【セッションNo．1】

VCLを使用した Windowsタブレット向けアプリ構築術

株式会社ミガロ.
RAD事業部　技術支援課
尾崎　浩司

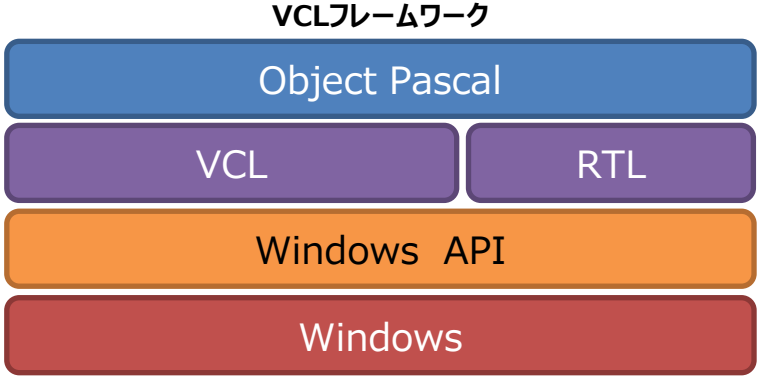
【アジェンダ】

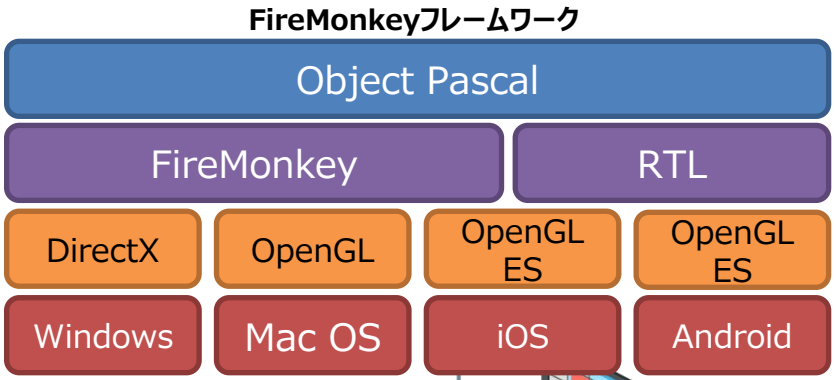
- Delphi/400におけるモバイルアプリ開発
- Windowsタブレットの特徴
- VCLを使用したWindowsタブレットアプリ構築術
  - カスタムソフトキーボードの作成
  - 入力項目に最適化した入力欄の作成
  - カメラの活用
  - QRコードスキャン
- まとめ

Delphi/400における
モバイルアプリ開発

Delphi/400におけるモバイルアプリ開発

- VCLとFireMonkey
  - VCL
    - 25年以上にわたって活用されているWindowsアプリ構築用フレームワーク
  - FireMonkey
    - Delphi XEシリーズより登場したマルチデバイスアプリ構築用フレームワーク





Delphi/400におけるモバイルアプリ開発

- FireMonkeyを使用したモバイルアプリ開発
  - **マルチデバイスアプリ**　（ネイティブアプリ）
    - iOS、Android、Windows(タブレット) いずれのOSにも対応
    - カメラやGPS等デバイス機能の活用
    - 「マルチデバイスデザイナ」により、デバイスに合わせた画面カスタマイズ

    - FireMonkeyの考慮点
      - 各OS固有の機能（API）への直接アクセスは原則不可
      - 特にiOS/Androidは頻繁なOSバージョンアップがあり、それに追随する為にはアプリの定期的なメンテナンスが必要　（但し多くは互換性により動作）
      - iOS/Androidの場合、データベースドライバが使用できない為、Rad ServerやDataSnapといった3層アプリフレームワークの組み合わせが必要

Delphi/400におけるモバイルアプリ開発

- VCLを使用したモバイルアプリ開発
  - **Webアプリ**
    - IntraWeb等を使用することで、各種OS上のブラウザから実行できるWebアプリを構築可能
    - デバイス機能の活用は制限され、操作性もネイティブアプリには劣る事が多い
  - **VCLフォームアプリ**（ネイティブアプリ）
    - 実行デバイスは、**Windows(タブレット)**に制限される
    - 従来の**PC向けアプリと同様の仕組み**（C/S型）がそのまま活かせる
    - IBM i への直接アクセスが可能な為、**アプリケーションサーバーの設置は不要**

今回は、VCLを使用したモバイルアプリについてご紹介！

Windowsタブレットの特徴

■ Windowsタブレットの特徴

- モバイルデバイスとして、Windowsタブレットを使用するメリット
  - 業務で使用するOfficeが、そのままフル機能使用できる
  - Office以外のPCで使用している各種ソフトも、全て使用できる
  - アプリの同時実行性に優れる（マルチタスク型）
  - 業務に求められるセキュリティが確保しやすい　(ドメイン認証が可能)
  - Windows用の周辺機器が、使用できる（接続端子の確認は必要）
  - 用途や予算にあわせて多彩な選択肢がある

Microsoft Surface

快適性に優れる

耐久性に優れる

Panasonic TOUGHBOOK

コスパに優れる

Lenovo ideapad

■ Windowsタブレットの特徴

- Windowsタブレットの留意点
  - Windows PCと同じように使用できるが、
    主にデスクで使用するPCとは実行環境が違う

  - 例えば、生産現場や出荷現場等で活用する場合
    - キーボードレスによるタッチ操作での利用となる
    - カメラやQRコードの使用するといったモバイル機能の活用が要求される

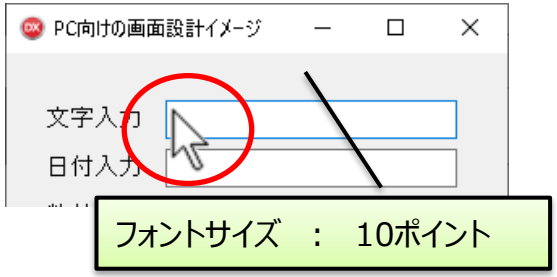
今回のポイント

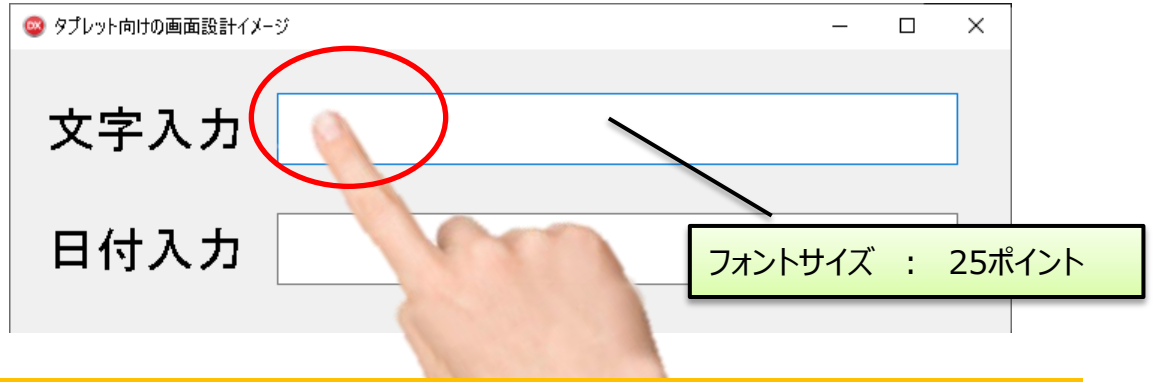
- タブレットに最適な画面サイズの設定
- 独自ソフトキーボードの使用や効率的な入力方法
- カメラやQRコードスキャンの活用術

VCLを使用したWindows タブレットアプリ構築術

■ VCLを使用したタブレットアプリ構築術

- PCとタブレットの画面設計の違い
  - PC　　：マウス操作が主体　→　正確な操作が可能
  - タブレット　：タッチ操作が主体　→　指タッチの為、狙ったポイントを外しやすい





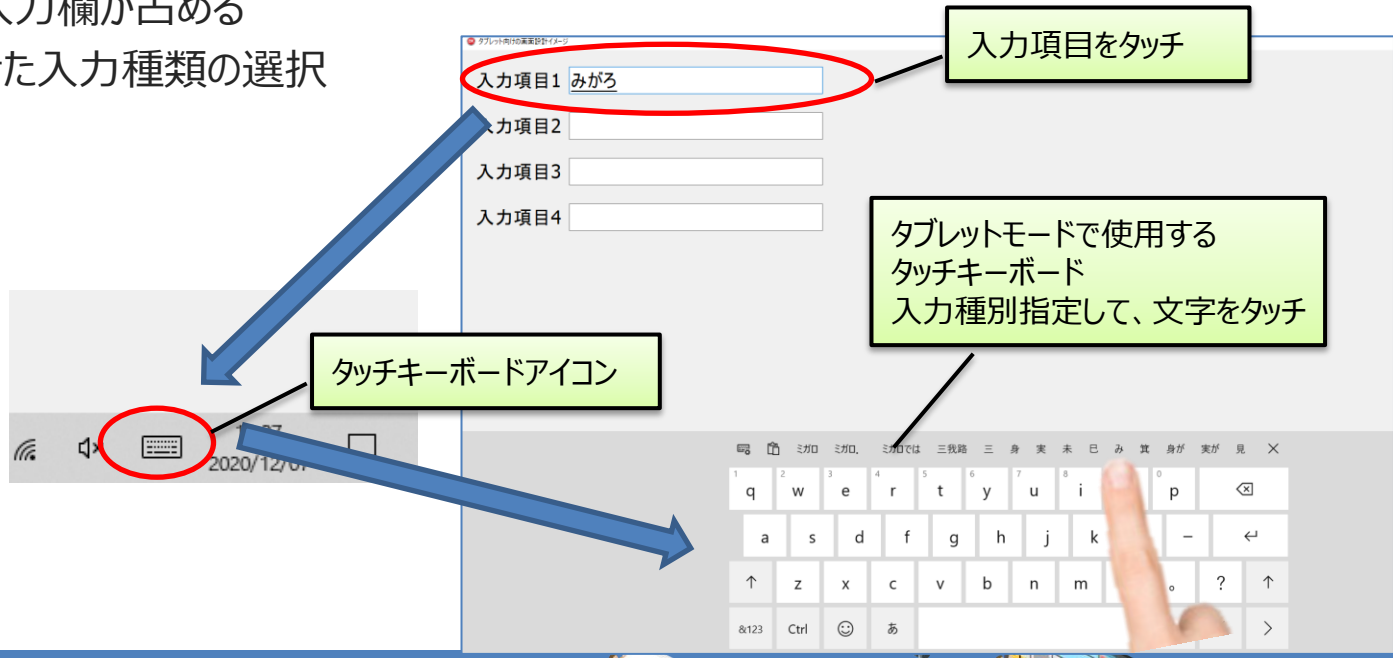
① 項目自体のサイズやフォントサイズを大きめに設計する
② 項目間の余白を広めにとる
③ 出来る限り、コンボボックスやラジオボタン等選択を中心とした入力を意識する

■ VCLを使用したタブレットアプリ構築術

- タッチ操作の入力項目
  - タブレットモードの場合、通常はWindowsのタッチキーボードを使用
    - 画面下部を入力欄が占める
    - 項目に合わせた入力種類の選択

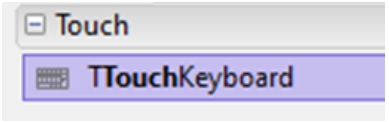
操作手順

項目選択
↓
キーボード起動
↓
入力種別選択
（文字、数字等）

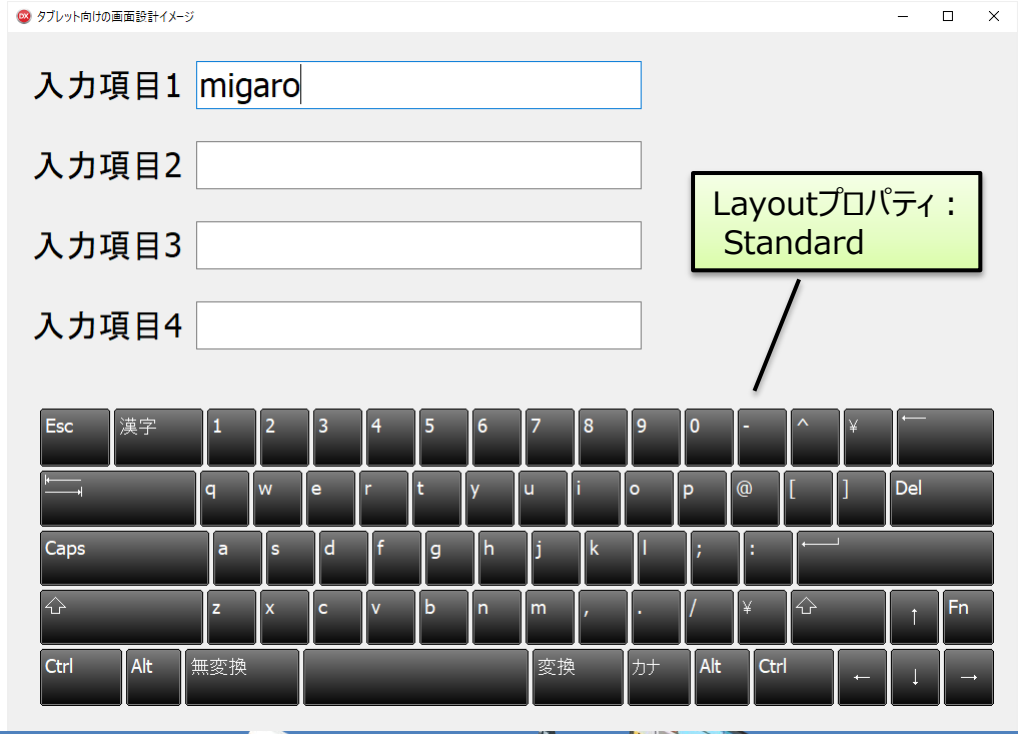


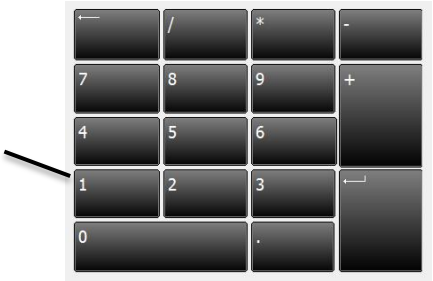
VCLを使用したタブレットアプリ構築術

- Delphiに用意されたタッチキーボードコンポーネント
  - TTouchKeyBoard



  - フォーム上に直接タッチキーボードが配置可能な為、キーボード込みでのレイアウト調整がしやすい

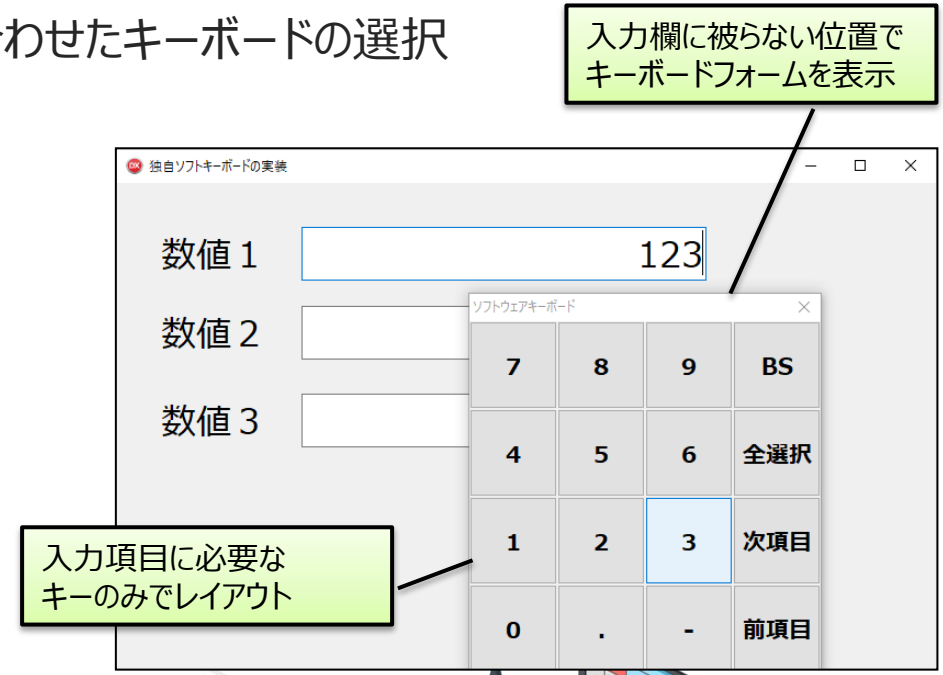




VCLを使用したタブレットアプリ構築術

- Windowsタッチキーボード、TTouchKeyBoardの課題
  - 項目多い画面でのキーボード領域確保
  - 項目の種類（金額/日付/文字）に合わせたキーボードの選択

- 別ウィンドウで表示可能な
  カスタムソフトキーボードにより対応
  - キー表示位置の自動調整
  - 項目に最適な入力パッドの作成



■ VCLを使用したタブレットアプリ構築術

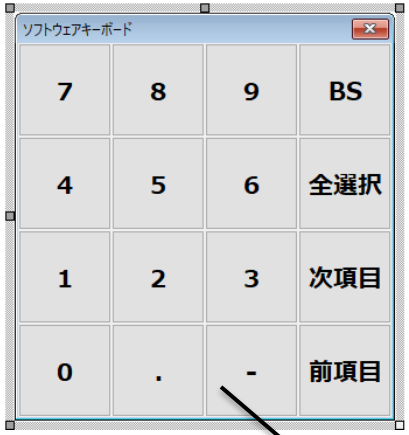
- カスタムソフトキーボードの作成
  - キーパッドフォームの作成

宣言部

```
type
  TfrmKeyBoard = class(TForm)
       ...(省略)...
  private
    { Private 宣言 }
  public
    { Public 宣言 }
    DeActivateForm: TForm; //入力フォーム
  end;
```

キーコードを送信する
フォームを保持する変数

キーパッドフォーム設計画面



btnKeyX: TBitBtn
X: 1,2,3...

フォーム生成(OnCreateイベント)

```
procedure TfrmKeyBoard.FormCreate(Sender: TObject);
begin
  // TBitBtnのTagプロパティを設定
  btnKey0.Tag      := Ord('0');
  btnKey1.Tag      := Ord('1');
  btnKey2.Tag      := Ord('2');
  btnKey3.Tag      := Ord('3');
  btnKey4.Tag      := Ord('4');
  btnKey5.Tag      := Ord('5');
  btnKey6.Tag      := Ord('6');
  btnKey7.Tag      := Ord('7');
  btnKey8.Tag      := Ord('8');
  btnKey9.Tag      := Ord('9');
  btnKeyDot.Tag    := Ord('.');
  btnKeyMinus.Tag  := Ord('-');
  btnKeyBS.Tag     := VK_BACK;  //BackSpace
  btnNext.Tag      := 0;        //次の項目へ移動
  btnPrior.Tag     := 1;        //前の項目へ移動

  // ソフトウェアキーボードの初期表示位置を設定
  Top  := Screen.WorkAreaTop  + Screen.WorkAreaHeight - Height;
  Left := Screen.WorkAreaLeft + Screen.WorkAreaWidth  - Width;
end;
```

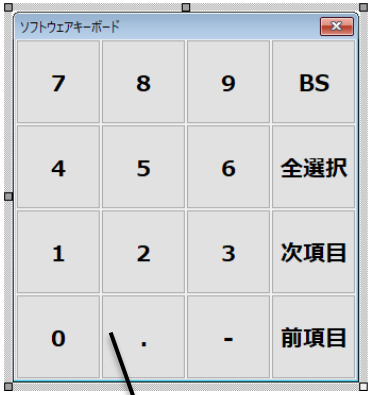
Tagプロパティに
キーコードをセット

ディスプレイの右下位置に
フォームを初期表示

VCLを使用したタブレットアプリ構築術

各ボタンクリック(OnClickイベント)

キーパッドフォーム設計画面



各ボタンのOnClickイベント割り当て
0…9,.,-,BS　：btnKeyBoardClick
全選択　：btnSelectALL
次項目、前項目　：btnFocusControl

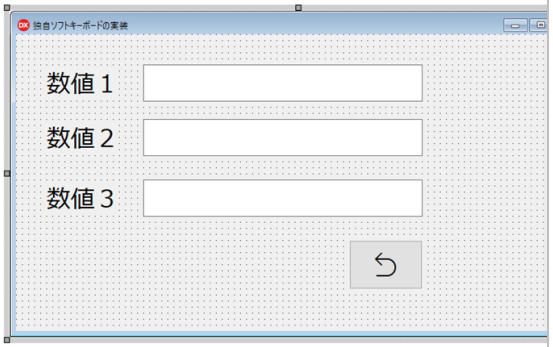
```
{*******************************************************************************
    キーボードクリック時
*******************************************************************************}
procedure TfrmKeyBoard.btnKeyBoardClick(Sender: TObject);
begin
  if Assigned(DeActivateForm) then
  begin
    DeActivateForm.BringToFront;
    PostMessage(DeActivateForm.ActiveControl.Handle, WM_CHAR, TBitBtn(Sender).Tag, 0);
  end;
end;
{*******************************************************************************
    全選択クリック時
*******************************************************************************}
procedure TfrmKeyBoard.btnSelectALL(Sender: TObject);
begin
  if Assigned(DeActivateForm) then
  begin
    DeActivateForm.BringToFront;
    PostMessage(DeActivateForm.ActiveControl.Handle, EM_SETSEL, 0, -1);
  end;
end;
{*******************************************************************************
    次項目/前項目クリック時
*******************************************************************************}
procedure TfrmKeyBoard.btnFocusControl(Sender: TObject);
begin
  if Assigned(DeActivateForm) then
  begin
    DeActivateForm.BringToFront;
    PostMessage(DeActivateForm.Handle, WM_NEXTDLGCTL, TBitBtn(Sender).Tag, 0);
  end;
end;
```

送信先フォームを前面にして、
入力キー情報をPostMessageで送信

■ VCLを使用したタブレットアプリ構築術

- カスタムソフトキーボードの作成
  - 入力用フォームの実装例

入力フォーム設計画面



実装部

```
{*************************************************************************
    フォーム終了時
*************************************************************************}
procedure TfrmEntry.FormClose(Sender: TObject; var Action: TCloseAction);
begin
  Action := caFree;  //フォーム及びキーボードフォームの破棄
end;
{*************************************************************************
    フォーム生成時
*************************************************************************}
procedure TfrmEntry.FormCreate(Sender: TObject);
begin
  frmKeyBoard := TfrmKeyBoard.Create(Self);
end;
{*************************************************************************
    画面非アクティブ時
*************************************************************************}
procedure TfrmEntry.FormDeactivate(Sender: TObject);
begin
  // ソフトウェアキーボードの"DeactivateForm"に自身をセット
  frmKeyBoard.DeactivateForm := Self;
end;
```

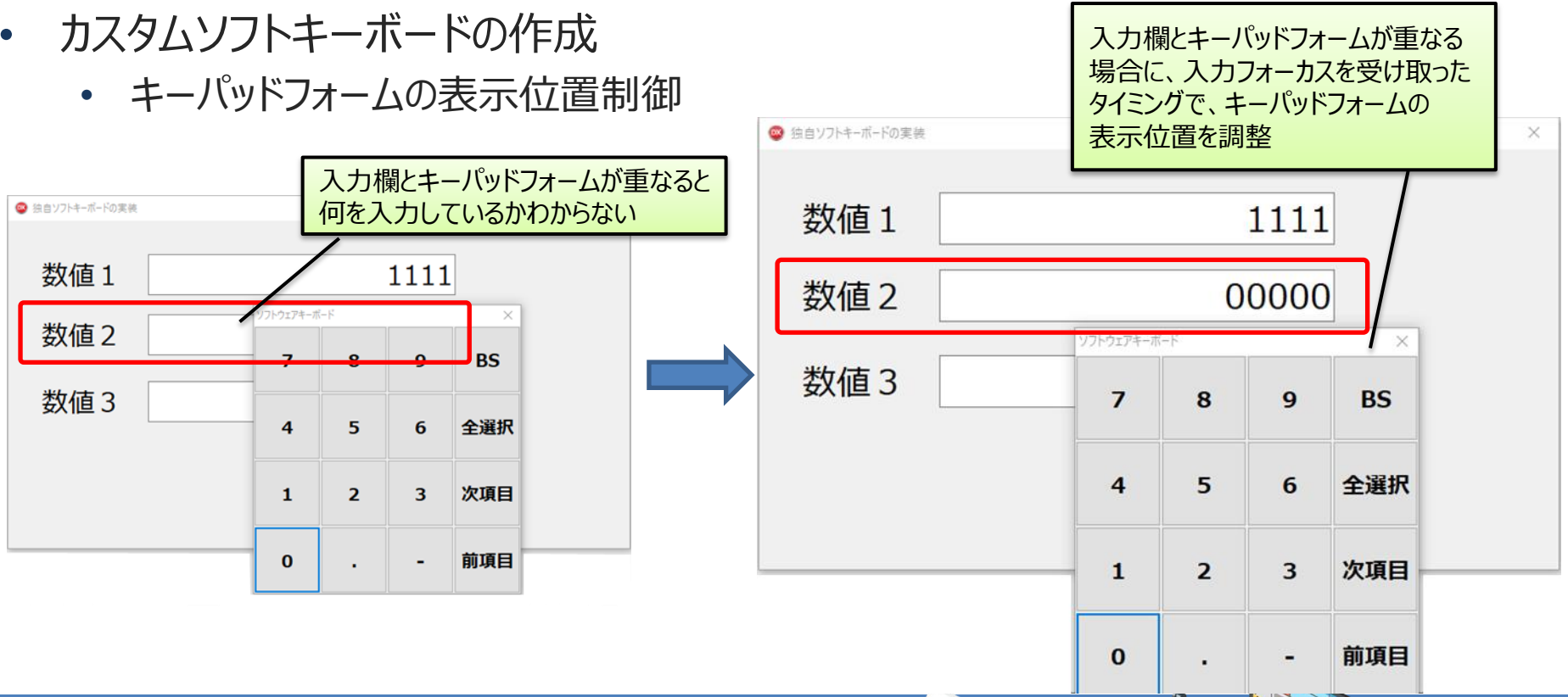
フォーム破棄に合わせてキーボードフォームも破棄

CreateのOwnerにSelfを指定

フォームからキーパッドフォームにフォーカスが移動するときに、フォーム変数にSelfをセット

■ VCLを使用したタブレットアプリ構築術

- カスタムソフトキーボードの作成
  - キーパッドフォームの表示位置制御



■ VCLを使用したタブレットアプリ構築術

- カスタムソフトキーボードの作成
  - キーパッドフォームの表示位置制御

実装部

```
procedure TfrmEntry.EditEnter(Sender: TObject);
var
  rCon, rFrm, rRes: TRect;
  pCon: TPoint;
  iBot: Integer;
begin
  if (Sender is TWinControl) and (Assigned(frmKeyBoard)) then
  begin
    // コンポーネントの位置をスクリーン座標で求める
    pCon.X := (Sender as TWinControl).Left;
    pCon.y := (Sender as TWinControl).Top;
    pCon   := Self.ClientToScreen(pCon);

    // コンポーネントのRect
    rCon.Top    := pCon.Y;
    rCon.Left   := pCon.X;
    rCon.Height := (Sender as TWinControl).Height;
    rCon.Width  := (Sender as TWinControl).Width;

    // キーボードのRect
    rFrm.Top    := frmKeyBoard.Top;
    rFrm.Left   := frmKeyBoard.Left;
    rFrm.Height := frmKeyBoard.Height;
    rFrm.Width  := frmKeyBoard.Width;
```

フォームとキーパッドフォームの
Rectを求める

重なりがある場合、キーパッドフォーム
の位置をコンポーネントの下か、上に移動

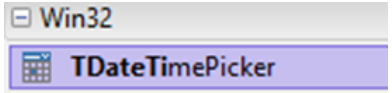
```
    // コンポーネントとフォームの重なりを求める
    IntersectRect(rRes, rCon, rFrm);
    // 重なりが有った場合キーボードを移動
    if rRes.Height > 0 then
    begin
      // キーボードを移動した結果のフォームの下位置を求める
      // [TEditのTop]＋[TEditのHeight]＋[キーボードのHeight]＋[調整値]
      iBot := rCon.Top + rCon.Height + frmKeyBoard.Height + 5;
      // キーボードがスクリーンの外にはみ出したかどうかを判断
      if iBot > Screen.Height then
        // TEditの上に移動
        frmKeyBoard.Top := rCon.Top - frmKeyBoard.Height - 5
      else
        // TEditの下に移動
        frmKeyBoard.Top := rCon.Top + rCon.Height + 5;
    end;
  end;
end;
```

■ VCLを使用したタブレットアプリ構築術

- カスタムソフトキーボードの作成
  - 実行例

■ VCLを使用したタブレットアプリ構築術

- 項目の種類に合わせて最適化した入力欄の作成
  - 日付値と時間値の入力
    - TEdit
      - 入力可能文字の制御や日付/時間妥当性チェックが必要
    - TDateTimePicker

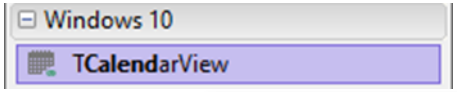


      - 日付選択がタブレットに最適化されない
      - 時間の指定がカスタマイズできない
        （5分単位入力等）
      - 時間のアップダウン部分が小さく押しにくい

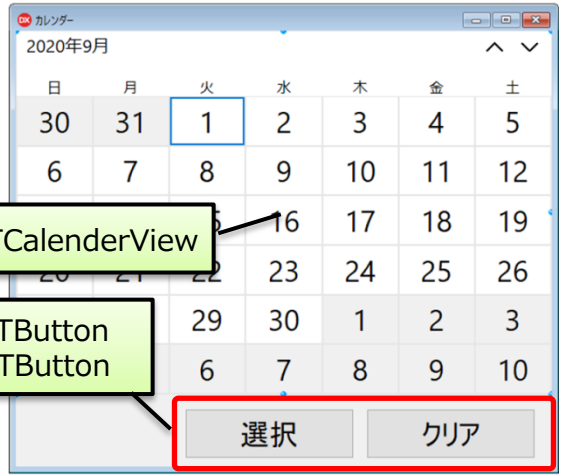
タブレットに最適な日付値、時間値の
入力方法を検討

■ VCLを使用したタブレットアプリ構築術

- 日付入力フォームの作成
  - TCalenderView



フォーム設計画面



CalenderView1: TCalenderView

[選択]: Button1: TButton
[クリア]: Button2: TButton

宣言部

遷移元画面の
日付入力
コンポーネントを
保持

```
type
  TfrmCalendar = class(TForm)
      ...(省略)...
  public
    { Public 宣言 }
    DateEdit: TEdit;
  end;
```

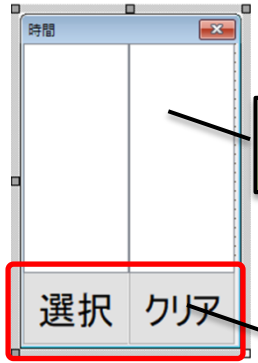
実装部

```
procedure TfrmCalendar.FormShow(Sender: TObject);
var
  sDate: String;  // システム日付
begin
  // システム日付の初期セット
  sDate := FormatDateTime('YYYY/MM/DD', Now);
  CalendarView1.Date := StrToDate(sDate);
  // 遷移元画面の日付≠ブランクの場合、遷移元画面の日付をセット
  if DateEdit.Text <> EmptyStr then
    CalendarView1.Date := StrToDate(DateEdit.Text);
end;

procedure TfrmCalendar.Button1Click(Sender: TObject);
begin
  // 遷移元画面の日付に選択値をセット
  DateEdit.Text := FormatDateTime('YYYY/MM/DD', CalendarView1.Date);
  // 画面を終了する
  Self.Close;
end;

procedure TfrmCalendar.Button2Click(Sender: TObject);
begin
  // 遷移元画面の日付をクリア
  DateEdit.Text := EmptyStr;
  // 画面を終了する
  Self.Close;
end;
```

■ VCLを使用したタブレットアプリ構築術

- 時間入力フォームの作成

フォーム設計画面



[時]: ListBox1: TListBox
[分]: ListBox2: TListBox

[選択]: Button1: TButton
[クリア]: Button2: TButton

宣言部

遷移元画面の時間入力コンポーネントを保持

```
type
  TfrmTime = class(TForm)
       ...(省略)...
  public
    { Public 宣言 }
    TimeEdit: TEdit;
  end;
```

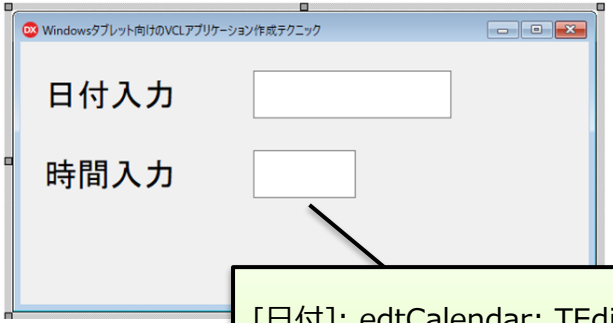
実装部

```
procedure TfrmTime.FormShow(Sender: TObject);
var
  i: Integer;
  sTime: String;
begin
  // 初期化
  for i := 0 to 23 do     // 時のセット(0-23)
    ListBox1.Items.Add(FormatFloat('00', i));
  for i := 0 to 11 do     // 分のセット(5分おき)
    ListBox2.Items.Add(FormatFloat('00', i * 5));
  // システム時間の取得（5分以下切捨て）
  sTime := FormatDateTime('HH:MM', Now);
  i := StrToInt(Copy(sTime, 5, 1));  // 分の下1桁
  sTime := Copy(sTime, 1, 4) + IntToStr((i div 5) * 5);
  // 遷移元画面の時間≠ブランクの場合、遷移元画面の時間をセット
  if TimeEdit.Text <> EmptyStr then
    sTime := TimeEdit.Text;
  //画面に値をセット
  ListBox1.ItemIndex := ListBox1.Items.IndexOf(Copy(sTime, 1, 2));
  ListBox1.TopIndex  := ListBox1.ItemIndex;
  ListBox2.ItemIndex := ListBox2.Items.IndexOf(Copy(sTime, 4, 2));
  ListBox2.TopIndex  := ListBox2.ItemIndex;
end;

procedure TfrmTime.Button1Click(Sender: TObject);
begin
  TimeEdit.Text := ListBox1.Items[ListBox1.ItemIndex] + ':' // 時
                 + ListBox2.Items[ListBox2.ItemIndex];      // 分
  Self.Close;
end;
```

■ VCLを使用したタブレットアプリ構築術

- 入力用フォームの実装例

フォーム設計画面



[日付]: edtCalendar: TEdit
[時間]: edtTime: TEdit

ReadOnly = True
OnClickイベントを定義

Editコンポーネントの下部に
日付入力フォームを表示

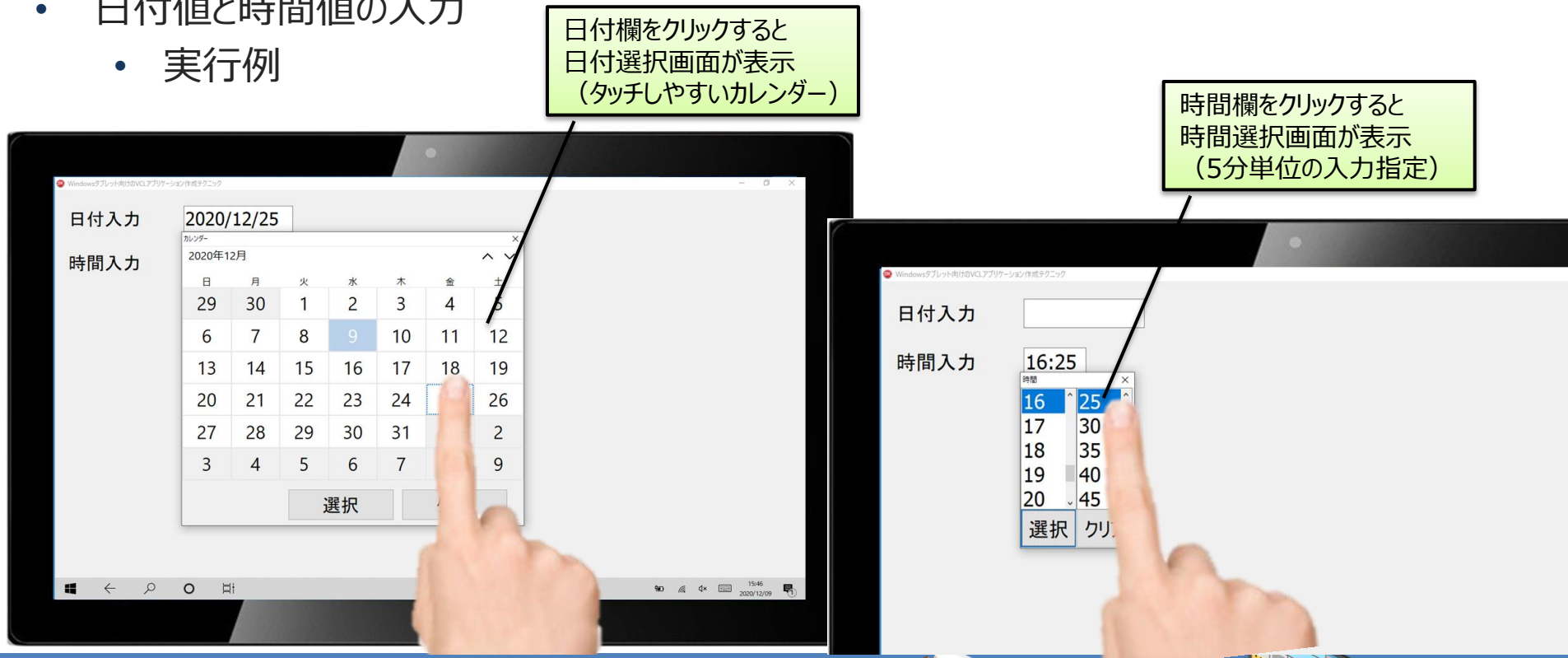
実装部

```
procedure TfrmEntry.edtCalendarClick(Sender: TObject);
begin
  // カレンダーフォームの起動
  frmCalendar := TfrmCalendar.Create(Self);
  try
    frmCalendar.DateEdit := edtCalendar; //日付値をセットするTEditを指定
    // Left位置調整
    frmCalendar.Left := Self.Left                          // 画面のLeft位置
                      + edtCalendar.Left;                  // EditのLeft位置
    // Top位置調整
    frmCalendar.Top := GetSystemMetrics(SM_CYCAPTION)      // タイトルの高さ
                     + Self.Top                            // 画面のTop位置
                     + edtCalendar.Top                     // EditのTop位置
                     + edtCalendar.Height;                 // Editの高さ
    // カレンダーフォームの表示
    frmCalendar.ShowModal;
  finally
    frmCalendar.Release;
  end;
end;
```

※　時間入力欄も同様の処理を実装

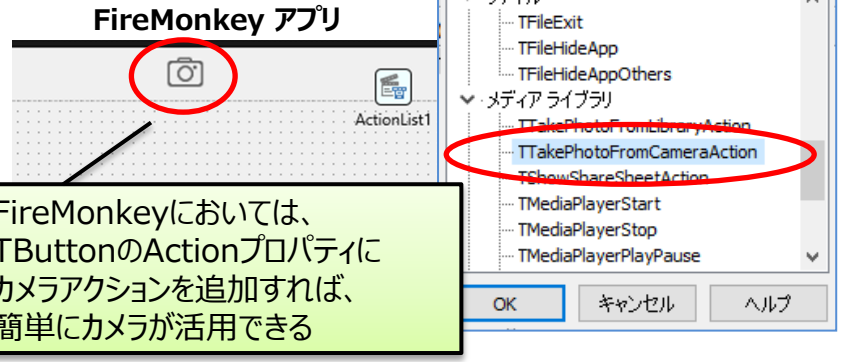
■ VCLを使用したタブレットアプリ構築術

- 日付値と時間値の入力
  - 実行例

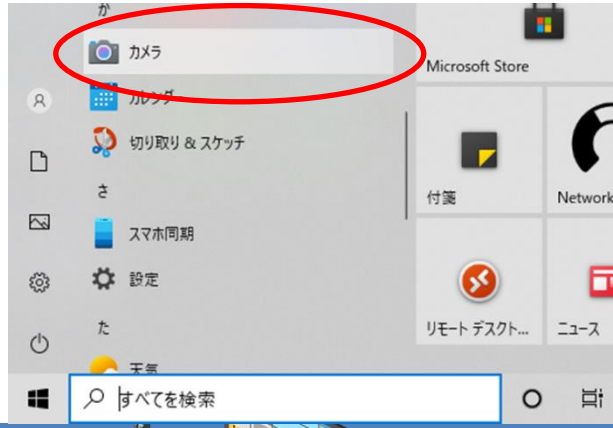


■ VCLを使用したタブレットアプリ構築術

- カメラの活用
  - TTakePhotoFromCameraAction
    - FireMonkeyのみ(VCLには無い)
    - Windowsでは使用不可



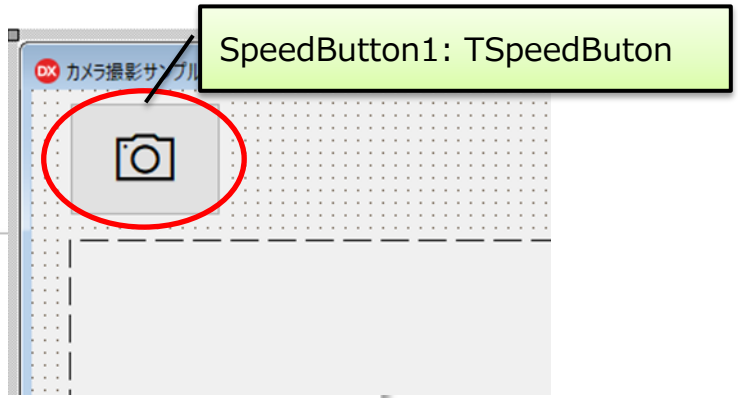
- Windows10には、「カメラ」アプリがある為、
  このアプリを呼び出す事でVCLアプリから
  カメラの活用が可能



Windows10 スタートメニュー

VCLを使用したタブレットアプリ構築術

- カメラの活用
  - カメラアプリの呼び出し
    - ShellExeuteを使用



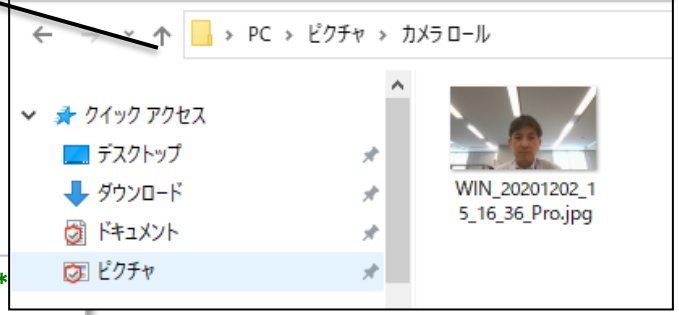

```
uses  Winapi.ShellAPI;


procedure TfrmCamera.SpeedButton1Click(Sender: TObject);
begin
  //カメラアプリを起動する
  ShellExecute(0, 'OPEN', PChar('microsoft.windows.camera:'), nil, nil, SW_SHOWMAXIMIZED);
end;
```

カメラを呼び出しても、撮影されたことがDelphiから把握できない。対処方法は？

■ VCLを使用したタブレットアプリ構築術

- カメラの活用
  - カメラロールを使用
    - カメラロールに新しいファイルが追加された事をチェックして、アプリに反映する。

カメラロール
C:¥Users¥[ユーザー名]¥Pictures¥Camera Roll¥



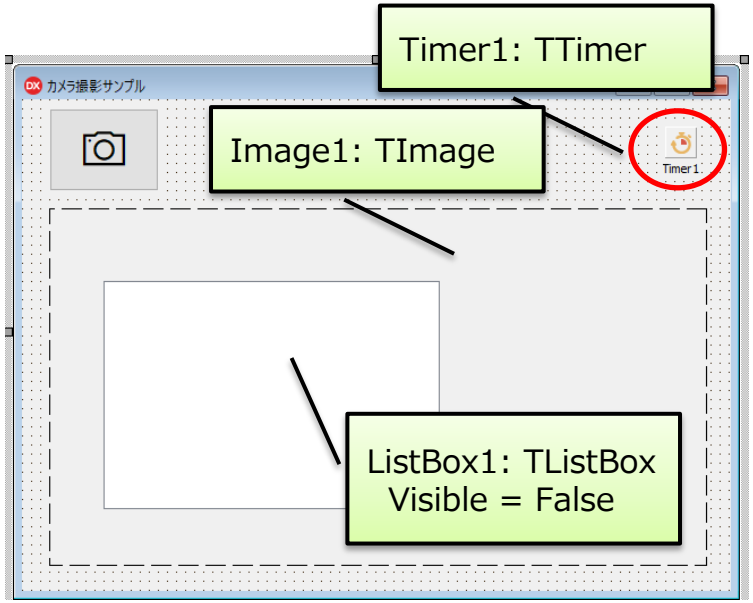
カメラロールのパスを取得　funcion GetCameraRollPath: String

```
{*****************************************************************************
 目的：カメラロールが設定されたパスを取得
 引数：なし
 戻値：取得されたパス
*****************************************************************************}
function TfrmCamera.GetCameraRollPath: String;
var
  pidl: PItemIDList;
  Path: array[0..MAX_PATH] of Char;
begin
  SHGetSpecialFolderLocation(0, CSIDL_MYPICTURES, pidl);
  SHGetPathFromIDList(pidl, Path);
  Result := Path + '¥Camera Roll';
end;
```

[ピクチャ]フォルダーのパスを取得

■ VCLを使用したタブレットアプリ構築術

- カメラの活用
  - 実装例




```
procedure TfrmCamera.SpeedButton1Click(Sender: TObject);
var
  sCameraRollPath: String;
  aFileNames: TStringDynArray;
  sFileName: String;
begin
  //カメラロールの取得
  sCameraRollPath := GetCameraRollPath;

  // 現時点のカメラロール内のファイル一覧をリストボックスにセット
  ListBox1.Items.Clear;
  aFileNames := TDirectory.GetFiles(sCameraRollPath);
  for sFileName in aFileNames do
  begin
    ListBox1.Items.Add(sFileName);
  end;

  //カメラアプリを起動する
  ShellExecute(0, 'OPEN', PChar('microsoft.windows.camera:'), nil,
     nil, SW_SHOWMAXIMIZED);
  Timer1.Enabled := True;
end;
```

カメラ起動時点のカメラロール内のファイル一覧を取得し、リストを保持

タイマーを起動し、カメラロールのチェックを行う

VCLを使用したタブレットアプリ構築術

- カメラの活用
  - 実装例

```
procedure TfrmCamera.Timer1Timer(Sender: TObject);
var
  sCameraRollPath: String;
  aFileNames: TStringDynArray;
  sFileName: string;
  CameraHWND: HWND;
begin
  sCameraRollPath := GetCameraRollPath; //カメラロールパスの取得
  //最新のカメラロールを確認
  aFileNames := TDirectory.GetFiles(sCameraRollPath);
  for sFileName in aFileNames do
  begin
    if ListBox1.Items.IndexOf(sFileName) = -1 then
    begin
      //新規ファイルの場合、フォーム上に画像を表示
      Image1.Picture.LoadFromFile(sFileName);
      //タイマー停止
      Timer1.Enabled := False;
      Sleep(500);
      //カメラアプリを終了する
      CameraHWND := FindWindow(nil, 'カメラ');
      if CameraHWND <> 0 then
        PostMessage(CameraHWND, WM_CLOSE, 0, 0);
    end;
  end;
end;
```

OnTimerイベントにて、
カメラロール内のファイルを一定間隔毎に
確認する

ListBoxに登録の無いファイルだった場合
新規に撮影されたファイルと判断

「カメラ」アプリのウィンドウを終了

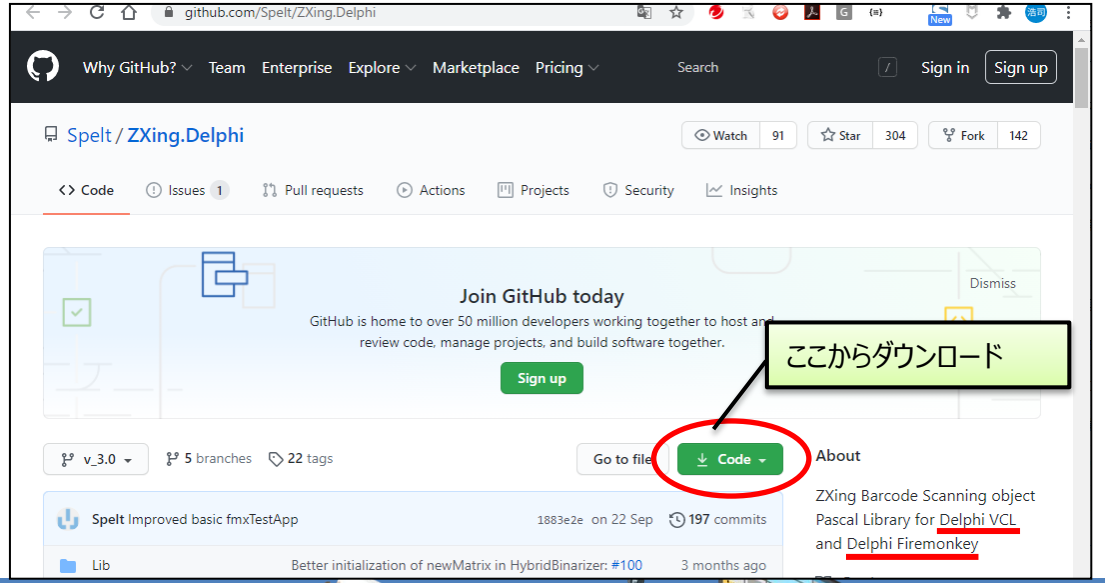
■ VCLを使用したタブレットアプリ構築術

- カメラの活用
  - 実行例

■ VCLを使用したタブレットアプリ構築術

- QRコードスキャン
  - VCL、FireMonkey共に、標準ではQRコードスキャンコンポーネントは無い
  - QRコードスキャンのオープンソースライブラリを活用することが可能
    - ZXing.Delphi

https://github.com/Spelt/ZXing.Delphi

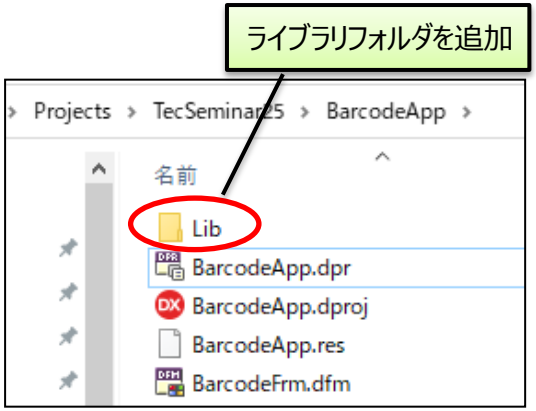


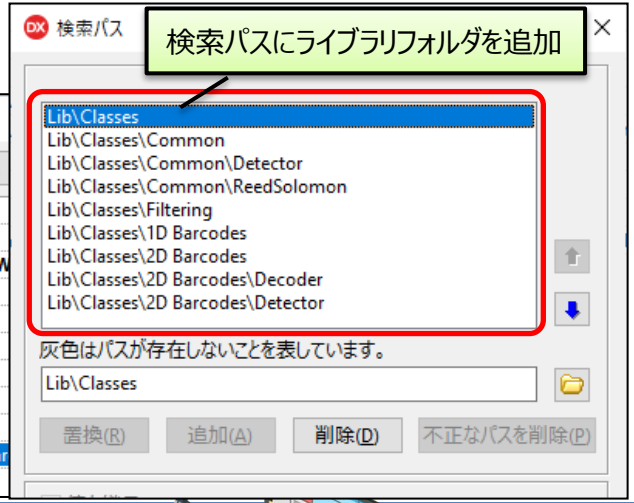
マルチデバイスで使用できる
QRコードスキャンライブラリ

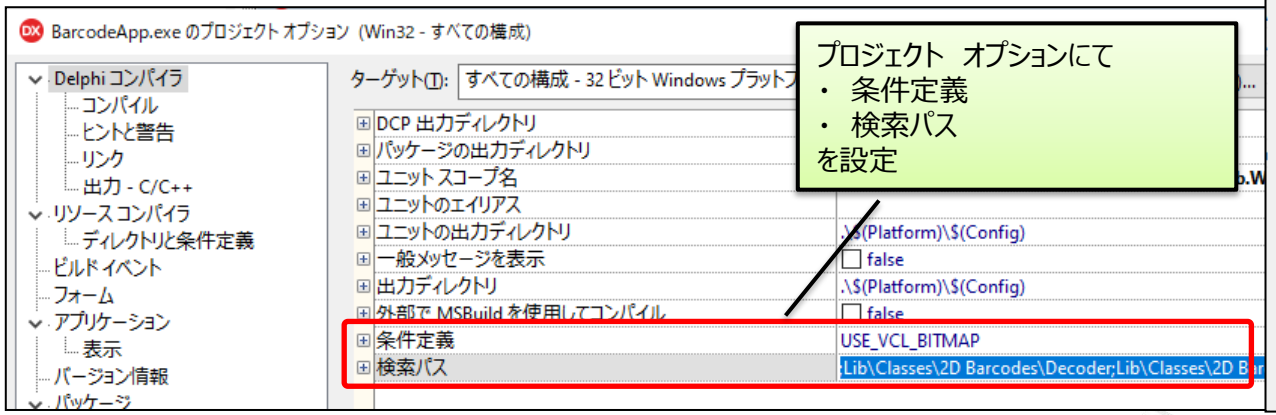
FireMonkeyだけでなく、VCLでも
使用可能

■ VCLを使用したタブレットアプリ構築術

- QRコードスキャン
  - ダウンロードしたライブラリの中にある[Lib]フォルダを
    プロジェクトフォルダの中に配置
  - プロジェクトのオプションにて下記を設定
    - 条件定義：USE_VCL_BITMAP (VCLの場合)
    - 検索パス：上記[Lib]フォルダ内の各フォルダをパスに追加

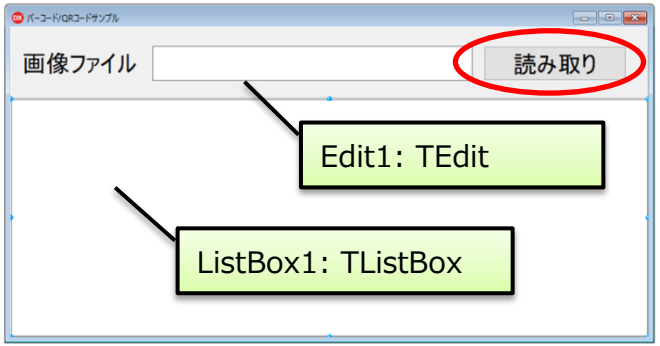






■ VCLを使用したタブレットアプリ構築術

- QRコードスキャン
  - 実装例



バーコードスキャン用ユニットを追加

```
//---- バーコードスキャン用ユニット
uses ZXing.ReadResult, ZXing.BarCodeFormat, ZXing.ScanManager;

procedure TfrmBarcode.Button1Click(Sender: TObject);
var
  bitmap: TBitmap;
  ReadResult: TReadResult;
  ScanManager: TScanManager;
  sFileName: String;
begin
  bitmap := TBitmap.Create;
  try
    //バーコードが撮影された画像を取得
    sFileName := Edit1.Text;
    bitmap.LoadFromFile(sFileName);
    // QRコードのスキャン
    try
      ScanManager := TScanManager.Create(TBarcodeFormat.QR_CODE, nil);
      ReadResult := ScanManager.Scan(bitmap);
      //コードがスキャンできた場合、取得値を画面にセット
      if ReadResult <> nil then
      begin
        ListBox1.Items.Add(ReadResult.text);
      end;
    finally
      FreeAndNil(ScanManager);
      FreeAndNil(ReadResult);
    end;
  finally
    bitmap.Free;
  end;
end;
```

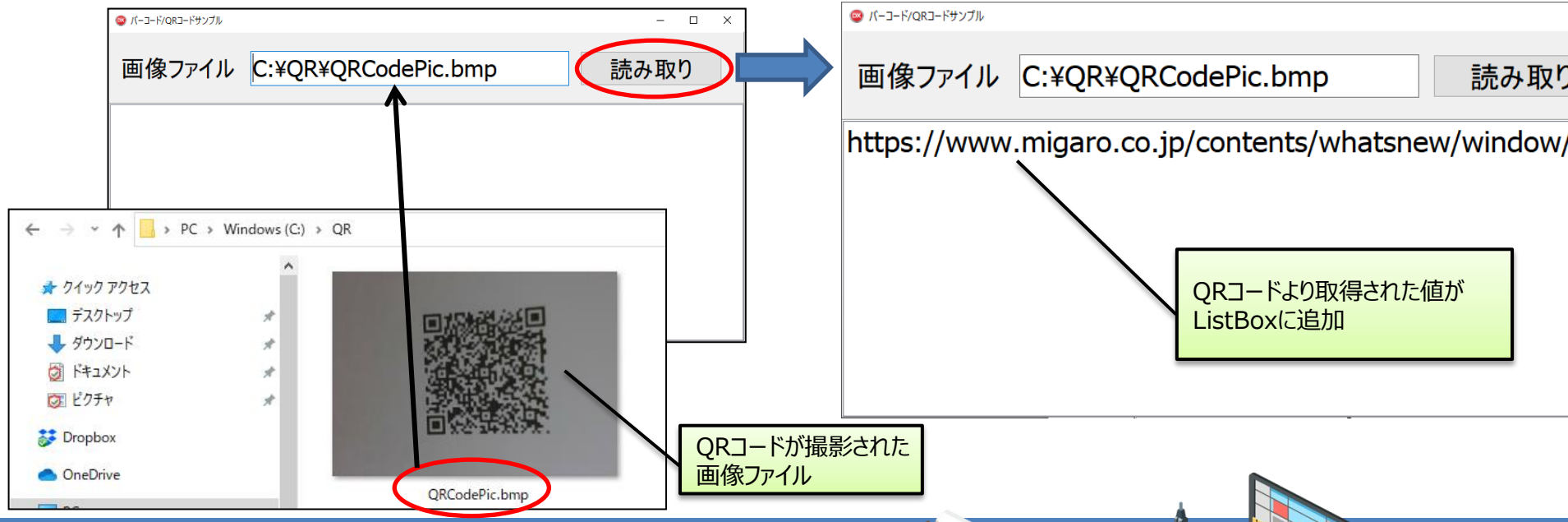
bitmap変数に画像ファイルを読み込む

画像ファイル上のQRコードをスキャン

スキャン成功の場合、取得値が
ReadResult.Textにセット

■ VCLを使用したタブレットアプリ構築術

- QRコードスキャン
  - 実行例

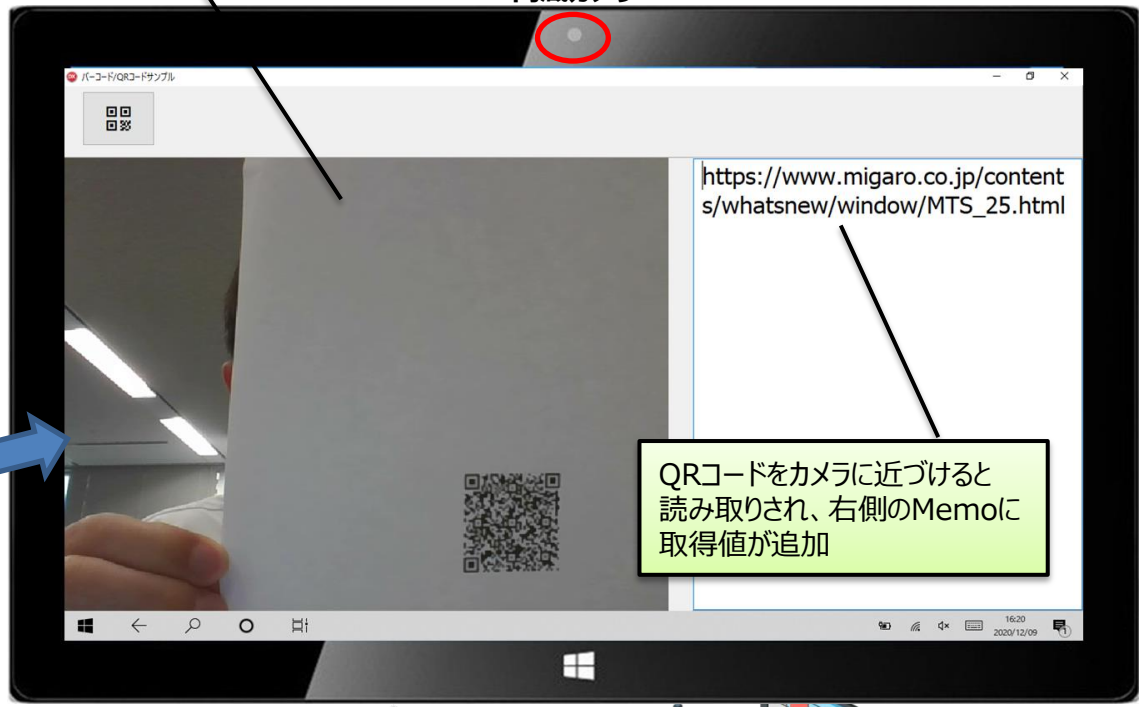


■ VCLを使用したタブレットアプリ構築術

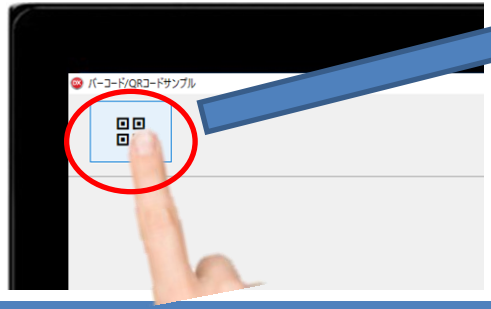
- QRコードスキャン
  - ライブラリには、PCカメラを連携してリアルタイムでスキャンするサンプルあり
    - WebCam.dpr

左側にカメラの動画が表示

内蔵カメラ



上記サンプルを元に作成したアプリ例



まとめ

■ まとめ

- VCLを使用したWindowsタブレットアプリ構築術
  - カスタムソフトキーボードの作成
  - 入力項目に最適化した入力欄の作成
  - カメラの活用
  - QRコードスキャン

ご清聴ありがとうございました。