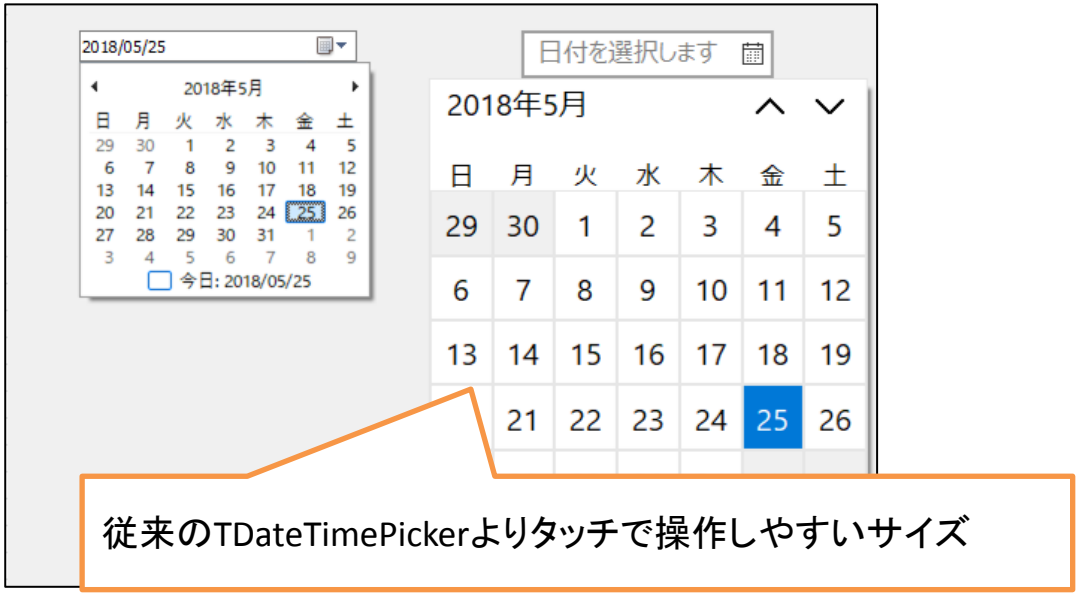


StateCaptionsプロパティ　：　オン/オフの状態で表示される文字列
CaptionOff　：　オフの状態で表示される文字列
CaptionOn　：　オンの状態で表示される文字列

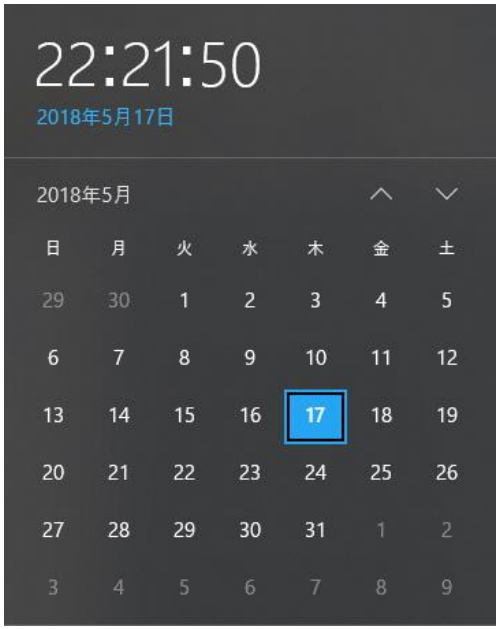
TCalendarPickerコンポーネント(Delphi/400 10Seattle～)

- 機能概要
  Windows10の通知領域のカレンダーと同じ操作性で日付を選択することができるコンポーネント。
  カレンダー部分は従来のTDateTimePickerより大きくなっており、タッチでの操作を行いやすくなっている。

TDateTimePicker（マウス操作）　TCalendarPicker（タッチ操作）

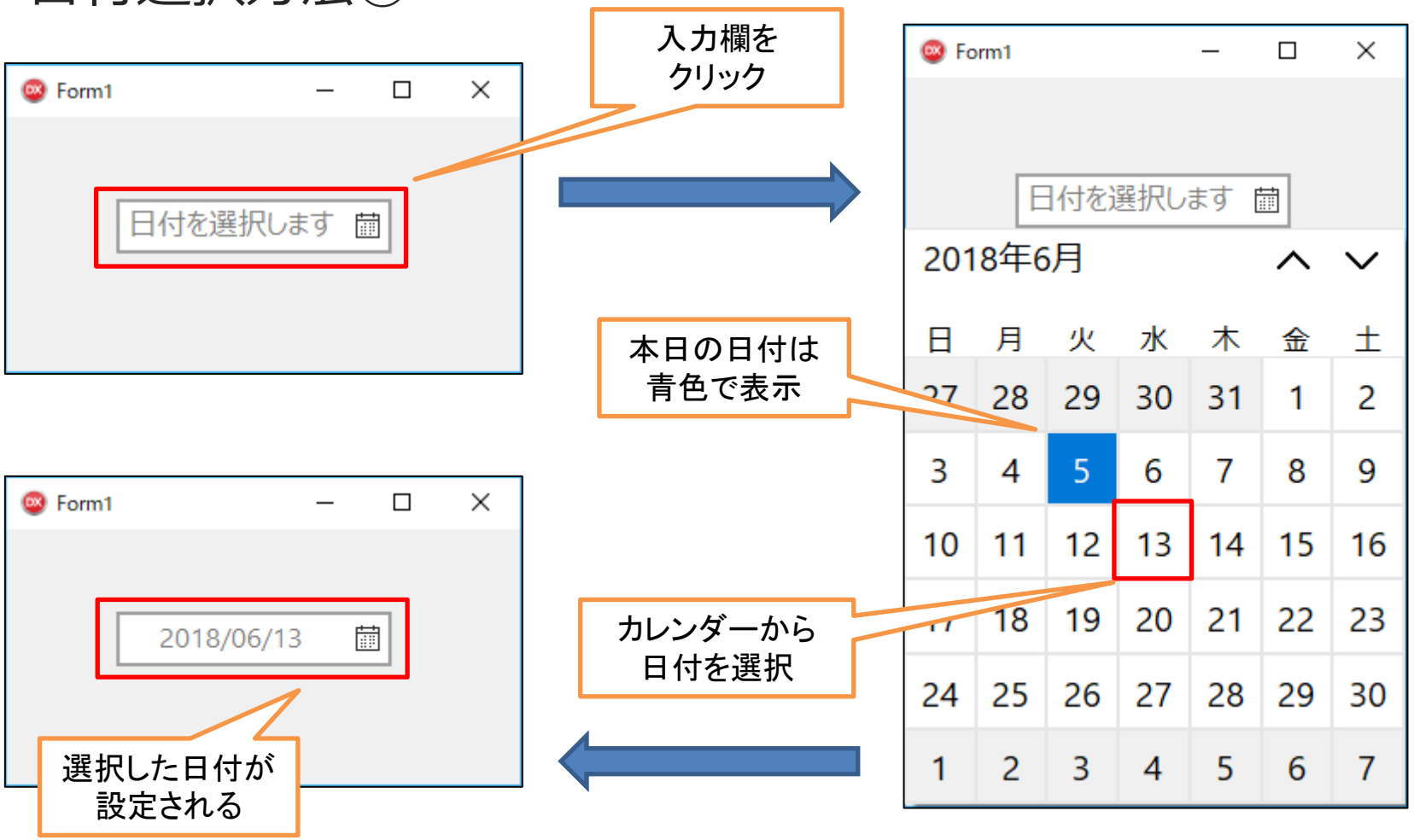


Windows10の通知領域のカレンダー（例）



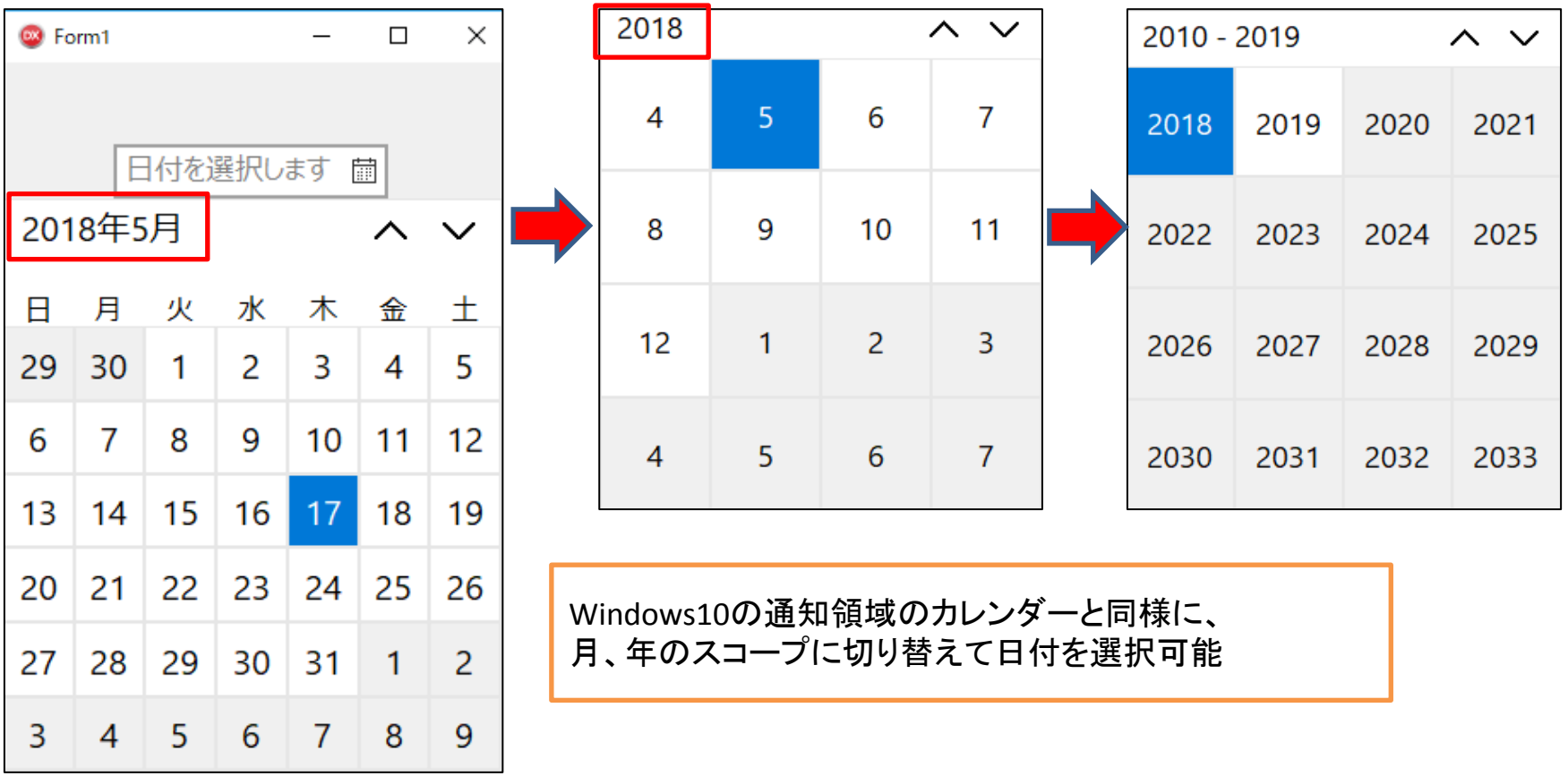
TCalendarPickerコンポーネントの使い方

- 日付選択方法①



TCalendarPickerコンポーネントの使い方

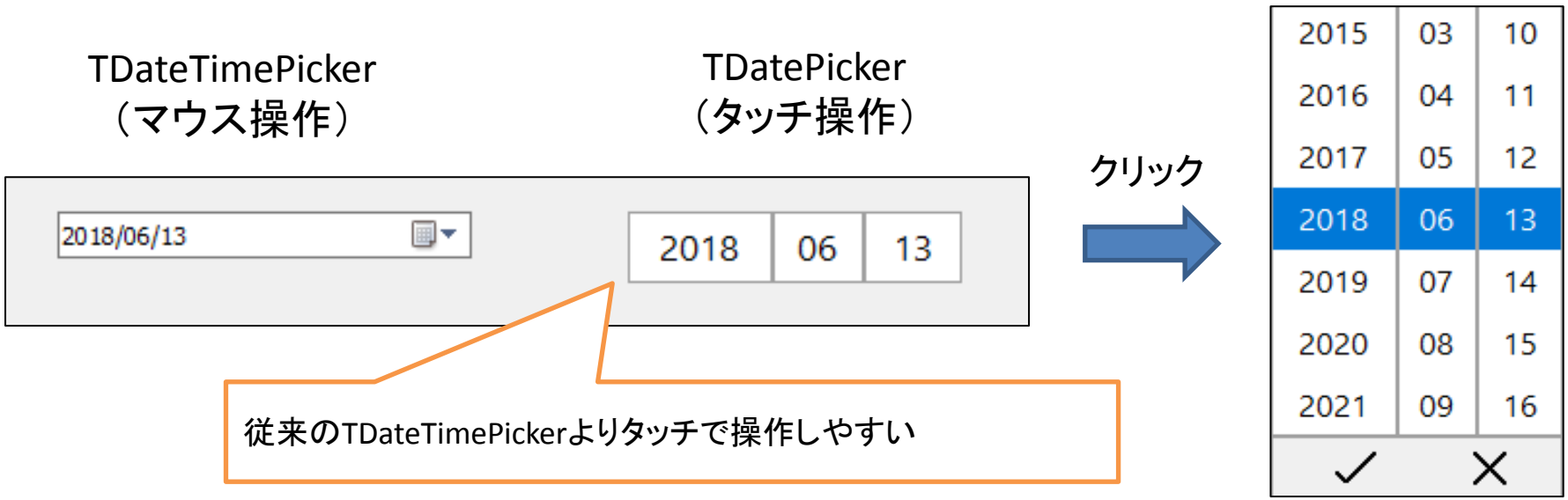
- 日付選択方法②



Windows10の通知領域のカレンダーと同様に、
月、年のスコープに切り替えて日付を選択可能

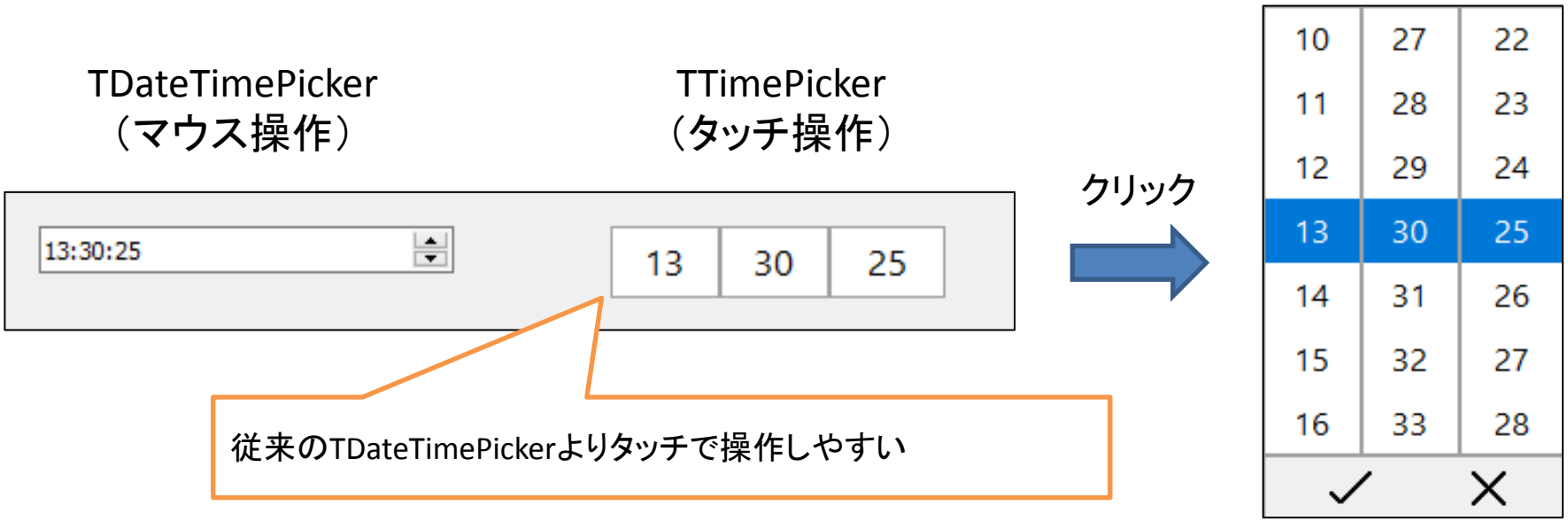
■ TDatePickerコンポーネント（Delphi/400 10.2 Tokyo～）

- 機能概要
  ドロップダウンで日付を選択することができるコンポーネント。
  従来のTDateTimePickerに比べて選択部分が大きくなっており、
  タッチの操作も行いやすくなっている。



TTimePickerコンポーネント（Delphi/400 10.2 Tokyo～）

- 機能概要
  ドロップダウンで時刻を選択することができるコンポーネント。
  従来のTDateTimePickerに比べて選択部分が大きくなっており、
  タッチの操作も行いやすくなっている。



■ 補足：新元号への対応

- 2019年5月以降の新元号の表示
  Windows10ではApril 2018 Update（1803）で新元号への対応として期間を保持しているレジストリの追加が行われている。

HKEY_LOCAL_MACHINE¥SYSTEM¥CurrentControlSet¥Control¥Nls¥Calendars¥Japanese¥Eras

名前	種類	データ
(既定)	REG_SZ	(値の設定なし)
1868 01 01	REG_SZ	明治_明_Meiji_M
1912 07 30	REG_SZ	大正_大_Taisho_T
1926 12 25	REG_SZ	昭和_昭_Showa_S
1989 01 08	REG_SZ	平成_平_Heisei_H
2019 05 01	REG_SZ	？？_？_??????_?

新元号への変更日は決まっているが
元号自体はまだ決まっていないため
「？？」として登録されている

■ 補足：新元号への対応

- Delphi/400での新元号の使用
  Delphi/400では西暦⇒和暦、和暦⇒西暦など日付の変換用メソッドが用意されており、このレジストリの値を参照している。

  そのため、メソッドで日付の変換を行っていた場合は新元号が開始した後も、これまでと同様に変換を行えるが、「平成XX年」のように固定でコーディングしていた場合はプログラムの変更が必要。

■ 補足：新元号への対応

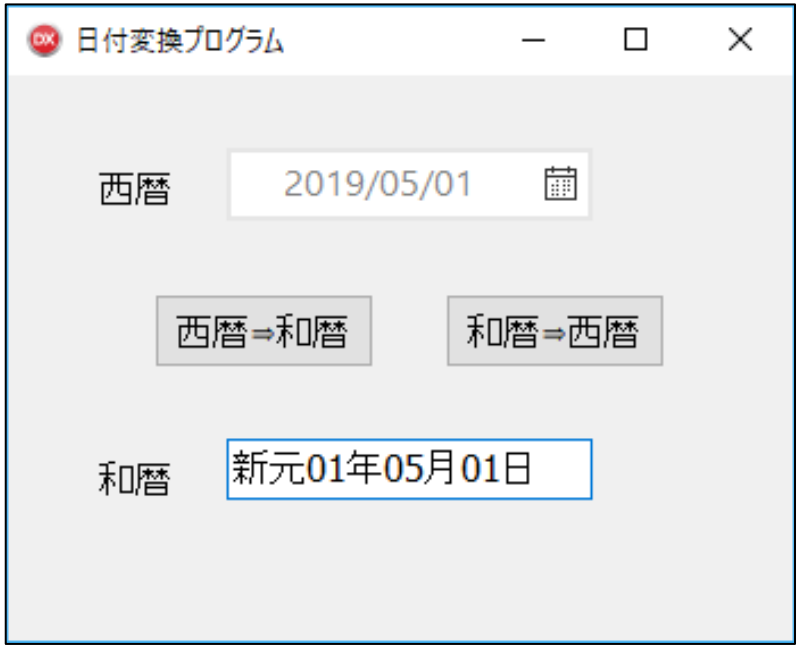
- 日付変換メソッドを使用した実装方法

例）西暦⇔和暦の変換プログラム

HKEY_LOCAL_MACHINE¥SYSTEM¥CurrentControlSet¥Control¥Nls¥Calendars¥Japanese¥Eras

名前	種類	データ
(既定)	REG_SZ	(値の設定なし)
1868 01 01	REG_SZ	明治_明_Meiji_M
1912 07 30	REG_SZ	大正_大_Taisho_T
1926 12 25	REG_SZ	昭和_昭_Showa_S
1989 01 08	REG_SZ	平成_平_Heisei_H
2019 05 01	REG_SZ	新元_新_New_N

※注意点※
本来、このレジストリは今後のWindowsアップデートにより新元号の値が設定されると思われますが、
今回は新元号への変換結果をご確認頂くために
一時的にレジストリの値を変更しております。



■ 補足：新元号への対応

- 西暦　⇒　和暦　の変換
  西暦から和暦の変換は、DateTimeToStringメソッドで可能。

和暦のフォーマット例　：　ggee年mm月dd日
gg　：　元号　　ee　：　年
mm　：　月　　dd　：　日

コーディング例

```
procedure TForm1.Button1Click(Sender: TObject);
var
  dateStr: String;
begin
  DateTimeToString(dateStr, 'ggee年mm月dd日', CalendarPicker1.Date);
  Edit1.Text := dateStr;
end;
```

■ 補足：新元号への対応

- 和暦　⇒　西暦　の変換
  和暦から西暦は、TFormatSettings型の変数で、事前にフォーマットを設定しておくことで変換可能。

コーディング例

```
procedure TForm1.Button2Click(Sender: TObject);
var
  LocaleSettings: TFormatSettings;
  dateStr: String;
  dt: TDateTime;
begin
  //入力値に対するチェック
  LocaleSettings := TFormatSettings.Create;
  LocaleSettings.ShortDateFormat := 'ggee/mm/dd';
  dateStr := Edit1.Text;
  //ggee/mm/ddの形式に変換
  dateStr := dateStr.Replace('年', '/');
  dateStr := dateStr.Replace('月', '/');
  dateStr := dateStr.Replace('日', '');
  dt := StrToDate(dateStr, LocaleSettings);
  CalendarPicker1.Date := dt;
end;
```

4.ジェスチャに対応するコンポーネント

■ ジェスチャとは？

- ジェスチャ
  ジェスチャはタッチパネルなどのタッチにおける指の動作の軌跡などを検出してイベントを発生させるもの。
  もちろんマウスのフォーカスで同様の動作も可能。

TGestureManagerコンポーネント(Delphi/400 V2010～)

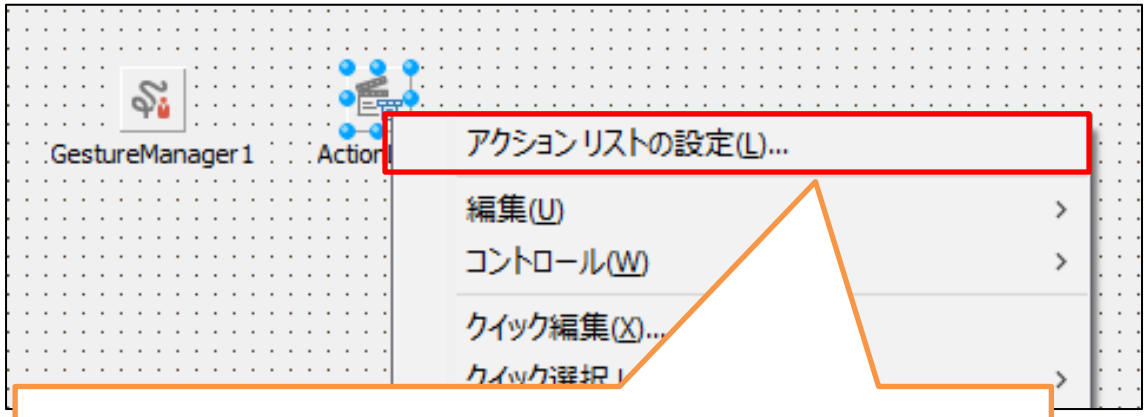
- 機能概要
  アプリケーションにおけるジェスチャを管理するコンポーネント。
  あらかじめ登録されているジェスチャの他に、新しいジェスチャを
  登録して使用することも可能。

  ジェスチャを他のコンポーネントと組み合わせることで、ジェスチャに
  対応した処理を行わせることもできる。

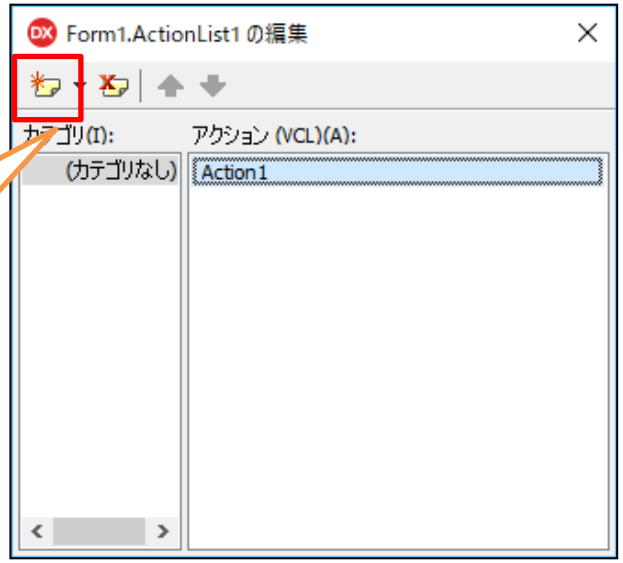
TGestureManagerコンポーネントの使い方

- ジェスチャの使用方法①

例）TCardPanelのカード切り替えと組み合わせる場合



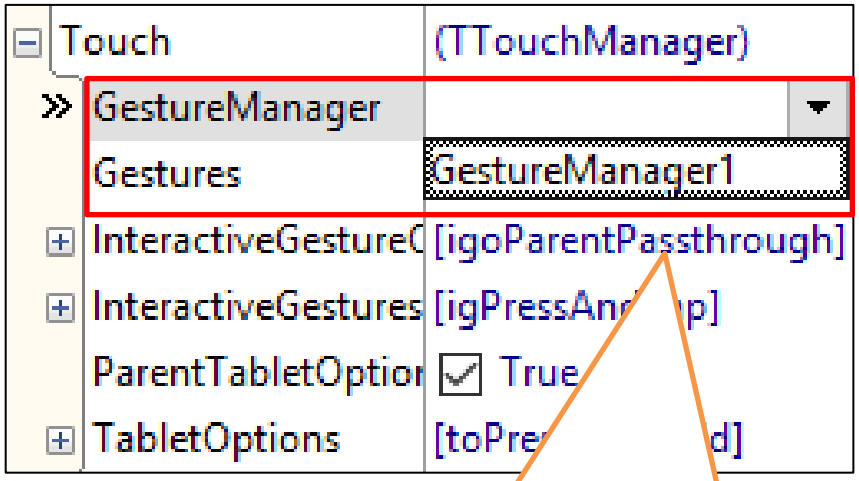
① TActionListを配置し、
コンポーネント上で右クリック⇒「アクションリストの設定」



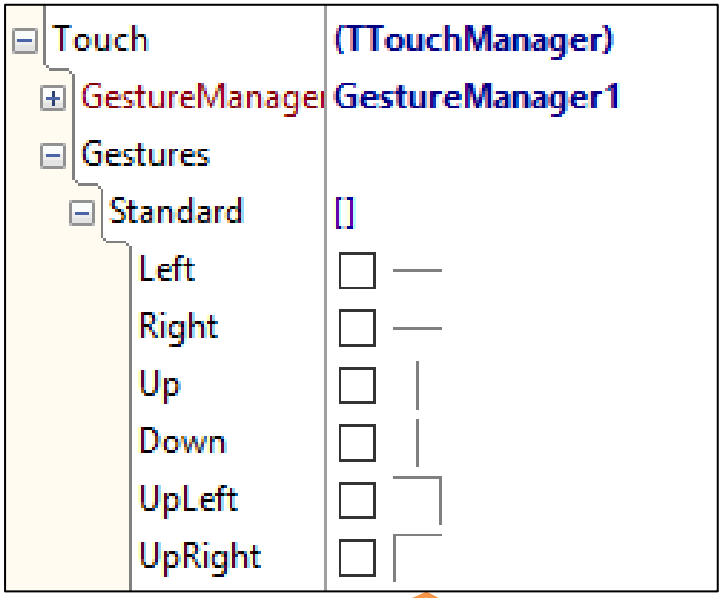
② アクションリストの編集画面で新規登録を選択し
アクションを登録する

TGestureManagerコンポーネントの使い方

- ジェスチャの使用方法②



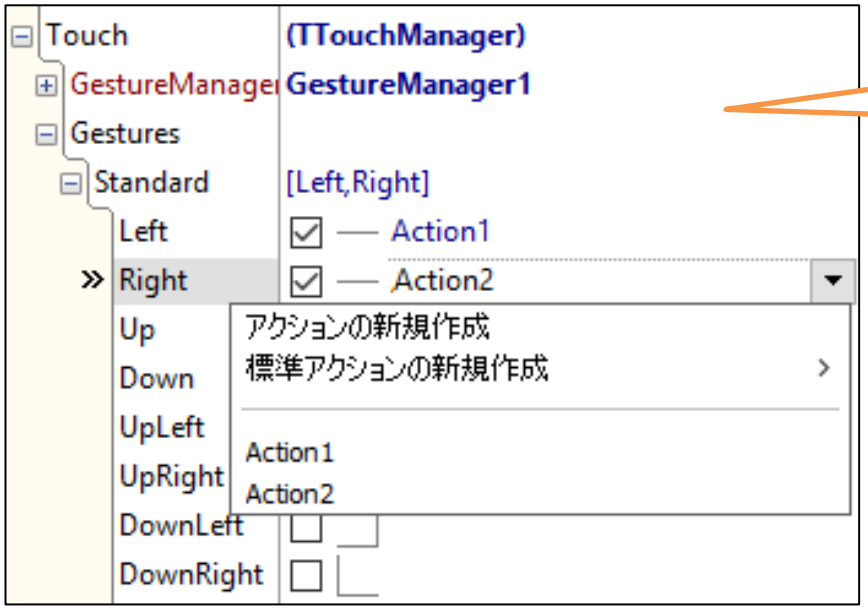
③ ジェスチャを設定したいコンポーネントの
Touchプロパティを開き、GestureManagerプロパティに
作成したTGestureManagerを指定する



④ Gesturesプロパティから、使用するジェスチャを選択する
※Standardには既存のジェスチャが登録されている

TGestureManagerコンポーネントの使い方

- ジェスチャの使用方法③



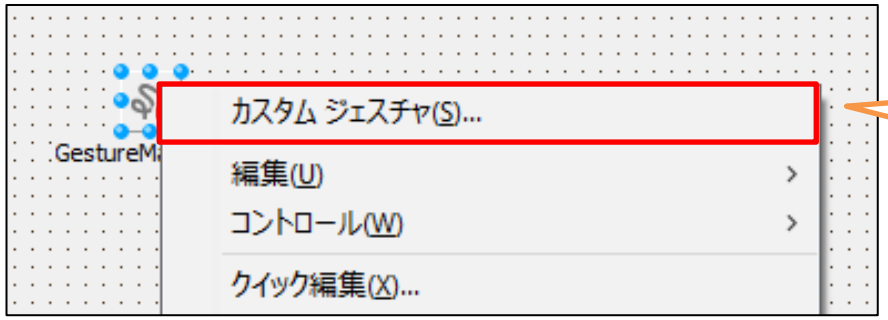
⑤ 使用するジェスチャにチェックを入れ、
その右のプルダウンで紐付けるアクションを選択する

⑥ 紐付けたアクションのExecuteイベントに
処理内容を記載する

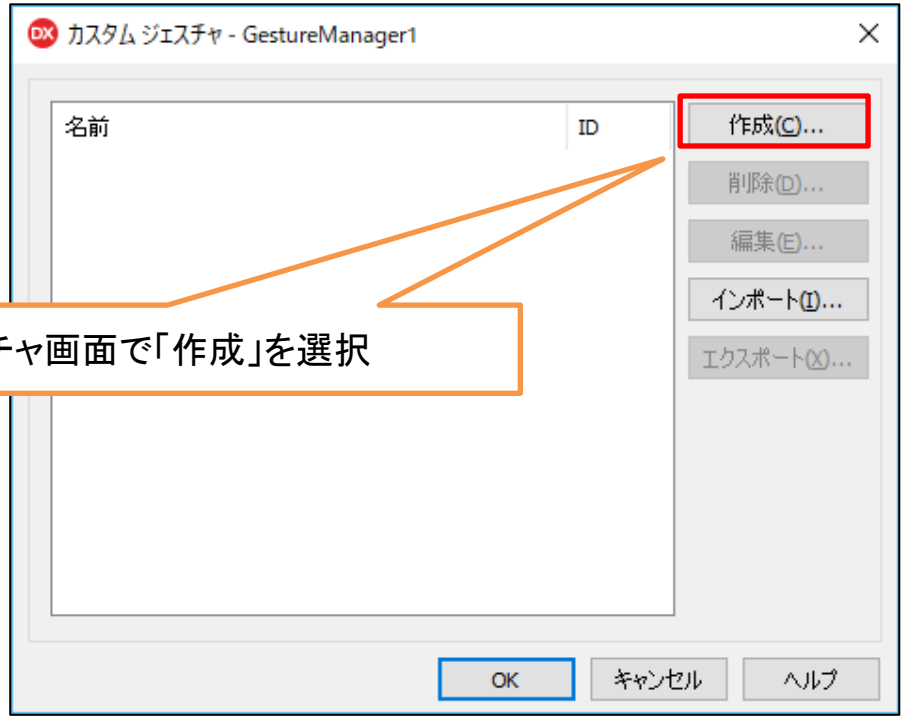
```
90  procedure TForm1.Action1Execute(Sender: TObject);
    begin
      CardPanel1.NextCard;
    end;

    procedure TForm1.Action2Execute(Sender: TObject);
    begin
      CardPanel1.PreviousCard;
    end;
```

TGestureManagerコンポーネントの使い方

- ジェスチャの登録方法①



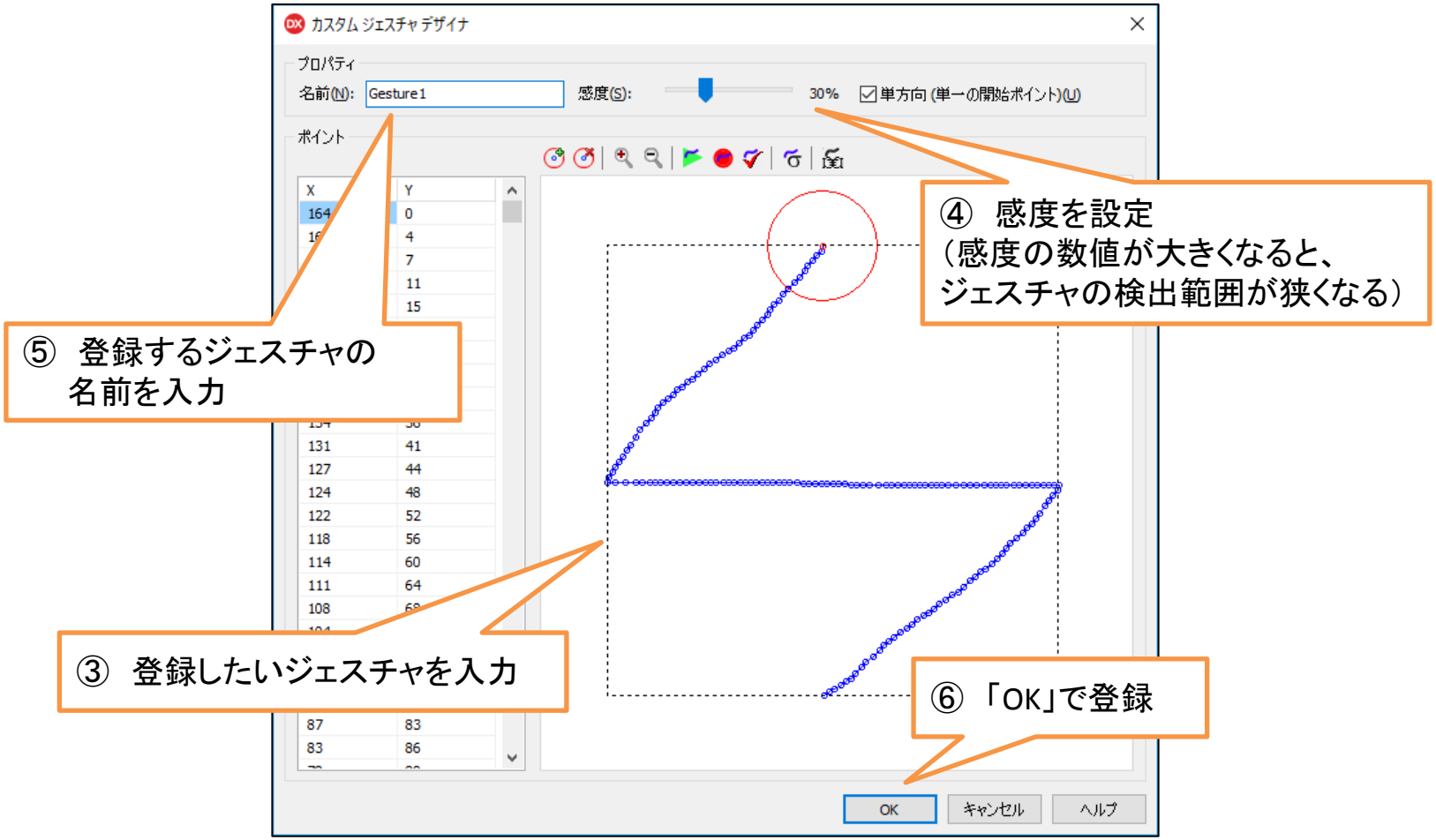
① TGestureManagerを配置し、
コンポーネント上で右クリック⇒「カスタムジェスチャ」



② カスタムジェスチャ画面で「作成」を選択

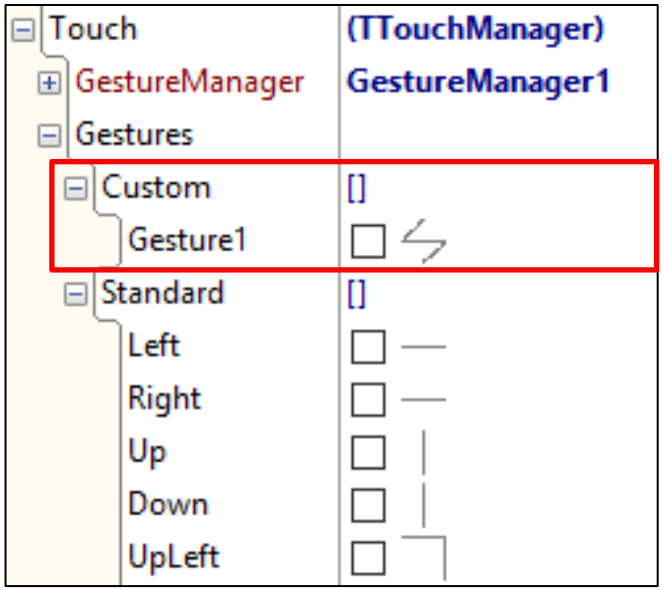
TGestureManagerコンポーネントの使い方

- ジェスチャの登録方法②

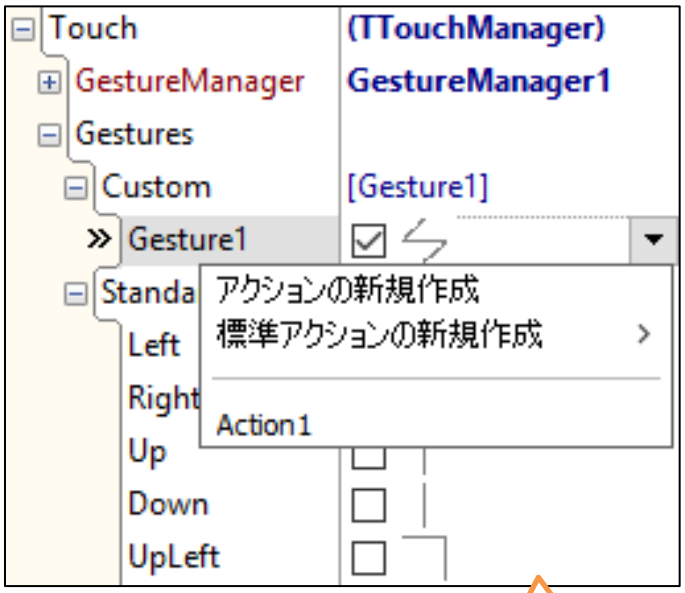


TGestureManagerコンポーネントの使い方

- ジェスチャの登録方法③



⑦ 新しく登録したジェスチャは
「Custom」に表示されている



⑧ 新しく登録したジェスチャにも
アクションを紐付けることができる
（対応した処理を行うことが可能）

5.まとめ

■ まとめ

- レスポンシブデザインの画面を開発する場合はTStackPanel TGridPanel、TFlowPanelコンポーネントが有効。
- Windowsタブレットで画面のサイズが制限される場合はTSplitView、TCardPanelコンポーネントを利用することで画面表示が工夫できる。
- Windowsタブレットで使用する画面は従来のコンポーネントに代わるTToggleSwitch、TCalendarPicker、TDatePicker、TTimePickerコンポーネントを使った方が操作性が良い。
- タッチ操作のジェスチャはTGestureManagerコンポーネントで制御することができる。

ご清聴ありがとうございました。