第 21 回エンバカデロ・デベロッパーキャンプ
[W5]ワークショップ

FireMonkey アプリケーション構築実習

目次

1. 概要説明

FireMonkey とは

FireMonkey は、ビジュアルに優れたアプリケーションを Delphi や C++Builder で作成するための新しいアプリケーションプラットフォームです。FireMonkey ではベースとなる OS のネイティブグラフィックススタイルライブラリを使用し、HD/3D グラフィックス、柔軟性のあるスタイル、効果、アニメーション、新しいバインディングモデルなどが提供されています。

また、FireMonkey で構築されたアプリケーションは Windows/Mac OSX でのクロスコンパイルが可能です。
コードは CPU およびオペレーティングシステムにあわせてネイティブコンパイルされます。ランタイム、インタープリタ、JIT コンパイラのようなものはありません。
配布またはインストールするライブラリはありません。FireMonkey プラットフォームはコンパイルされて、実際のアプリケーション内に組み込まれるため、作成したアプリケーションを簡単に配布することができます。
DirectX や OpenGL のようなさまざまな GPU ライブラリに適合させることができます。

FireMonkey では、コンパイルされたネイティブアプリケーションにより、クロスプラットフォームかつオープンな環境で、CPU および GPU の最高のパフォーマンスを実現する一方、仮想マシン、インタープリタ、ランタイムなどをユーザーのコンピュータにインストールする必要はありません。

この「FireMonkey アプリケーション構築実習」では、具体的なアプリケーション構築の例題を通して、FireMonkey によるプログラミング手法を学びます。

2. 2D のピザアプリケーション

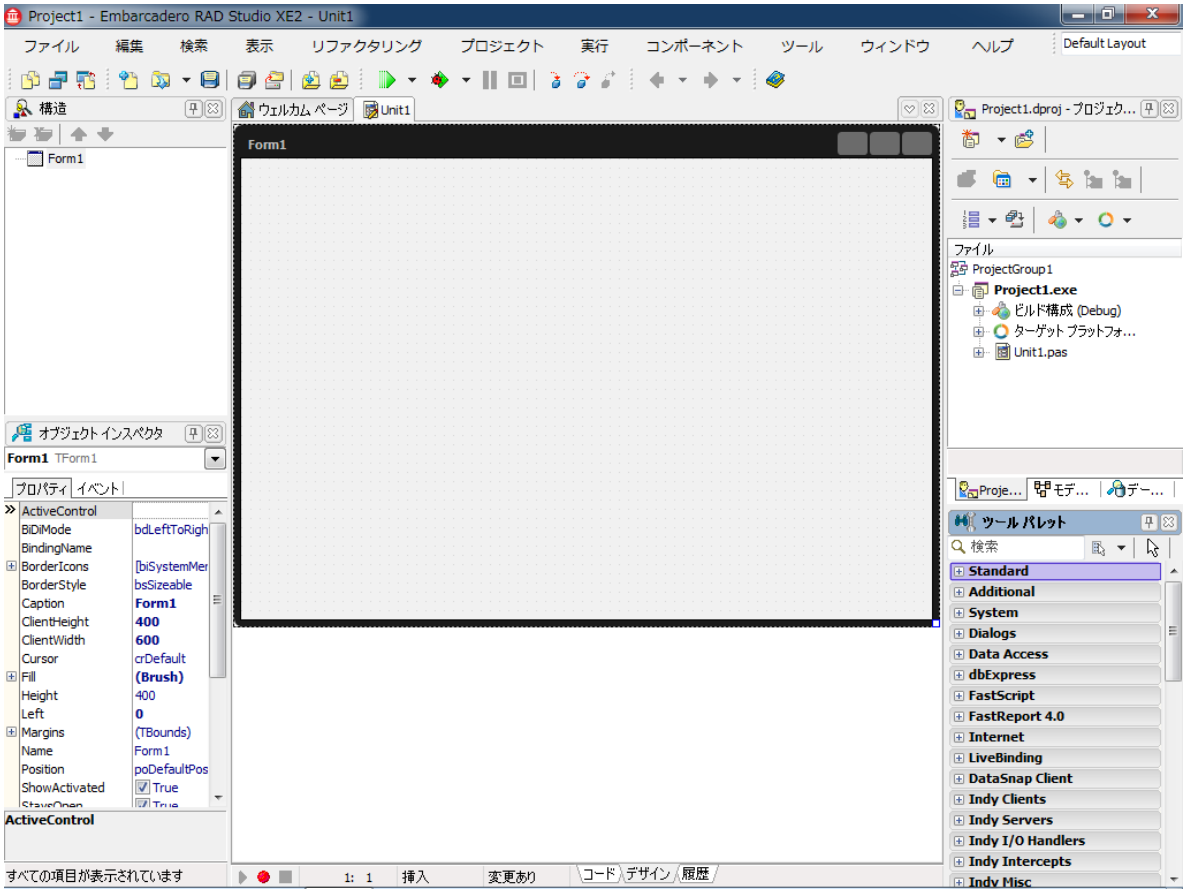
2-1. 最初のピザアプリケーション

FireMonkey HD アプリケーションの基本的な設計方法等について学びます。

この演習では、Button, Edit, Listbox を配置し、ボタンをクリックすると ListBox に Edit に入力した内容が追加されるという基本動作のコーディングまで行います。

演習 1.　2D のピザアプリケーションの作成

1. メニューの [ファイル | 新規作成 | FireMonkey HD アプリケーション – Delphi] 、または、[ファイル | 新規作成 | FireMonkey HD アプリケーション – C++Builder] を選択します。



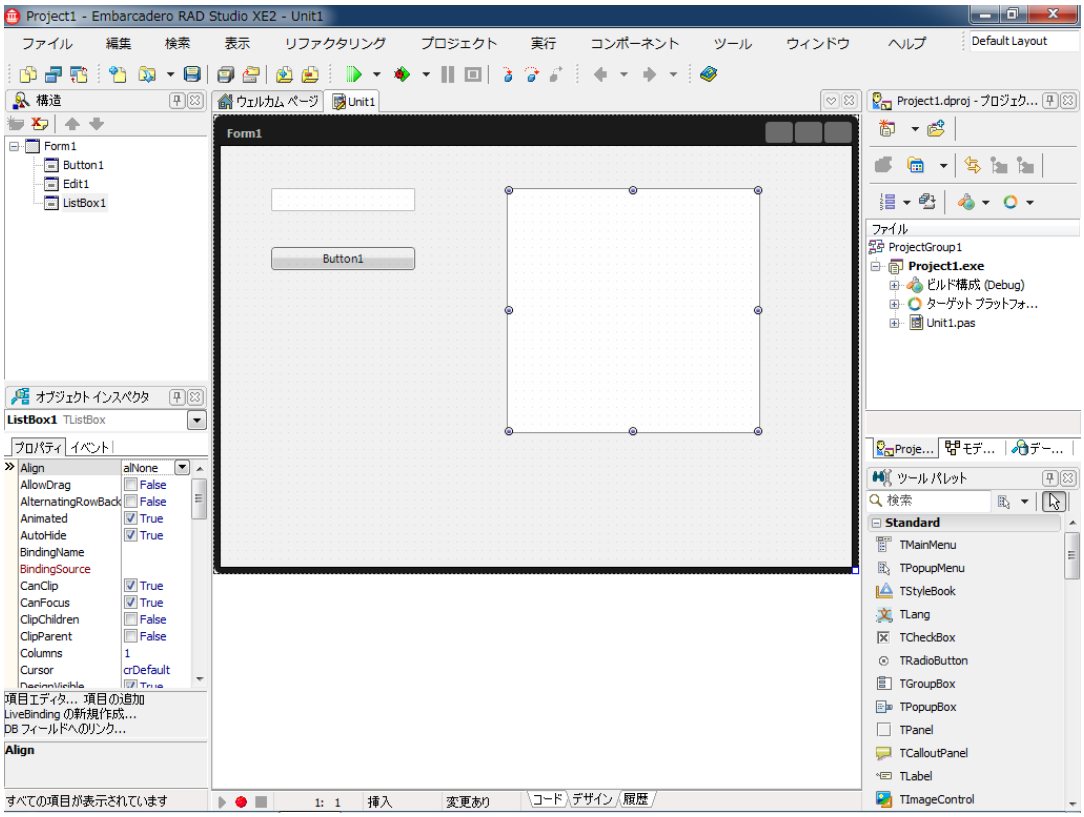
2. ツールパレットの Standard カテゴリから
   - TButton
   - TEdit
   - TListBox

   をフォーム上に配置します。

注意:

VCL の場合は、TPanel 等の限られたコンポーネントのみ親子関係にすることができましたが、FireMonkey 上では何でも親子関係にできます。そのため、ツールパレットからダブルクリックして配置すると、例えば TButton の中 (子として) に TEdit が配置されることがありますのでご注意ください。

3. それぞれのコンポーネントを、適当な大きさに調整します。



4. 各コンポーネントの Name プロパティの値を次のように設定します。
   - TForm：PizzaForm
   - TEdit：edTopping
   - TButton：btnAddTopping
   - TListBox：lbToppings

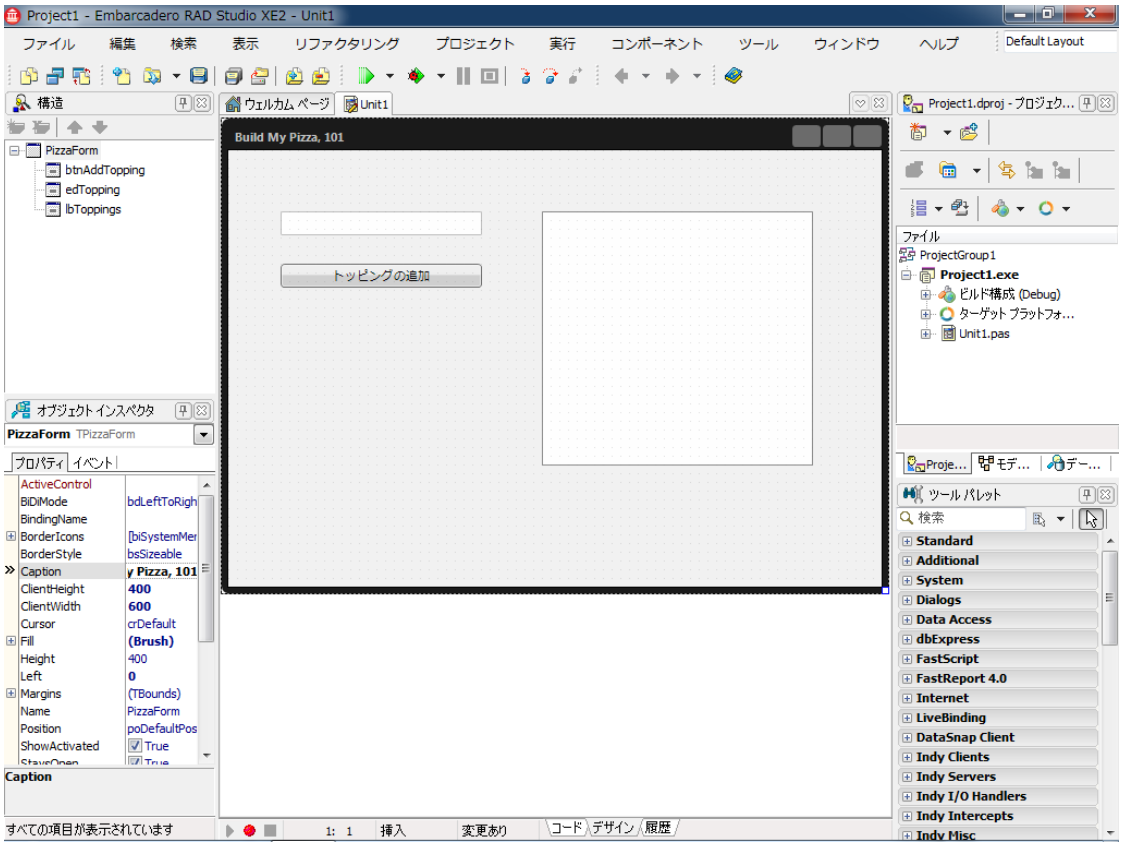
5. ここで、一旦プロジェクトを保存しておきましょう。[ファイル | すべて保存] とし、FireMonkey2D フォルダを作成し、保存します。

6. 次にフォームのキャプションと、ボタンの表示を修正します。
   - PizzaForm の Caption プロパティの内容を “ Build My Pizza, 101 “ に変更します。
   - btnAddTopping の Text プロパティの内容を “トッピングの追加” に変更します。

注意:

VCL フォームの場合、Button コンポーネントの表示は Caption プロパティですが、FireMonkey の Button コンポーネントの場合は Text プロパティとなります。

7. 変更した画面のイメージは以下のようになります。



8. btnAddPizza(Button) をマウスでダブルクリックし、以下のコードを記述します。

Delphi

```
lbToopings.Items.Add(edTopping.Text);
```

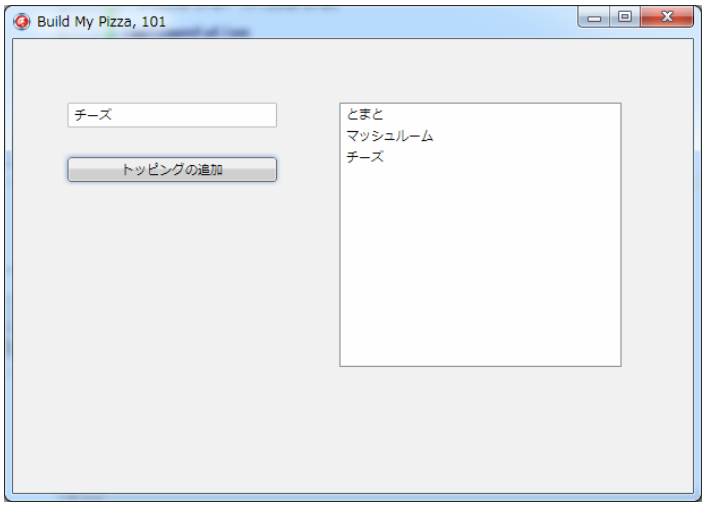
C++Builder

```
lbToppings->Items->Add(edTopping->Text);
```

9. [ファイル | すべて保存] でここまでの作業を保存します。

10. [実行 | 実行] で、作成したアプリケーションを実行します。

入力のボックスに何か入力し、[トッピングの追加]のボタンをクリックすると、右のリストに入力した内容が追加されます。



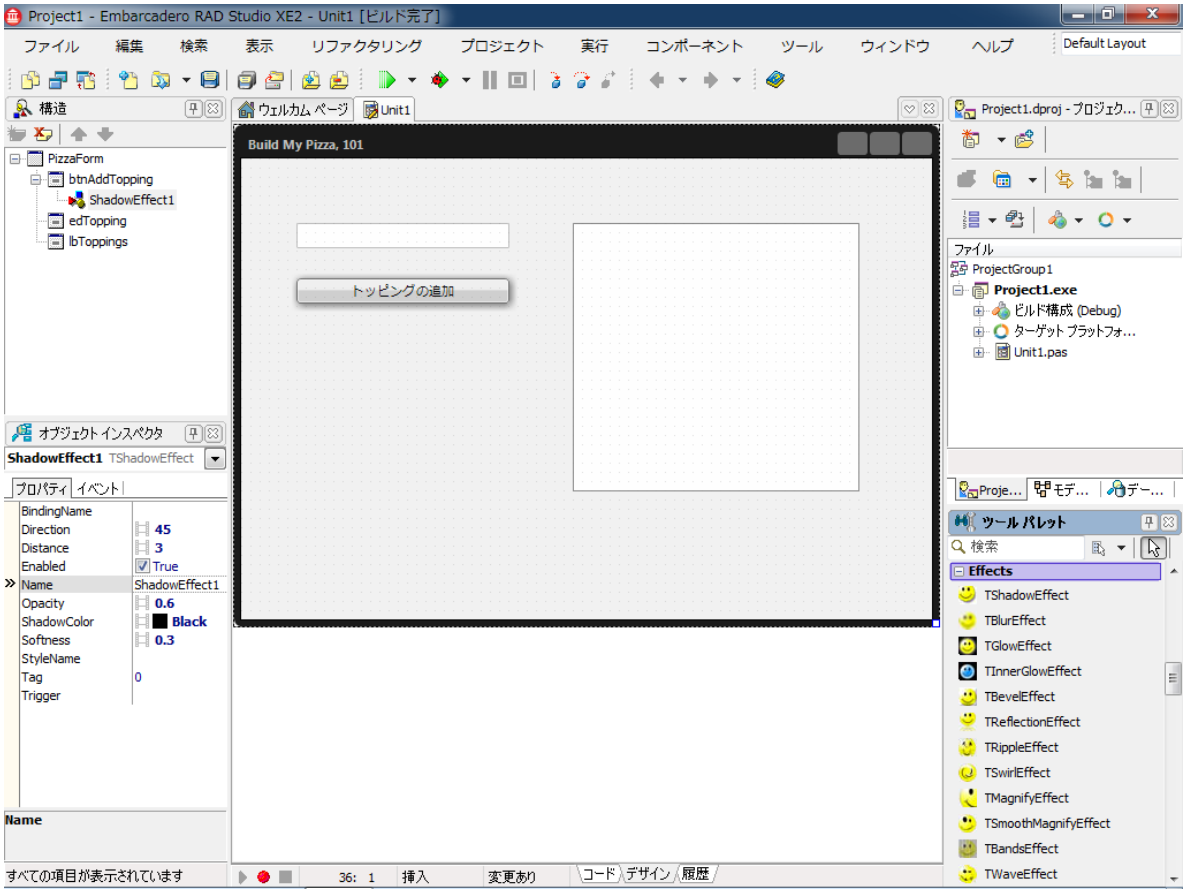
2-2. 効果とスタイルの適用

FireMonkey では、定義済みの表示効果を使用して、コントロールに簡単に表示効果を追加することができます。

演習 2. 効果の適用

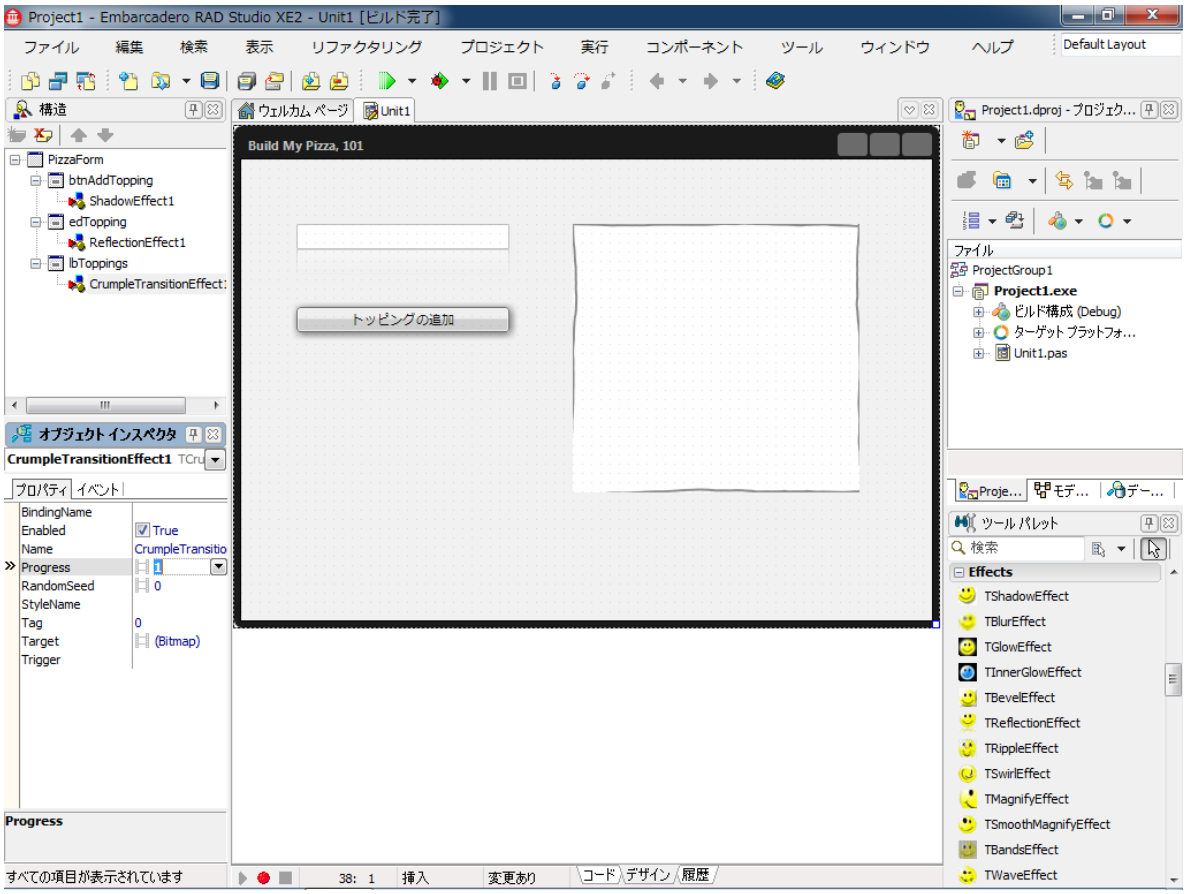
この演習では、演習 1 で作成したフォームに表示効果を追加します。

1. ツールパレットの Effects カテゴリ内の TShadowEffect をフォーム上に追加します。
2. 構造ペイン上で ShadowEffect をドラッグして、btnAddTooping にドロップし、以下の図のように親子関係になるように設定します。ボタンの周りの影が濃くなったことを確認できると思います。



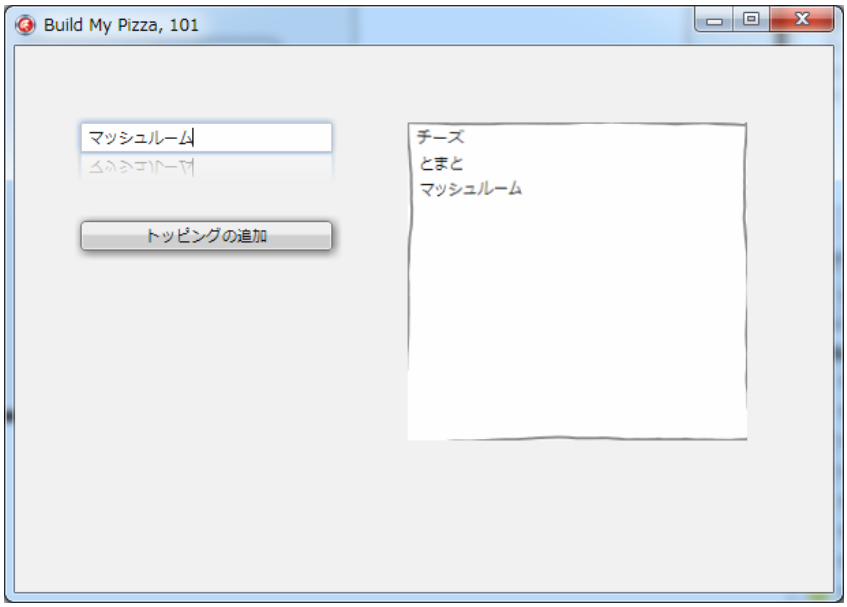
3. 次に TRefrectionEffect をフォームに追加し、edTopping にドラッグ＆ドロップで関連付けます。
4. RefrectionEffect の Length プロパティの値を 1 に設定します。これにより、edTopping コントロールの下に鏡のようにうっすらと表示が映るようになります。
5. TCrumpleTransitionEffect をフォームに追加し、lbToppings にドラッグ＆ドロップで関連付けます。
6. CrumpleTransitionEffect の Progress プロパティの値を 1 に設定します。これにより、ldToppings の外枠が軽くクシャっとした感じになります。

この３つの効果の追加で、フォームは以下のようになります。



7. メニューより、[ファイル | すべて保存] を選択し、ここまでの作業を保存します。
8. メニューの [実行 | 実行] を選択し、作成したアプリケーションを動作させます。

edTopping に値を入れると、その値がコントロールの下に鏡のように表示されることを確認できます。



lbToppings のクシャクシャ感を変化させるために、ここで TButton を１つ追加します。

9. ツールパレットからフォームに TButton を追加します。
10. Name プロパティを btnCrumple に修正します。
11. Text プロパティを“クシャクシャ”に設定します。

12. ツールパレットからフォームに TShadowEffect を追加し、btnCrumple にドラック＆ドロップします。
13. btnCrumple をダブルクリックし、以下のコードを追加します。

Delphi

```
CrumpleTransitionEffect1.Progress := CrumpleTransitionEffect1.Progress + 1;
```

C++Builder

```
CrumpleTransitionEffect1->Progress = CrumpleTransitionEffect1->Progress + 1;
```

14. フォームの大きさの変化と共にコントロールの大きさも変化するよう、ツールパレットの Layouts カテゴリから TScaledLayout コンポーネントをフォームに追加し、フォーム上に広げます。
15. 次に、構造ペイン上で、フォーム上のすべてのコンポーネントを ScaledLayout コンポーネントにドラック＆ドロップします。
16. TScaledLayout コンポーネントの Align プロパティを alClient に設定します。
17. オブジェクトインスペクタ上で、edTopping の「イベント」タブを選択し OnKeyDown と OnTypeing イベントの部分をクリックし、それぞれのイベント内に以下のコードを記述します。

Delphi

```
ReflectionEffect1.UpdateParentEffects;
```

C++Builder

```
ReflectionEffect1->UpdateParentEffects();
```

18. ツールパレットからフォーム上に TTimer を追加します。
19. オブジェクトインスペクタ上で、Timer1 の Interval プロパティの値を 500 に設定します。
20. オブジェクトインスペクタ上で、Timer1 の「イベント」タブを選択し、OnTimer イベントの部分をダブルクリックします。
21. OnTimer イベントが作成されますので、以下のコードを記述します。

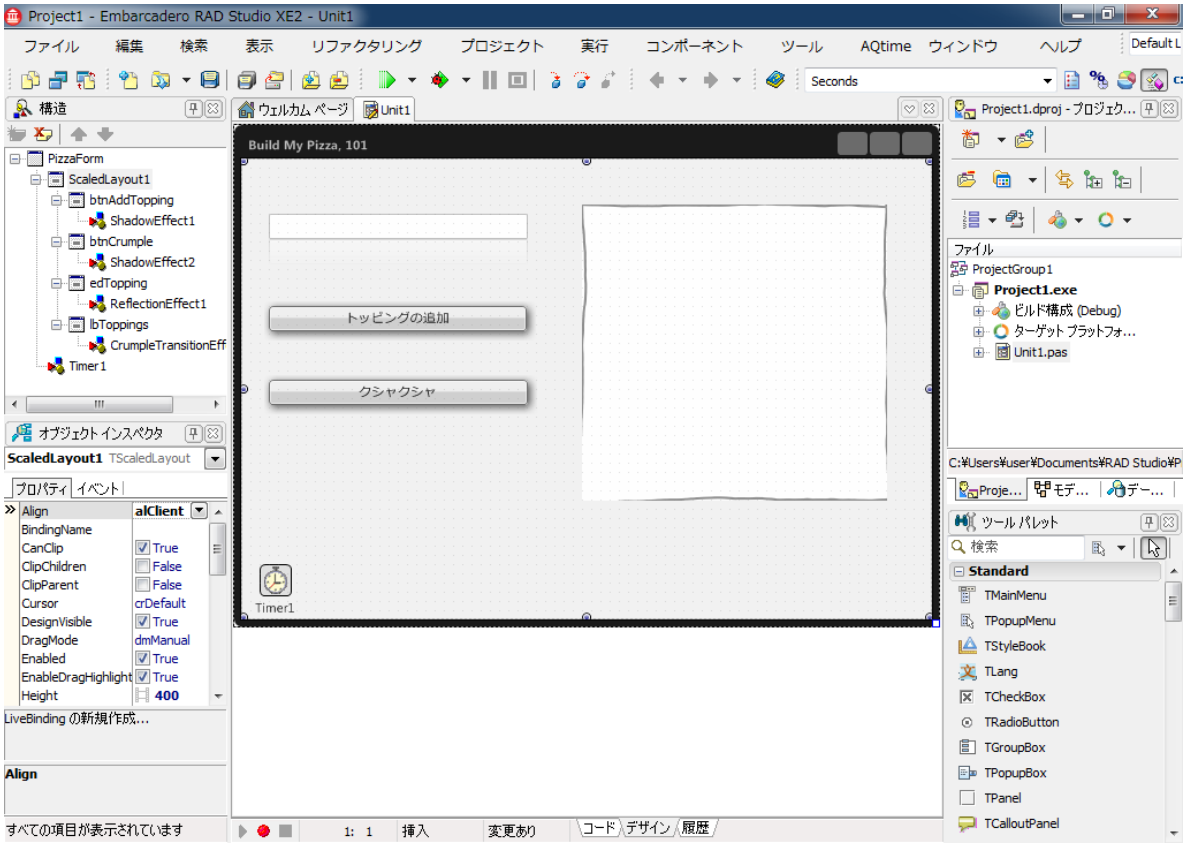
Delphi

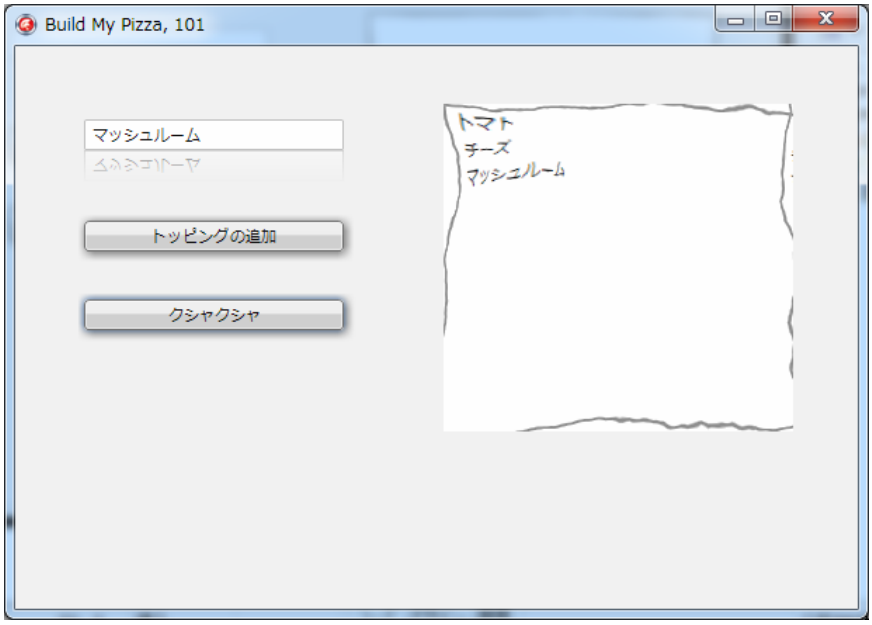
```
Salf.Invalidate;
```

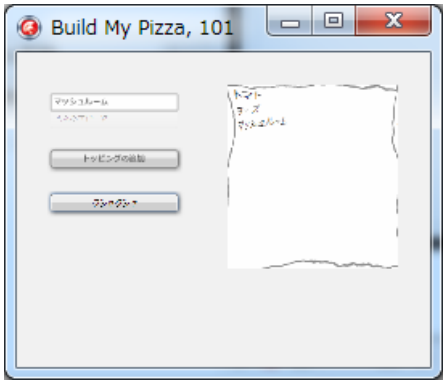
C++Builder

```
this->Invalidate();
```

22. メニューより、[ファイル | すべて保存]を選択し、ここまでの作業を保存します。
23. メニューより、[実行 | 実行]を選択し、作成したアプリケーションを動作させます。フォームの大きさを変更すると、その大きさと共に中のコンポーネントも縮小/拡大されます。また、[クシャクシャ]のボタンを押していくと lbToppings の外枠が変化していきます。


Project1 - Embarcadero RAD Studio XE2 - Unit1
ファイル 編集 検索 表示 リファクタリング プロジェクト 実行 コンポーネント ツール AQtime ウィンドウ ヘルプ
構造
PizzaForm
ScaledLayout1
btnAddTopping
ShadowEffect1
btnCrumple
ShadowEffect2
edTopping
ReflectionEffect1
lbToppings
CrumpleTransitionEff
Timer1
ウェルカム ページ
Unit1
Build My Pizza, 101
トッピングの追加
クシャクシャ
Timer1
オブジェクト インスペクタ
ScaledLayout1 TScaledLayout
プロパティ イベント
Align alClient
BindingName
CanClip True
ClipChildren False
ClipParent False
Cursor crDefault
DesignVisible True
DragMode dmManual
Enabled True
EnableDragHighlight True
Height 400
LiveBinding の新規作成...
Align
すべての項目が表示されています
1: 1 挿入 変更あり コード デザイン 履歴
Project1.dproj - プロジェク...
ファイル
ProjectGroup1
Project1.exe
ビルド構成 (Debug)
ターゲット プラットフォ...
Unit1.pas
C:¥Users¥user¥Documents¥RAD Studio¥P
ツール パレット
検索
Standard
TMainMenu
TPopupMenu
TStyleBook
TLang
TCheckBox
TRadioButton
TGroupBox
TPopupBox
TPanel
TCalloutPanel


Build My Pizza, 101
マッシュルーム
トッピングの追加
クシャクシャ
トマト
チーズ
マッシュルーム


Build My Pizza, 101

24. デザイナ上で、lbToppings を選択し、マウスの右ボタンをクリックして表示されたポップアップメニューから「項目の追加」を選択、またはオブジェクトインスペクタ上で「項目の追加」を選択し、ibToppings に TListBoxItem を追加します。
25. ツールパレットから TImegeControl をフォームに追加し、構造ペイン上で TListBoxItem にドラッグ＆ドロップで関連付けます。
26. 追加した TListBoxItem の大きさを調整します。
27. ボタンの画像を追加するため、オブジェクトインスペクタ上で、TImegeControl の Bitmap プロパティの横の [...] ボタンをクリックし、「編集...」を選択します。
28. ビットマップエディタが表示されますので [読み込み]のボタンをクリックして、ボタンの画像を選択し [開く] のボタンをクリックし、更に [OK]のボタンをクリックします。
29. また、TListBoxItem の Text プロパティの値を “トマト” に変更します。
30. オブジェクトインスペクタ上で、TImageControl を選択し、「イベント」タブをクリックします。表示されたイベントの「OnClick」の部分をダブルクリックして、OnClick イベントを生成し、以下のコードを記述します。

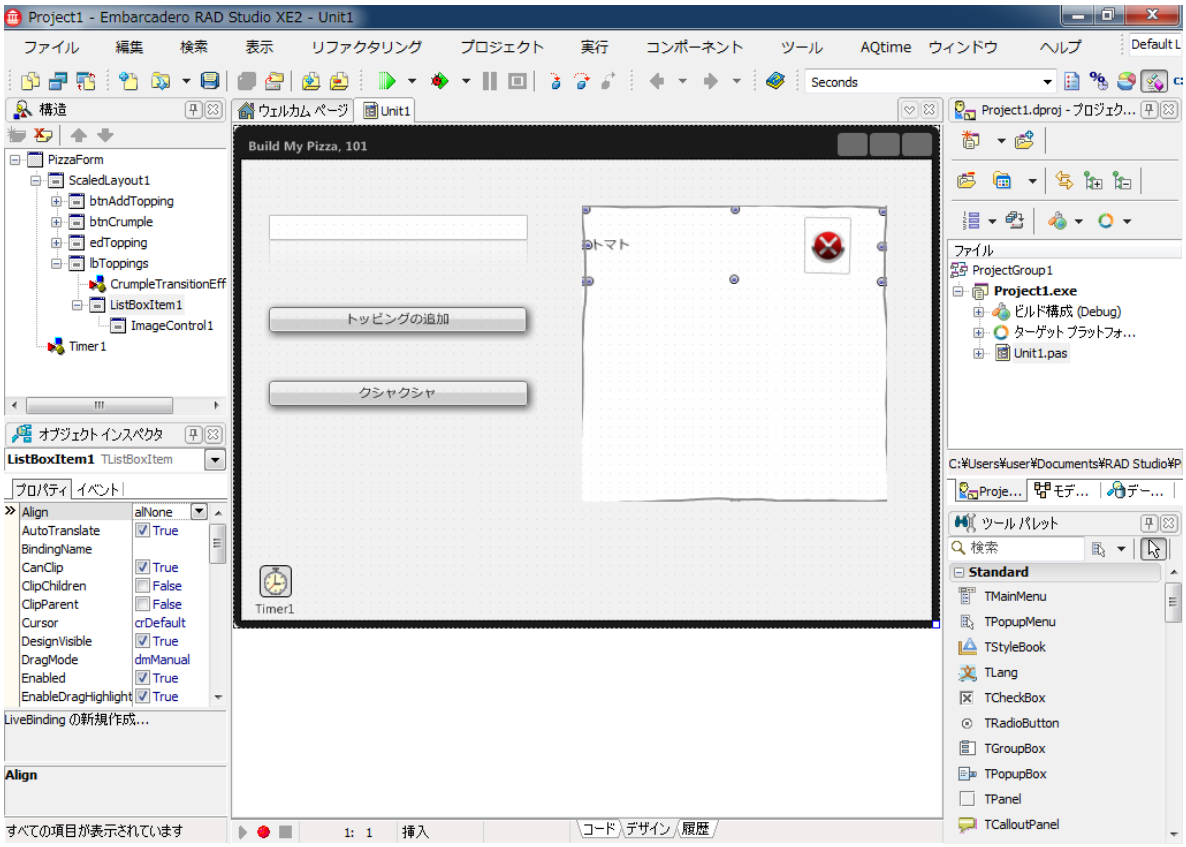
Delphi

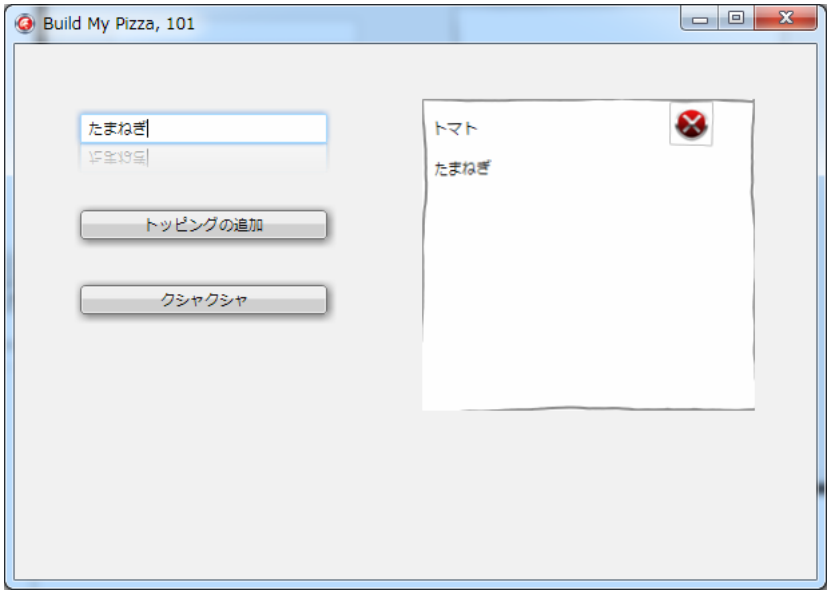
```
lbToppings.Items.Delete(0);
```

C++Builder

```
lbToppings->Items->Delete(0);
```

これによりあらかじめ ListBox 内に表示されている×ボタンをクリックすると、ListBox の項目が削除されるという動作が実装されます。




Build My Pizza, 101
たまねぎ
トッピングの追加
クシャクシャ
トマト
たまねぎ

スタイルの適用

FireMonkey では、どのビジュアルコンポーネントに対しても異なるスタイルで描画することができます。
また、スタイルに、ネストしたコントロールを追加することができるので、画像の付いたボタンなどの
組み合わせによるスタイルの作成も可能です。

演習 3. スタイルの適用

1. メニューの [ファイル | 新規作成 | FireMonkey HD アプリケーション – Delphi] 、または、[ファイル | 新規作成 | FireMonkey HD アプリケーション – C++Builder] を選択します。
2. ツールパレットの Standard カテゴリから
   - TButton
   - TListBox
   - TSizeGrip

   を、Additional カテゴリから
   - TComboEdit

   フォーム上に配置します。
3. ここで、一旦プロジェクトを保存しておきましょう。[ファイル | すべて保存] とし、¥FireMonkey2DStyle フォルダを作成し、このフォルダ以下に保存します。
4. 先程と同様に以下のコンポーネントの Name プロパティを変更します。
   - TButton → btnAddTopping
   - TListBox → lbToppings
   - TComboEdit → cbToppings

次にフォームのキャプションと、ボタンの表示を修正します。

5. PizzaForm の Caption プロパティの内容を “ Build My Pizza, 101 “ に変更します。
6. btnAddTopping の Text プロパティの内容を “トッピングの追加” に変更します。
7. TSizeGrip コンポーネントの Align プロパティを alScale に設定します。
8. 画像ファイルを¥FireMonkey2DStyle フォルダ下にコピーします。

cbTopping の項目を、Form 作成時に動的に作成します。
コピーした画像ファイルのファイル名を読み込み、cbToppings の Items に追加することによって、項目を作成しています。

9. uses に IOUtils を追加します。(C++Builder の場合は #include <IOUtils.hpp> を追加します)
10. Form の Create イベントに以下を記述します。

Delphi

```
var
  strArray: TStringDynArray;
  aStr: string;
begin
  strArray := TDirectory.GetFiles('.¥..¥..¥', '*.jpg');
  for aStr in strArray do
```

```
    cbToppings.Items.Add(TPath.GetFileNameWithoutExtension(aStr));
end;
```

C++Builder

```
UnicodeString aStr;
TStringDynArray strArray = TDirectory::GetFiles(".¥¥..¥¥..¥¥", "*.jpg");

for (int i = 0; i < strArray.Length; i++) {
  aStr = strArray[i];
  cbTopping->Items->Add(System::Ioutils::TPath::GetFileNameWithoutExtension(aStr));
}
```

11. btnAddTopping の OnClick イベントを作成し、先程と同様に選択した内容を lbToppings に追加する処理を記述します。

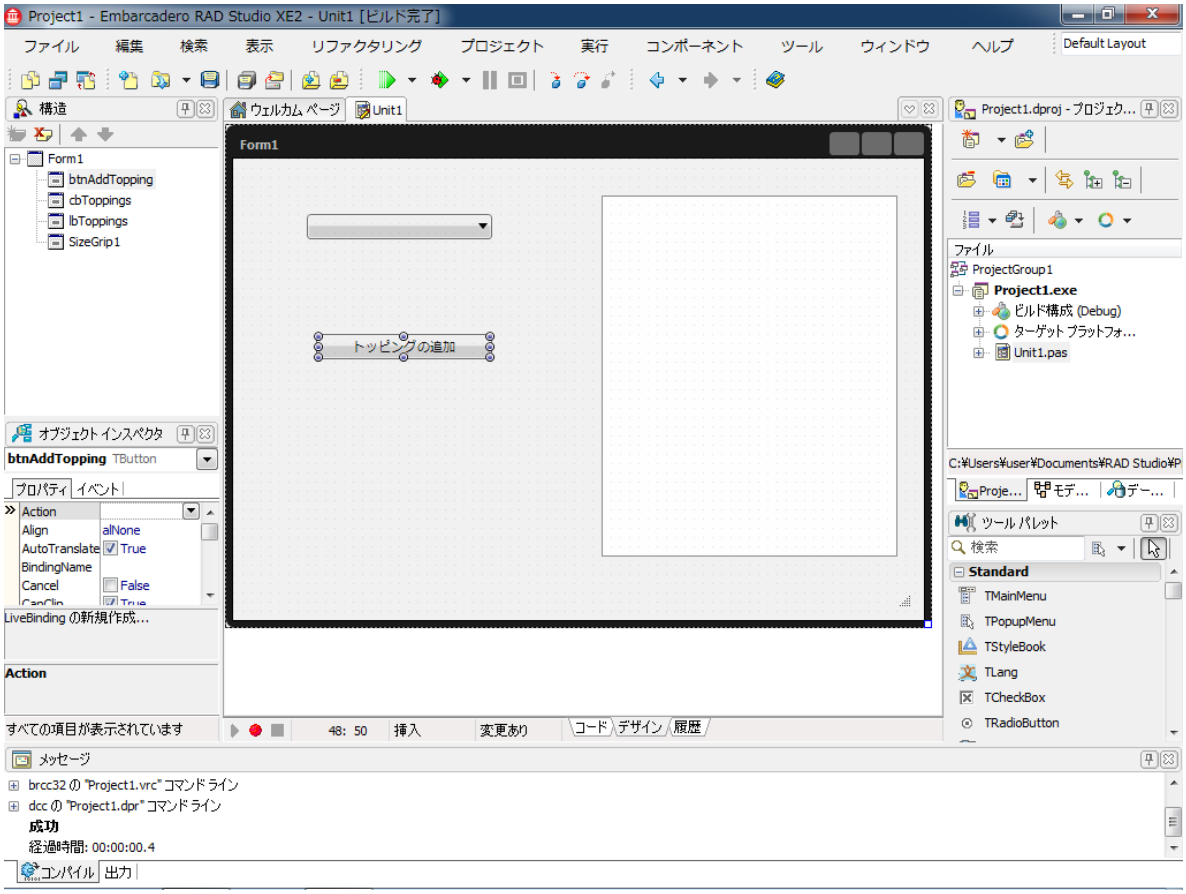
Delphi

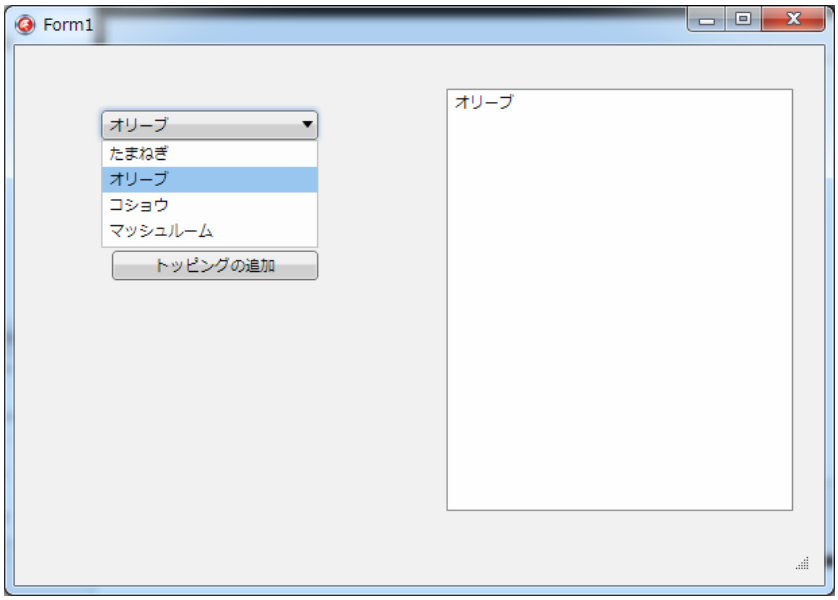
```
  lbToopings.Items.Add(cbToppings.Text);
```

C++Builder

```
  lbToppings->Items->Add(cbToppings->Text);
```

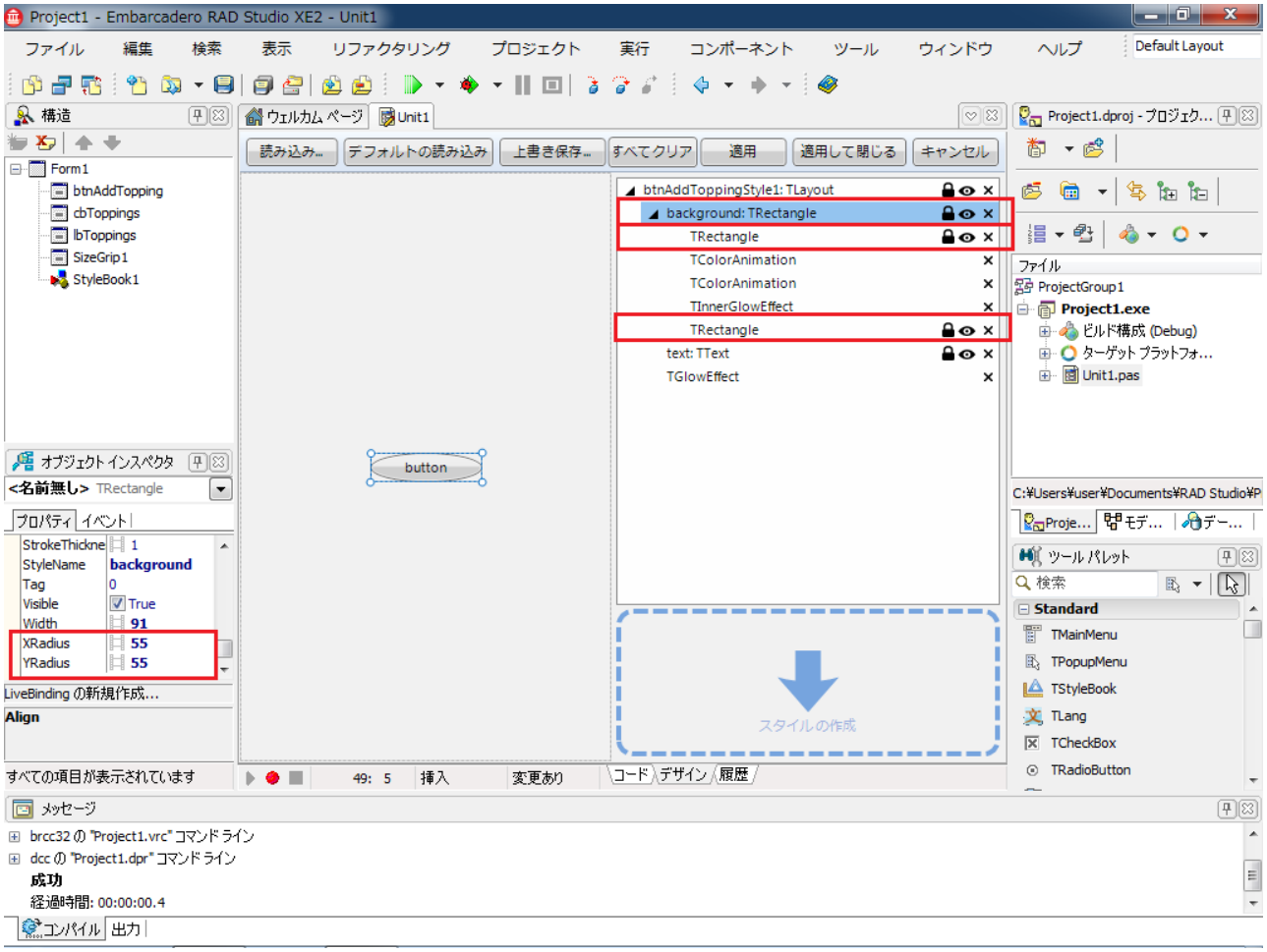
12. [ファイル | すべて保存] でここまでの内容を保存し、[実行|実行] で動作を確認します。

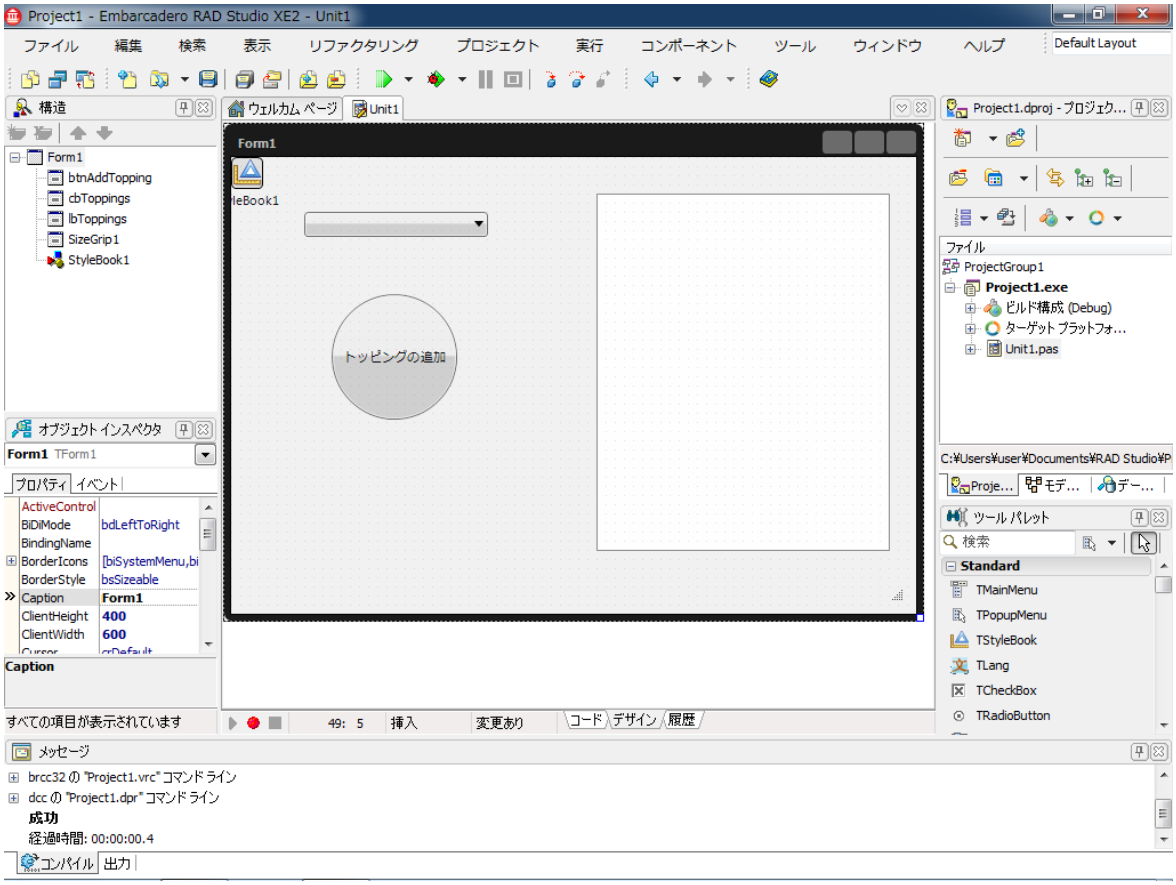




ボタンのスタイルを変更します。

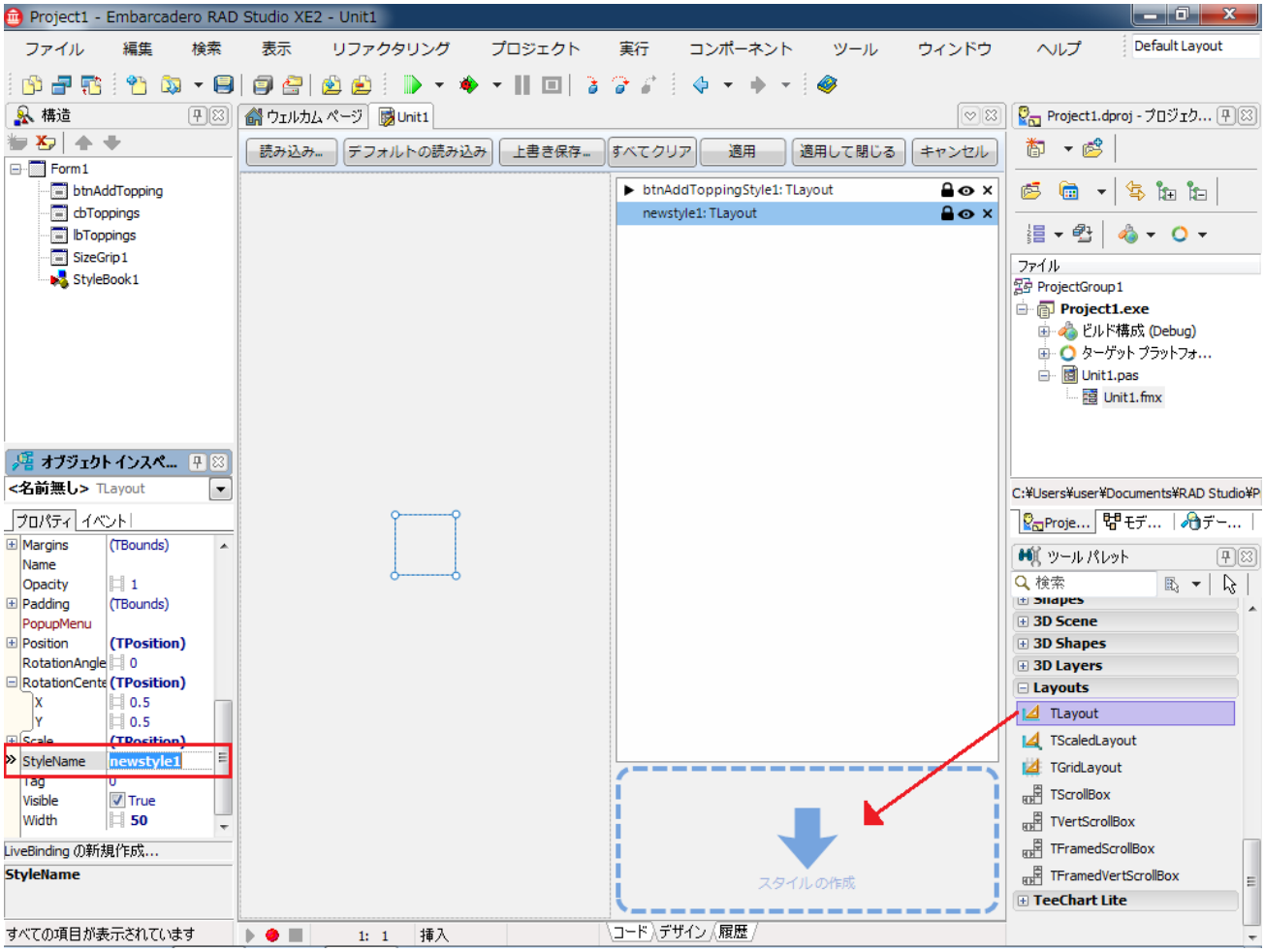
13. 丸いボタンに変更するため、btnAddTopping をデザイナ上で選択して、マウスの右ボタンをクリックします。
14. 表示されたポップアップメニューより「カスタムスタイルの編集」を選択し、スタイルエディタを開きます。
15. btnAddToppingStyle1 のツリーにある 3 つの TRectangle の XRadius プロパティと YRadius プロパティの値をすべて 55 に設定します。
16. [適用して閉じる]のボタンをクリックし、デザイン画面に戻り、btnAddTopping の Height プロパティと Width プロパティの値を 110 に設定します。これにより丸いボタンが表示されます。





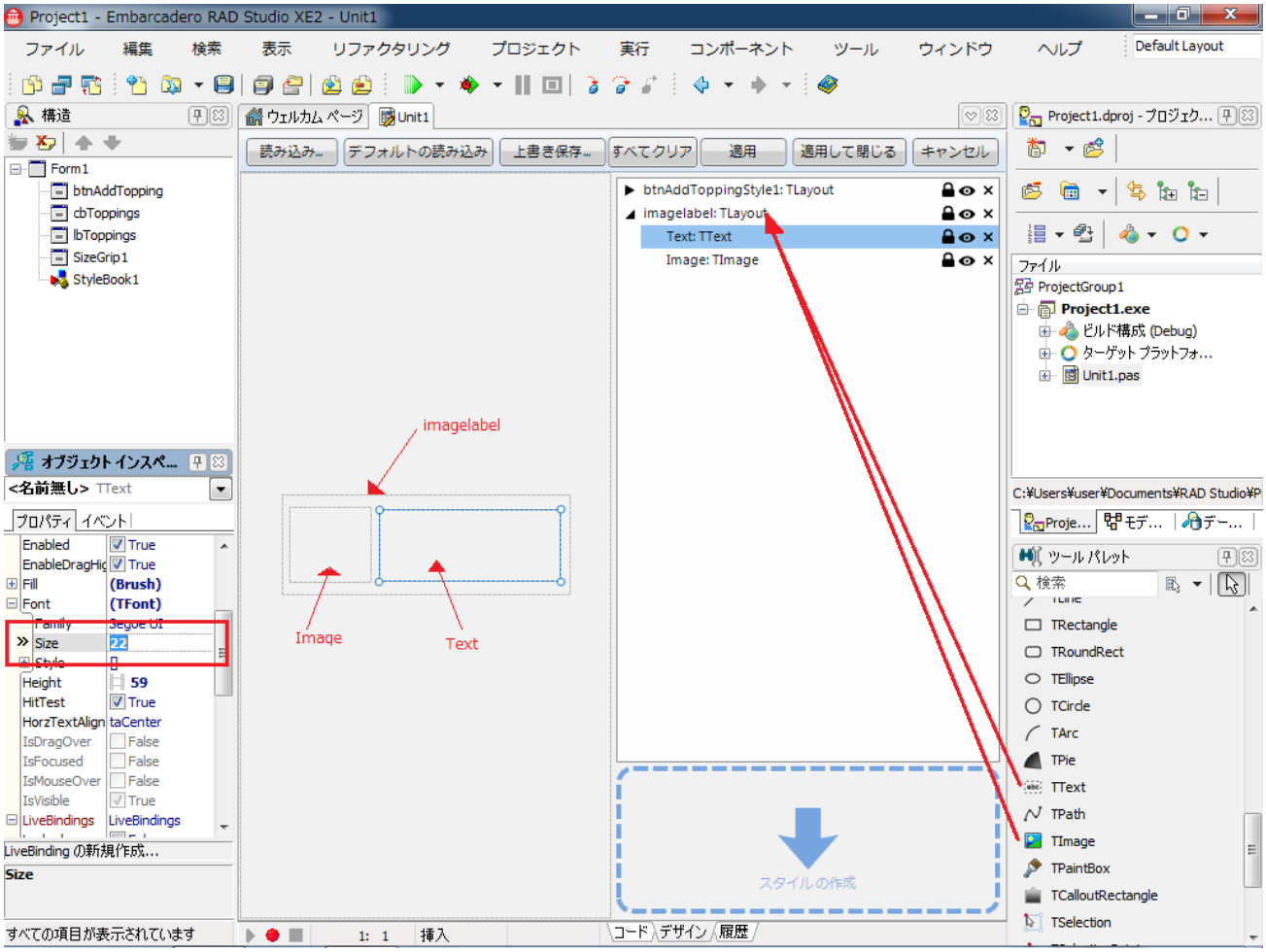
次に、ListBox に追加される項目に対してもスタイルを作成し、適用します。

17. まず btnAddTopping のスタイルの設定で自動的にフォームに追加された StyleBook1 コンポーネントをダブルクリックします。
18. ツールパレットから TLayout を、スタイルエディタの「スタイルの作成」の部分へドラッグ＆ドロップします。
19. 追加された新しいスタイルに対し、オブジェクトインスペクタ上の StyleName プロパティを imagelabel に変更します。



次に、このスタイルに、トッピング名を表示するための TText と、画像を表示する TImage を追加します。

20. まず、imagelabel スタイルのレイアウトの大きさを適度な大きさに変更します。
21. ツールパレットの Shapes カテゴリ内にある TText と TImage を imagelabel にドラッグ＆ドロップします。
22. 先程と同様に StyleName プロパティを Text1 から Text へ、 Image1 から Image に設定します。
23. TText の Font の Size プロパティを 11 から 22 に変更します。
24. それぞれのコンポーネントの大きさを図のように調整します。
25. [適用して閉じる] をクリックします。



次に、ボタンがクリックされた際に、ListBox に TListBoxItem を追加し、追加された ListBoxItem にこのスタイルを適用して、選択されたトッピングとその画像が追加されるコードを記述します。

26. 作業用に TImage をフォーム上に追加し、TImage の Visible プロパティを false に設定しておきます。

27. 同じく作業用のエリアを private 部分に宣言しておきます。

Delphi

```
private
  { private 宣言 }
  strTopping: string;  // 追加
```

C++Builder (.h ファイル側)

```
  private:          // ユーザー宣言
  UnicodeString strTopping;  // 追加
```

28. btnAddTopping をダブルクリックし、現在記述されているコードを以下のように書き換えます。

Delphi

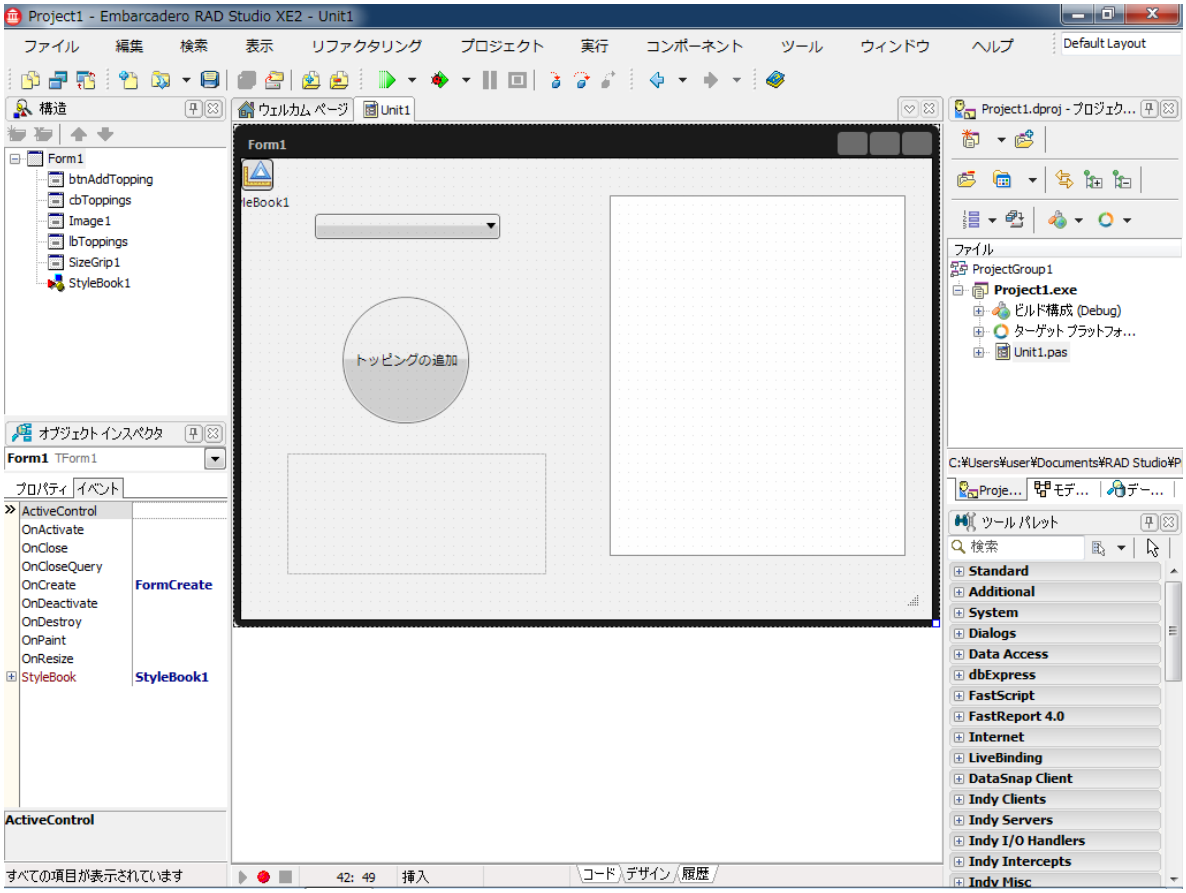
```
var
  listItem: TListBoxItem;
  itemText: TText;
  itemImage: TImage;
begin
  // lbToppings.Items.Add(cbToppings.Text);
```

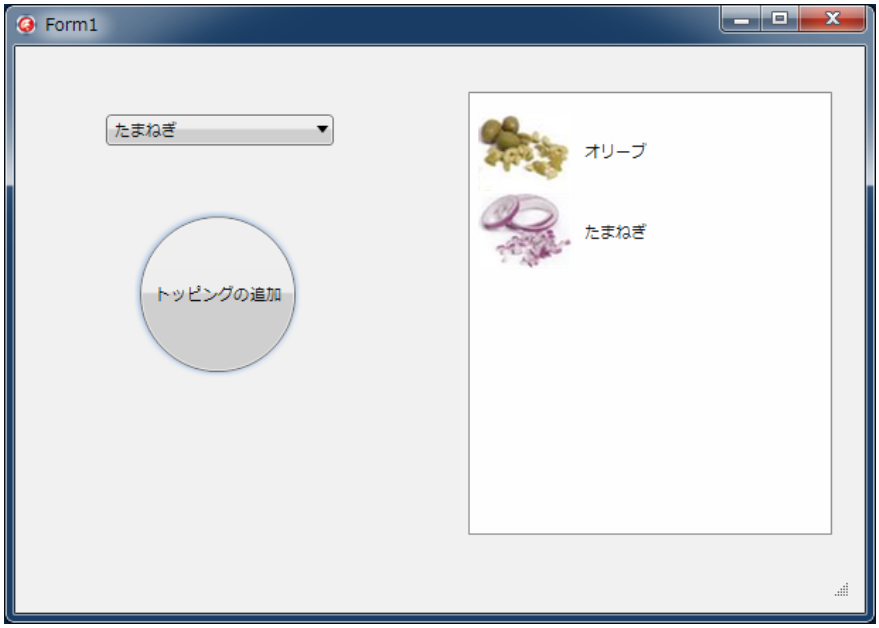
```
  if not Assigned (cbToppings.ListBox.Selected) then
  begin
    ShowMessage ('何も選択されて'+#13#10+'いません');
    Exit;
  end;

  strTopping := cbToppings.ListBox.Selected.Text;
  Caption := 'Adding' + strTopping;
  Image1.Bitmap.LoadFromFile('.¥..¥..¥'+ strTopping +    '.jpg');

  listItem := TListBoxItem.Create(lbToppings);
  listItem.Parent := lbToppings;
  listItem.StyleLookup := 'imagelabel';

  itemText := listItem.FindStyleResource('Text') as TText;
  if Assigned (itemText) then
    itemText.Text := strTopping ;

  itemImage := listItem.FindStyleResource('Image') as TImage;
  if Assigned (itemImage) then
  begin
    itemImage.Bitmap := Image1.bitmap;
    listItem.Height := Image1.bitmap.height;
  end
  else
    ShowMessage('バインドする'+#13#10+'イメージがありません');

  Image1.Bitmap.Clear(claWhite);
  Image1.Position.X := 32;
end;
```

C++Builder

```
//   lbToppings->Items->Add(cbToppings->Text);

  if (cbToppings->ListBox->Selected == NULL) {
        ShowMessage("何も選択されて¥r¥n いません");
  } else {

        strTopping = cbToppings->ListBox->Selected->Text;
        Caption = "Adding " + strTopping;
        Image1->Bitmap->LoadFromFile(".¥¥..¥¥..¥¥"+ strTopping +".jpg");

        TListBoxItem *listItem = new TListBoxItem(lbToppings);
```

```
        listItem->Parent = lbToppings;

        TText *itemText;
        TImage *itemImage;

        listItem->StyleLookup = "imagelabel";

        itemText = (TText*)(listItem->FindStyleResource("Text"));
        itemText->Text = strTopping;

        itemImage = (TImage*)(listItem->FindStyleResource("Image"));
        if (itemImage != NULL) {
          itemImage->Bitmap = Image1->Bitmap;
          listItem->Height = Image1->Bitmap->Height;
        } else {
          ShowMessage("バインドする¥r¥n イメージがありません");
        }

        Image1->Bitmap->Clear(claWhite);
        Image1->Position->X = 32;
}
```




Form1
たまねぎ
トッピングの追加
オリーブ
たまねぎ

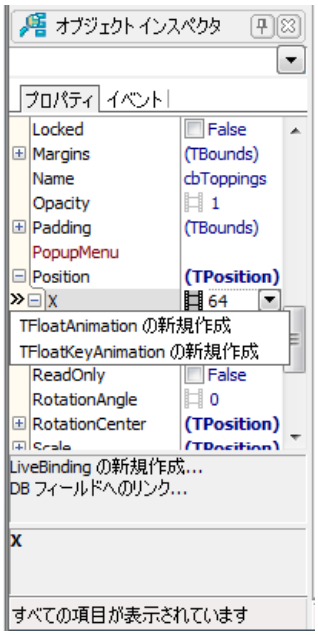
2-3. アニメーションの使用

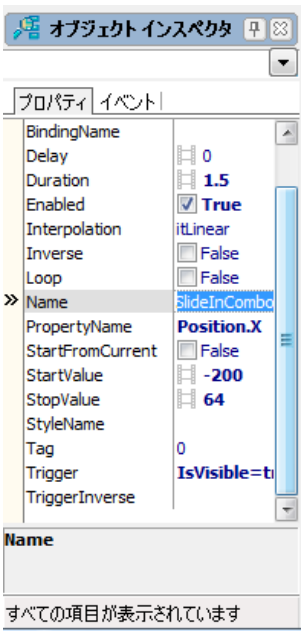
FireMonkey の特徴として「アニメーション化」があります。
これは、プロパティに時間的に変化する値を設定し動作させるというものです。
オブジェクトインスペクタから簡単に設定することもできますし、コードとして記述したものを実行させることもできます。

演習 4. アニメーションの使用

演習 3 で作成したフォームへアニメーションによる動作を追加します。
まず、フォームが表示された際に、cbToppings(ComboEdit) が横から現れるように、アニメーションの動作をつけましょう。

1. デザイン画面上で cbToppings を選択し、オブジェクトインスペクタで Position の X プロパティの部分にある▼を押し、「TFloatAnimation の新規作成」を選択します。





2. TFloatAnimation コンポーネントが作成されますので、それぞれのプロパティの値を以下のように設定します。

Duration	1.5
Enabled	True
Name	aniSlideInCombo
StartValue	-200
StopValue	32 (現在の cbToppings の Position.X の値に)
Trigger	IsVisible=True

3. これだけです。[ファイル|すべて保存] でいったん保存し、[実行|実行] で実行すると、左から cbToppings が現れます。

次に [トッピングの追加] ボタンを押した際に、画像が回転しながら現れ、lbToppings(ListBox) に追加される部分のアニメーションを設定します。

4. デザイン画面上で Image1 を選択し、オブジェクトインスペクタ上で、Visible プロパティの値を True に戻しま

す。

5. 同じく、デザイン画面上で Image1 を選択し、オブジェクトインスペクタ上で、Position プロパティの X の部分にある▼をクリックし 「TFloatAnimation の新規作成」を選択します。
6. TFloatAnimation が作成されますので、オブジェクトインスペクタ上で、プロパティの値を以下のように設定します。

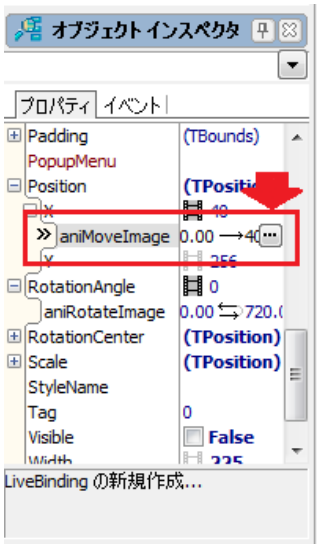
Duration	2
Name	aniMoveImage
StartFromCurrent	True
StopValue	400

7. もう一度デザイン画面に戻り、先程と同様に Image1 を選択し、オブジェクトインスペクタ上で、RotationAngle プロパティにある▼をクリックし「TFloatAnimation の新規作成」を選択します。
8. こちらも TFloatAnimation が作成されますので、オブジェクトインスペクタ上で、プロパティの値を以下のように設定します。

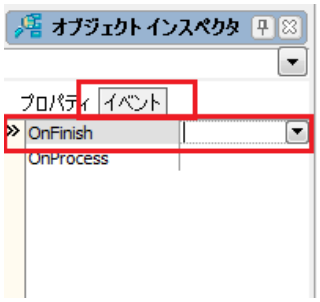
AutoReverse	True
Duration	1.5
Name	aniRotateImage
StartFromCurrent	True
StopValue	720

また、ボタンをクリックして上記のアニメーションが終了した時点で、lbToppings に追加されるように、以前の演習で記述したコードを別のイベントのコードとして記述します。

9. デザイン画面上で Image1 を選択し、オブジェクトインスペクタ上で、Position プロパティの X の部分に追加された aniMoveImage の所にある [...]ボタンをクリックします。



10. オブジェクトインスペクタ上で aniMoveImage 側に移動したら、「イベント」タブをクリックして OnFinish イベント部分をダブルクリックします。



11. aniMoveImageFinish イベントが自動的に作成されますので、btnAddTopping の OnClick イベントのコードの以下の部分をコピー&ペーストします。コピー元の方は {....} または /* ... */でくくってコメントにしておきます。

Delphi

```
var の宣言部分と Image1.Bitmap.LoadFromFile('.¥..¥..¥'+ strTopping +  '.jpg'); の次の文から最後まで
```

C++Builder

```
Image1->Bitmap->LoadFromFile(".¥¥..¥¥..¥¥"+ strTopping +".jpg"); の次の文から } の前まで (if の{}に気を付けてください)
```

12. 設定したアニメーションがボタンを押したときに動作するように、btnAddTopping の OnClick イベントを以下のように修正します。

Delphi

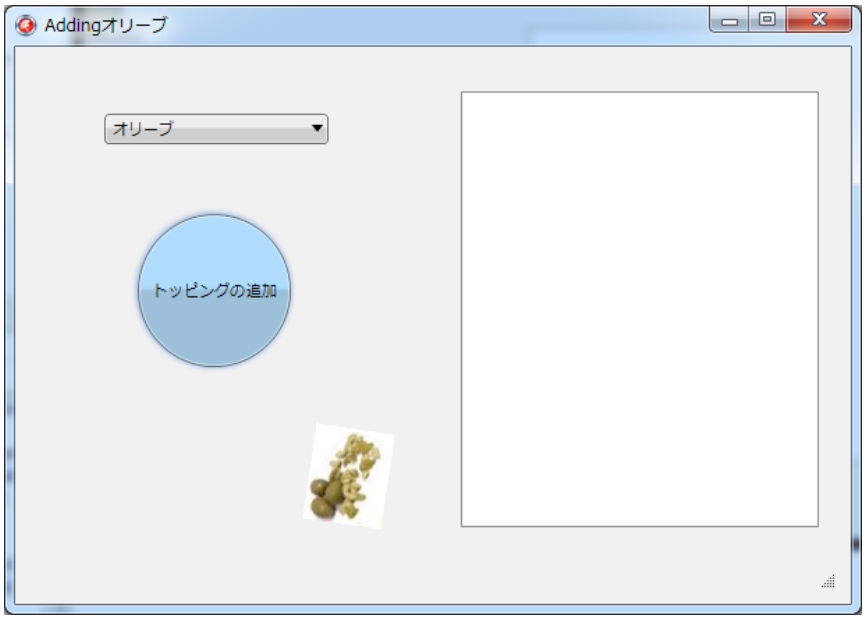
```
  procedure TForm1.btnAddToppingClick(Sender: TObject);
{
var
  listItem: TListBoxItem;
  itemText: TText;
  itemImage: TImage;
}
begin
  // lbToppings.Items.Add(cbToppings.Text);
  if not Assigned (cbToppings.ListBox.Selected) then
  begin
    ShowMessage ('何も選択されて'+#13#10+'いません');
    Exit;
  end;

  strTopping := cbToppings.ListBox.Selected.Text;
  Caption := 'Adding' + strTopping;
  Image1.Bitmap.LoadFromFile('.¥..¥..¥'+ strTopping +  '.jpg');

  aniRotateImage.Start;
  aniMoveImage.StopValue := lbToppings.Position.X + 20;
  aniMoveImage.Start;
```

```
{ 以下コメントアウト
  listItem := TListBoxItem.Create(lbToppings);
  listItem.Parent := lbToppings;
  listItem.StyleLookup := 'imagelabel';
```

C++Builder

```
//   ListBox1->Items->Add(ComboEdit1->Text);
  if (cbToppings->ListBox->Selected == NULL) {
        ShowMessage("何も選択されて¥n いません");
  } else {

        strTopping = cbToppings->ListBox->Selected->Text;
        Caption = "Adding " + strTopping;
        Image1->Bitmap->LoadFromFile(".¥¥..¥¥..¥¥"+ strTopping +".jpg");

        aniRotateImage->Start();
        aniMoveImage->StopValue = lbToppings->Position->X + 20;
        aniMoveImage->Start();
  }

/* 以下コメントアウト 最後の }はコメントの直前に移動
        TListBoxItem *listItem = new TListBoxItem(lbToppings);
        listItem->Parent = lbToppings;
```

13. これで完成です。[ファイル|すべて保存]して、[実行|実行]でアプリケーションを動作させてみましょう。



コード部分はそれぞれ以下のようになります（コメントにした部分は省いています）

Delphi

```
unit Unit1;
```

```
interface

uses
  System.SysUtils, System.Types, System.UITypes, System.Classes, System.Variants,
  FMX.Types, FMX.Controls, FMX.Forms, FMX.Dialogs, FMX.Layouts, FMX.ListBox,
  FMX.Edit, FMX.Objects, FMX.Ani;

type
  TForm1 = class(TForm)
    cbToppings: TComboEdit;
    btnAddTopping: TButton;
    lbToppings: TListBox;
    SizeGrip1: TSizeGrip;
    StyleBook1: TStyleBook;
    Image1: TImage;
    aniSlideInCombo: TFloatAnimation;
    aniMoveImage: TFloatAnimation;
    aniRotateImage: TFloatAnimation;
    procedure FormCreate(Sender: TObject);
    procedure btnAddToppingClick(Sender: TObject);
    procedure aniMoveImageFinish(Sender: TObject);
  private
    { private 宣言 }
    strTopping: string;
  public
    { public 宣言 }
  end;

var
  Form1: TForm1;

implementation
uses IOUtils;

{$R *.fmx}

procedure TForm1.aniMoveImageFinish(Sender: TObject);
var
  listItem: TListBoxItem;
  itemText: TText;
  itemImage: TImage;
```

```
begin
   listItem := TListBoxItem.Create(lbToppins);
   listItem.Parent := lbToppings;
   listItem.StyleLookup := 'imagelabel';

   itemText := listItem.FindStyleResource('Text') as TText;
   if Assigned (itemText) then
      itemText.Text := strTopping ;

   itemImage := listItem.FindStyleResource('Image') as TImage;
   if Assigned (itemImage) then
   begin
      itemImage.Bitmap := Image1.bitmap;
      listItem.Height := Image1.bitmap.height;
   end
   else
      ShowMessage('バインドする'+#13#10+'イメージがありません');

   Image1.Bitmap.Clear(claWhite);
   Image1.Position.X := 32;
end;

procedure TForm1.btnAddToppingClick(Sender: TObject);
begin
   if not Assigned (cbToppings.ListBox.Selected) then
   begin
      ShowMessage ('何も選択されて'+#13#10+'いません');
      Exit;
   end;

   strTopping := cbToppings.ListBox.Selected.Text;
   Caption := 'Adding' + strTopping;
   Image1.Bitmap.LoadFromFile('.¥..¥..¥'+ strTopping +   '.jpg');

   aniRotateImage.Start;
   aniMoveImage.StopValue := lbToppings.Position.X + 20;
   aniMoveImage.Start;

end;

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

```
  strArray: TStringDynArray;
  aStr: string;
begin
  strArray := TDirectory.GetFiles('.¥..¥..¥', '*.jpg');
  for aStr in strArray do
    cbToppings.Items.Add(TPath.GetFileNameWithoutExtension(aStr));
end;

end.
```

C++Builder

.h 側

```
//---------------------------------------------------------------------------

#ifndef bcbUnit1H
#define bcbUnit1H
//---------------------------------------------------------------------------
#include <System.Classes.hpp>
#include <FMX.Controls.hpp>
#include <FMX.Forms.hpp>
#include <FMX.Edit.hpp>
#include <FMX.Types.hpp>
#include <FMX.Layouts.hpp>
#include <FMX.ListBox.hpp>
#include <FMX.Objects.hpp>
#include <FMX.Ani.hpp>
//---------------------------------------------------------------------------
class TForm1 : public TForm
{
__published:        // IDE で管理されるコンポーネント
        TComboEdit *cbToppings;
        TButton *btnAddTopping;
        TListBox *lbToppings;
        TImage *Image1;
        TStyleBook *StyleBook1;
        TFloatAnimation *aniSlideInCombo;
        TFloatAnimation *aniMoveImage;
        TFloatAnimation *aniRotateImage;
        void __fastcall btnAddToppingClick(TObject *Sender);
        void __fastcall FormCreate(TObject *Sender);
        void __fastcall aniMoveImageFinish(TObject *Sender);
private:  // ユーザー宣言
```

```
    UnicodeString strTopping;
public:                 // ユーザー宣言
        __fastcall TForm1(TComponent* Owner);
};
//---------------------------------------------------------------------------
extern PACKAGE TForm1 *Form1;
//---------------------------------------------------------------------------
#endif
```

.cpp 側

```
//---------------------------------------------------------------------------

#include <fmx.h>
#include <IOUtils.hpp>
#pragma hdrstop

#include "bcbUnit1.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.fmx"
TForm1 *Form1;
//---------------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------------------
void __fastcall TForm1::btnAddToppingClick(TObject *Sender)
{
  if (cbToppings->ListBox->Selected == NULL) {
        ShowMessage("何も選択されて¥r¥n いません");
  } else {

        strTopping = cbToppings->ListBox->Selected->Text;
        Caption = "Adding " + strTopping;
        Image1->Bitmap->LoadFromFile(".¥¥..¥¥..¥¥"+ strTopping +".jpg");

        aniRotateImage->Start();
        aniMoveImage->StopValue = lbToppings->Position->X + 20;
        aniMoveImage->Start();
  }
```

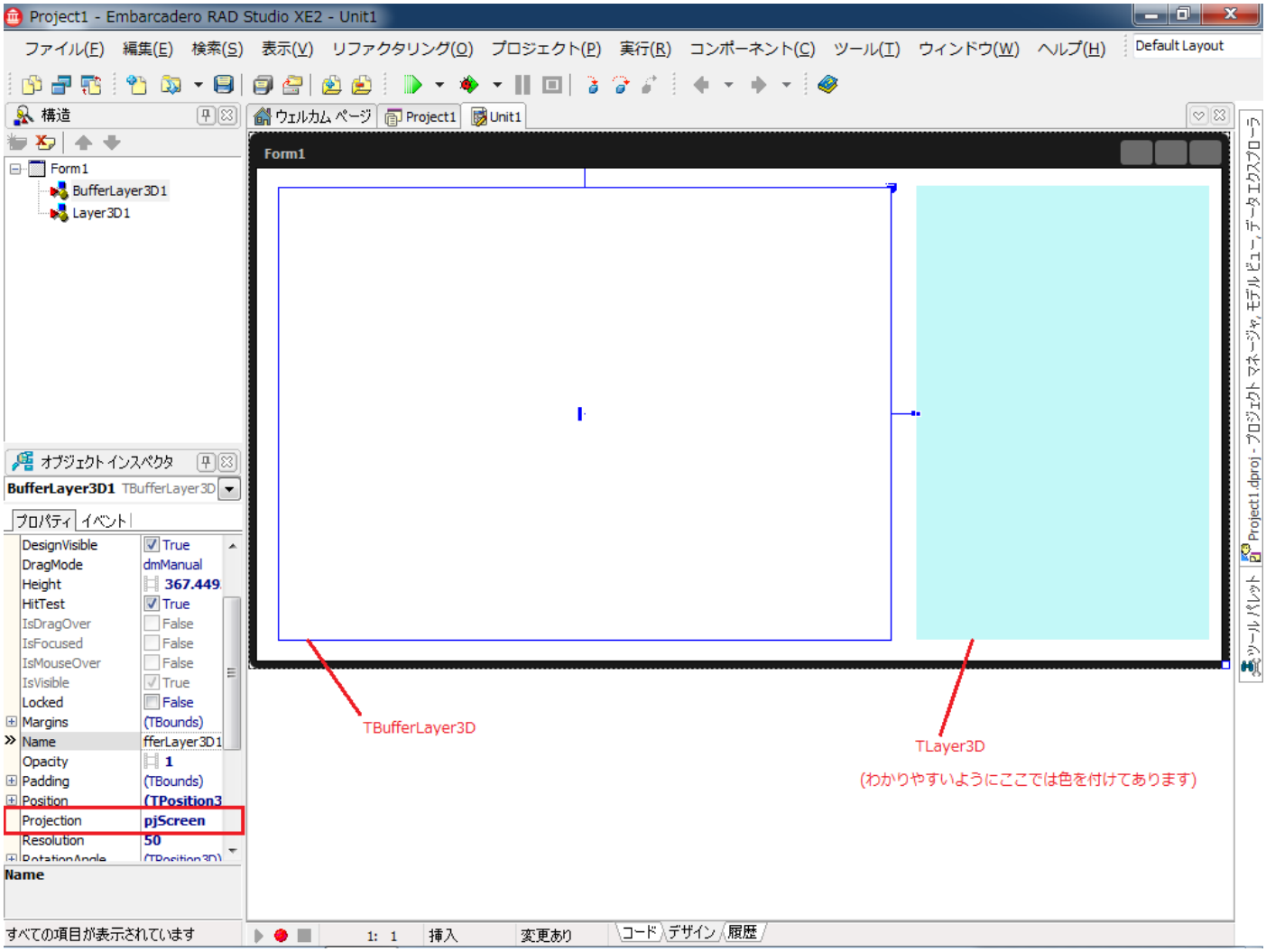
```
}
//---------------------------------------------------------------------------
void __fastcall TForm1::FormCreate(TObject *Sender)
{
   UnicodeString aStr;
   TStringDynArray strArray = TDirectory::GetFiles(".¥¥..¥¥..¥¥", "*.jpg");

   for (int i = 0; i < strArray.Length; i++) {
              aStr = strArray[i];
              cbToppings->Items->Add(System::Ioutils::TPath::GetFileNameWithoutExtension(aStr));
   }
}
//---------------------------------------------------------------------------

void __fastcall TForm1::aniMoveImageFinish(TObject *Sender)
{
           TListBoxItem *listItem = new TListBoxItem(lbToppings);
           listItem->Parent = lbToppings;

           TText *itemText;
           TImage *itemImage;

           listItem->StyleLookup = "imagelabel";

           itemText = (TText*)(listItem->FindStyleResource("Text"));
           itemText->Text = strTopping;

           itemImage = (TImage*)(listItem->FindStyleResource("Image"));
           if (itemImage != NULL) {
              itemImage->Bitmap = Image1->Bitmap;
              listItem->Height = Image1->Bitmap->Height;
           } else {
              ShowMessage("バインドする¥r¥n イメージがありません");
           }

           Image1->Bitmap->Clear(claWhite);
           Image1->Position->X = 32;
}
//---------------------------------------------------------------------------
```

3. 3D のピザアプリケーション

ここまで FireMonkey の 2D フォームを使用して、基本的な効果やスタイルの適用、アニメーションの設定などを行ってきました。このセクションでは、2 次元オブジェクトを 3 次元空間で動かし、FireMonkey 3D の基本的な扱い方を学びます。

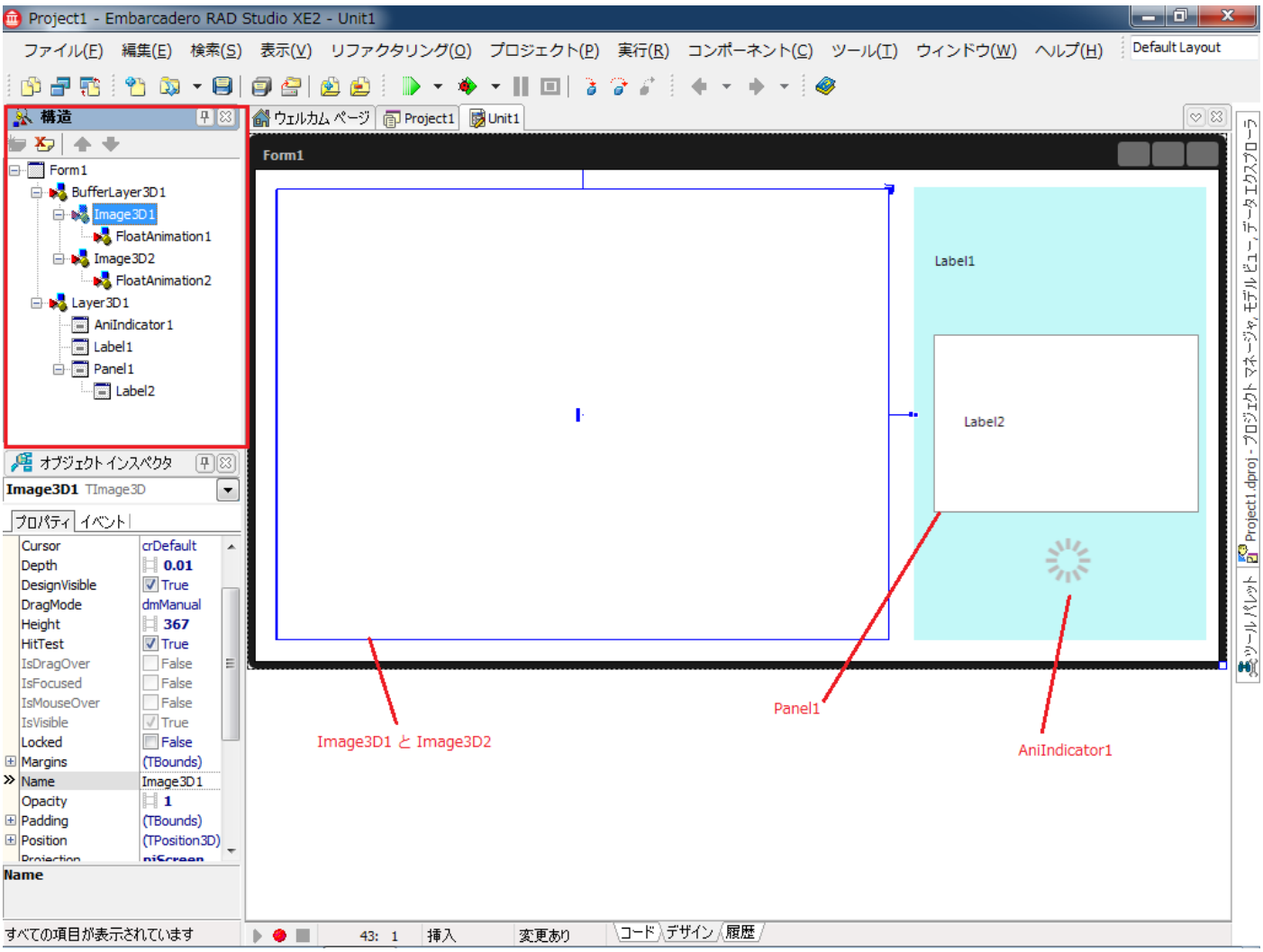
演習 5. 3D のピザアプリケーション

1. 新しくアプリケーションを作成します。[ファイル|新規作成|その他] を選択し、表示された「新規作成ダイアログ」の左側のツリーから 「Delphi プロジェクト」または「C++Builder プロジェクト」を選択し、右側の項目から「FireMonkey 3D アプリケーション」を選択します。
2. フォームを作成されたところで、[ファイル|すべて保存]で、 ¥FireMonkey3D フォルダを作成し、生成したフォーム等を保存します。
3. 次にツールパレットからフォーム上に TBufferLayer3D コンポーネントと TLayer3D コンポーネントを以下のように配置します。また双方の Projection プロパティを pjScreen に設定します。



4. BufferLayer3D と親子関係になるようにツールパレットから TImage3D を 2 つ置きます。先程と同様に TImage3D それぞれの Projection プロパティは pjScreen にします。また、それぞれの TImage3D に対してアニメーションを設定したいので、ツールパレットから TFloatAnimation を TImage3D それぞれに配置します。
5. Layer3D 側ですが、これも Layer3D と親子関係になるように、ツールパレットから TAniIndicator, TLabel, TPanel を置きます。また TPanel と親子関係になるようにツールパレットから TLabel を配置します。(構造ペインの状

態を確認しつつ作業を行ってください)



6. 配置するコンポーネントは以上です。次にそれぞれのコンポーネントのプロパティをオブジェクトインスペクタ上で設定します。

■Form3D1

Caption	Pizza メニュー3D
Color	Black

■BufferLayer3D1

Align	alMostLeft
Height	408
Projection	pjScreen
Resolution	50
Width	600
ZWrite	False

■Image3D1

Align	alCenter
Height	350

Projection	pjScreen
TwoSide	False
Width	500

■FloatAnimation1

AnimationType	atInOut
Enabled	True
Duration	2
Interpolation	itBack
StartValue	1
Stopvalue	179
PropertyName	RotationAngle.Y

■Image3D2

Align	alCenter
Height	350
Projection	pjScreen
TwoSide	False
Width	500

■FloarAnimation2

AnimationType	atInOut
AutoReverse	True
Duration	2
Enabled	True
Interpolation	itBack
StartValue	-180
PropertyName	RotationAngle.Y

■Layer3D1

Align	alClient
Fill.Color	White
Fill.kind	bkSolid
Height	408
Projection	pjScreen
Width	274

■AniIndicator1

Enabled	True
Height	113
TabOrder	0

Width	105

■Label1

Font.Size	44
Height	87
TabOrder	2
Text	Pizza Menu
TextAlign	taCenter
Width	249

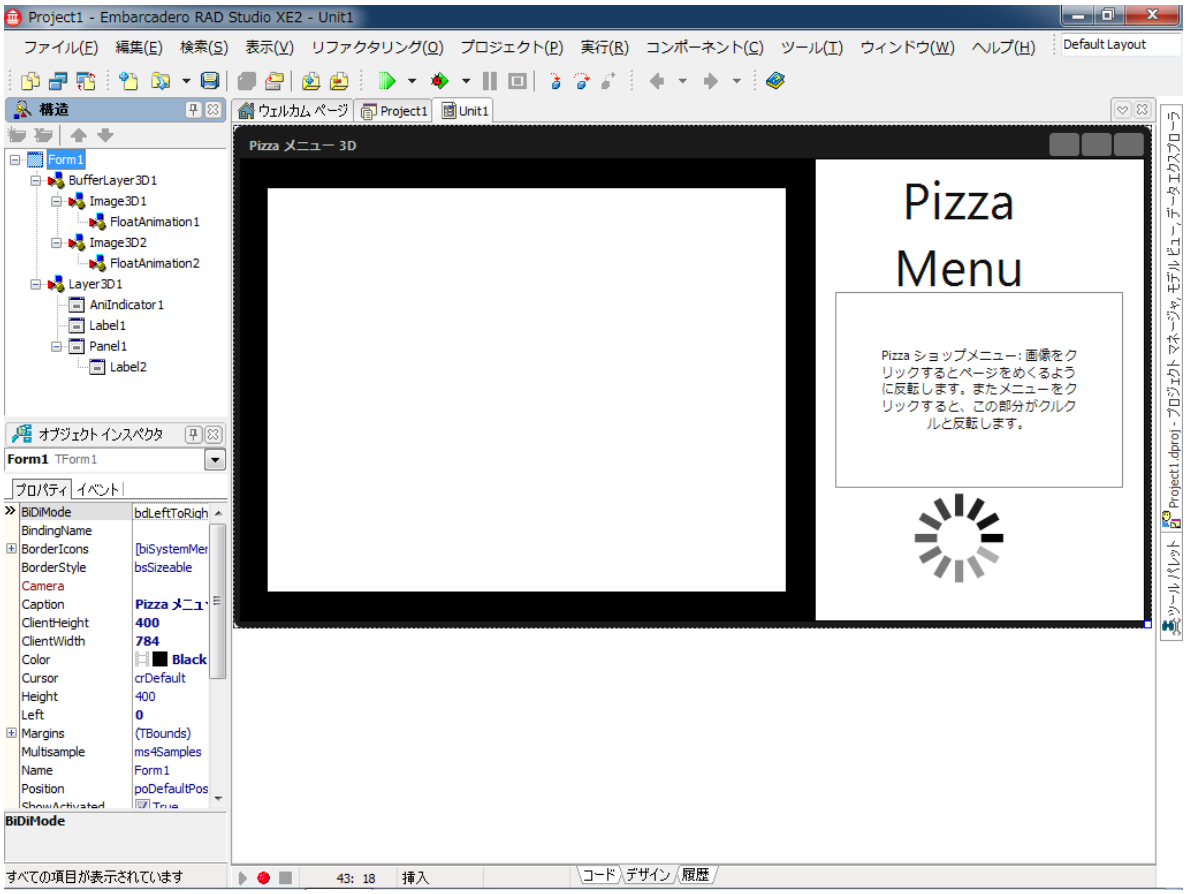
■Panel1

Align	alCenter
Height	200
TabOrder	1
Width	393

■Label2

Align	alCenter
Height	121
TabOrder	0
Text	Pizza ショップメニュー: 画像をクリックするとページをめくるように反転します。またメニューをクリックすると、この部分がクルクルと反転します。
TextAlign	taCenter
Width	241

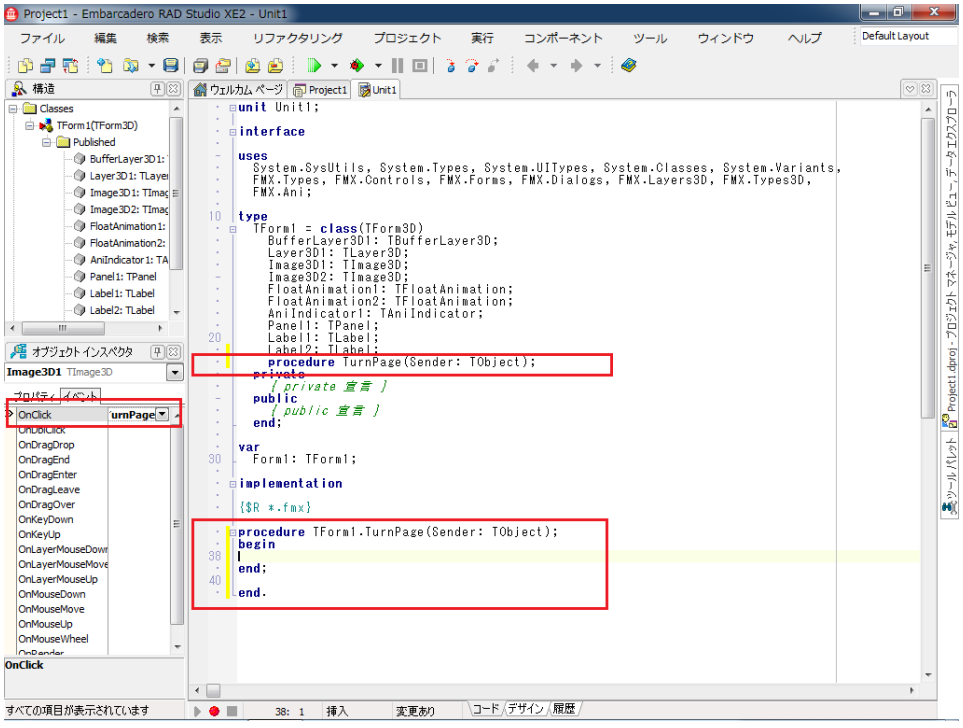
7. ここまで設定したところで、[ファイル|すべて保存]で保存します。



8. この段階で実行すると、左側のイメージの部分のみ反転するというアニメーションを確認することができます。

次に、この左側をクリックすると再度反転するというアクション処理を記述しましょう。

9. 構造ペイン上で Image3D1 を選択し、オブジェクトインスペクタで「イベント」タブをクリックします。
10. 表示されたイベントの OnClick の右の部分に TurnPage と入力します。イベントがクリックされた際に実行される TurnPage に関するコードが自動生成されます。



11. 生成された TurnPage に以下のコードを記述します。Y 軸を中心に 180 度回転させる仕組みとなります。

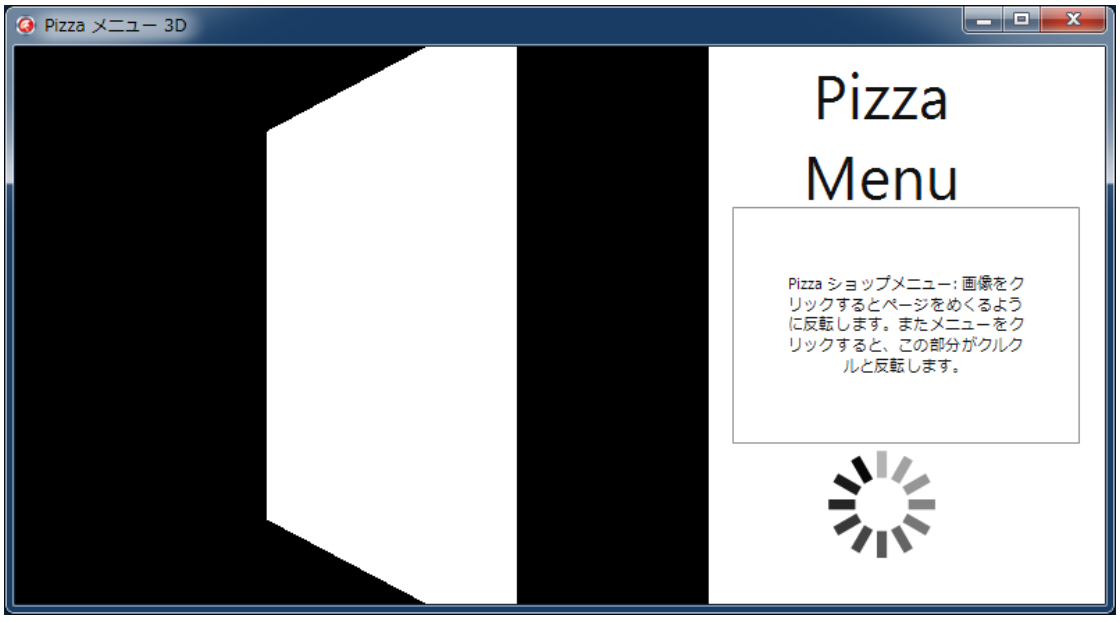
Delphi

```
FloatAnimation1.StartValue := Image3D1.RotationAngle.Y;
FloatAnimation1.StopValue := Image3D1.RotationAngle.Y + 180;

FloatAnimation2.StartValue := Image3D2.RotationAngle.Y;
FloatAnimation2.StopValue := Image3D2.RotationAngle.Y + 180;

FloatAnimation1.Start;
FloatAnimation2.Start;
```

C++Builder

```
FloatAnimation1->StartValue = Image3D1->RotationAngle->Y;
FloatAnimation1->StopValue = Image3D1->RotationAngle->Y + 180;

FloatAnimation2->StartValue = Image3D2->RotationAngle->Y;
FloatAnimation2->StopValue = Image3D2->RotationAngle->Y + 180;

FloatAnimation1->Start();
FloatAnimation2->Start();
```

12. また Image3D2 の OnClick イベントにも、この TurnPage を関連付けます。
13. [ファイル|すべて保存]で保存します。
14. [実行|実行] で実行させ、左側の部分をクリックすると、ページが回転するのを確認できます。



もうひとつのアニメーションですが、これは設計時にアニメーションオブジェクトを設定するのではなく、実行時にアニメーションを定義して動作するようにコードを記述します。

15. デザイン画面上で、Layer3D1 を選択し、オブジェクトインスペクタで「イベント」タブをクリックし、Image3D の時と同様に OnClick イベントの部分に FlipSide と入力し、クリックした際に実行されるコードを生成します。

16. 生成された FlipSide に以下のコードを記述します。

Delphi

```
Layer3D1.AnimateFloat('RotationAngle.Y',360, 2, TAnimationType.atInOut, TInterpolationType.itBack);
Layer3D1.AnimateFloat('Position.Z', 500, 1);
Layer3D1.AnimateFloatDelay('Position.Z', 0, 1, 1);
```

C++Builder

```
Layer3D1->AnimateFloat("RotationAngle.Y",360, 2, TAnimationType::atInOut, TInterpolationType::itBack);
Layer3D1->AnimateFloat("Position.Z", 500, 1);
Layer3D1->AnimateFloatDelay("Position.Z", 0, 1, 1);
```

Y 軸を中心に回転させる動作と、Z 軸に対する移動の動作を記述しています。
2D の時は X 軸と Y 軸に対する平面的な動作のアニメーションでしたが、3D にすると Z 軸が加わり、奥行きのある 3D の動作を行わせることができます。

17. [ファイル|すべて保存]で保存します。
18. [実行|実行] で実行させ、右側の部分をクリックすると、記述したアニメーションが動作するのを確認できます。

仕上げにピザの画像を Image3D に張り付けて、回転するたびに別の画像が表示される部分を作成します。

19. 画像ファイルは、¥FireMonkey3D フォルダ下にコピーしておきます。
20. ファイルに関する IO 処理を行うため、Delphi では、uses に IOUtils を追加します。また、C++Builder では、#include <IOUtils.hpp>と #include <System.Types.hpp>(.h 側) を追加します。
21. 作業用のエリアを private に追加します。

Delphi

```
private
  { private 宣言 }
  imagesList: TStringDynArray;
  imagePos: Integer;
```

C++Builder

```
private:  // ユーザー宣言
```

```
        TStringDynArray   imagesList;
        int imagePos;
```

22. 画像に関する処理を行う関数を public に作成します。

Delphi

```
  Public
    { public 宣言 }
    procedure LoadNewImage;
```

C++Builder

```
public:                 // ユーザー宣言
        __fastcall TForm3D1(TComponent* Owner);
        void __fastcall LoadNewImage();
```

23. 記述した、LoadNewImage にカーソルを載せた状態で、Ctrl + Shift + C キーを押すと、関数の実装部が生成されますので(Delphi のみ)、ここに以下のコードを記述します。

Delphi

```
procedure TForm1.LoadNewImage;
begin
  if (imagePos mod 2) = 0 then
  begin
    Image3D1.Bitmap.Clear(claWhite);
    Image3D1.Bitmap.LoadThumbnailFromFile(imagesList[imagePos], 500, 350, false);
  end
  else begin
    Image3D2.Bitmap.Clear(claWhite);
    Image3D2.Bitmap.LoadThumbnailFromFile(imagesList[imagePos], 500, 350, false);
  end;
  Inc(imagePos);
  if imagePos >= Length (imagesList) then
    imagePos := 0; // reset
end;
```

C++Builder の場合は、以下のコードを記述します。

C++Builder

```
void __fastcall TForm3D1::LoadNewImage()
{
  if (imagePos%2 == 0) {
        Image3D1->Bitmap->Clear(claWhite);
        Image3D1->Bitmap->LoadThumbnailFromFile(imagesList[imagePos], 500, 350, false);
  } else {
        Image3D2->Bitmap->Clear(claWhite);
        Image3D2->Bitmap->LoadThumbnailFromFile(imagesList[imagePos], 500, 350, false);
  }
```

```
  imagePos++;
  if (imagePos >= imagesList.Length) {
        imagePos = 0; // reset
  }
}
```

24. また、フォームの生成時にイメージファイルを読み込むよう Form3D の OnCreate イベントを作成し、以下のコードを記述します。

Delphi

```
procedure TForm1.Form3DCreate(Sender: TObject);
begin
  imagesList := TDirectory.GetFiles('.¥..¥..¥', '*.jpg');
  imagePos := 0;

  LoadNewImage;
  LoadNewImage;
end;
```

C++Builder

```
void __fastcall TForm3D1::Form3DCreate(TObject *Sender)
{
  imagesList = TDirectory::GetFiles(".¥¥..¥¥..¥¥", "*.jpg");
  imagePos = 0;

  LoadNewImage();
  LoadNewImage();
}
```

25. 最後に前に作成した TurnPage の最初に、画像ファイルを読み込むための関数呼び出しを追加します。

Delphi

```
procedure TForm1.TurnPage(Sender: TObject);
begin
  LoadNewImage;

  FloatAnimation1.StartValue := Image3D1.RotationAngle.Y;
  FloatAnimation1.StopValue := Image3D1.RotationAngle.Y + 180;
<以下省略>
```

C++Builder

```
void __fastcall TForm3D1::TurnPage(TObject *Sender)
{
  LoadNewImage();

  FloatAnimation1->StartValue = Image3D1->RotationAngle->Y;
```

```
  FloatAnimation1->StopValue = Image3D1->RotationAngle->Y + 180;
<以下省略>
```

26. 以上で、完成です。[ファイル|すべて保存] で保存し、[実行|実行]で作成したアプリケーションを動作させます。

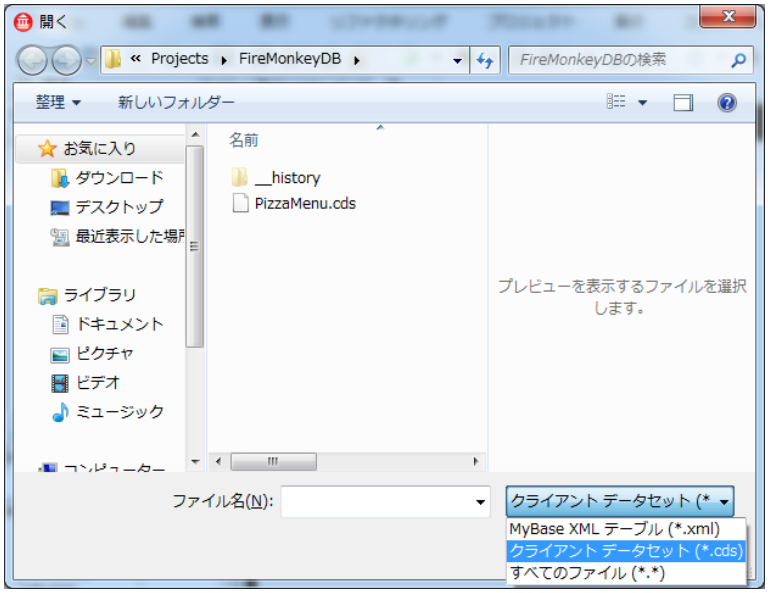
4. DataBinding のピザアプリケーション

FireMonkey には、TDBxxx のような特別なデータ対応コントロールはありません。LiveBinding と呼ばれる、オブジェクトをプロパティを介して相互にバインドする機能を用います。

この演習では、読み取り専用のデータエディタを作成します。

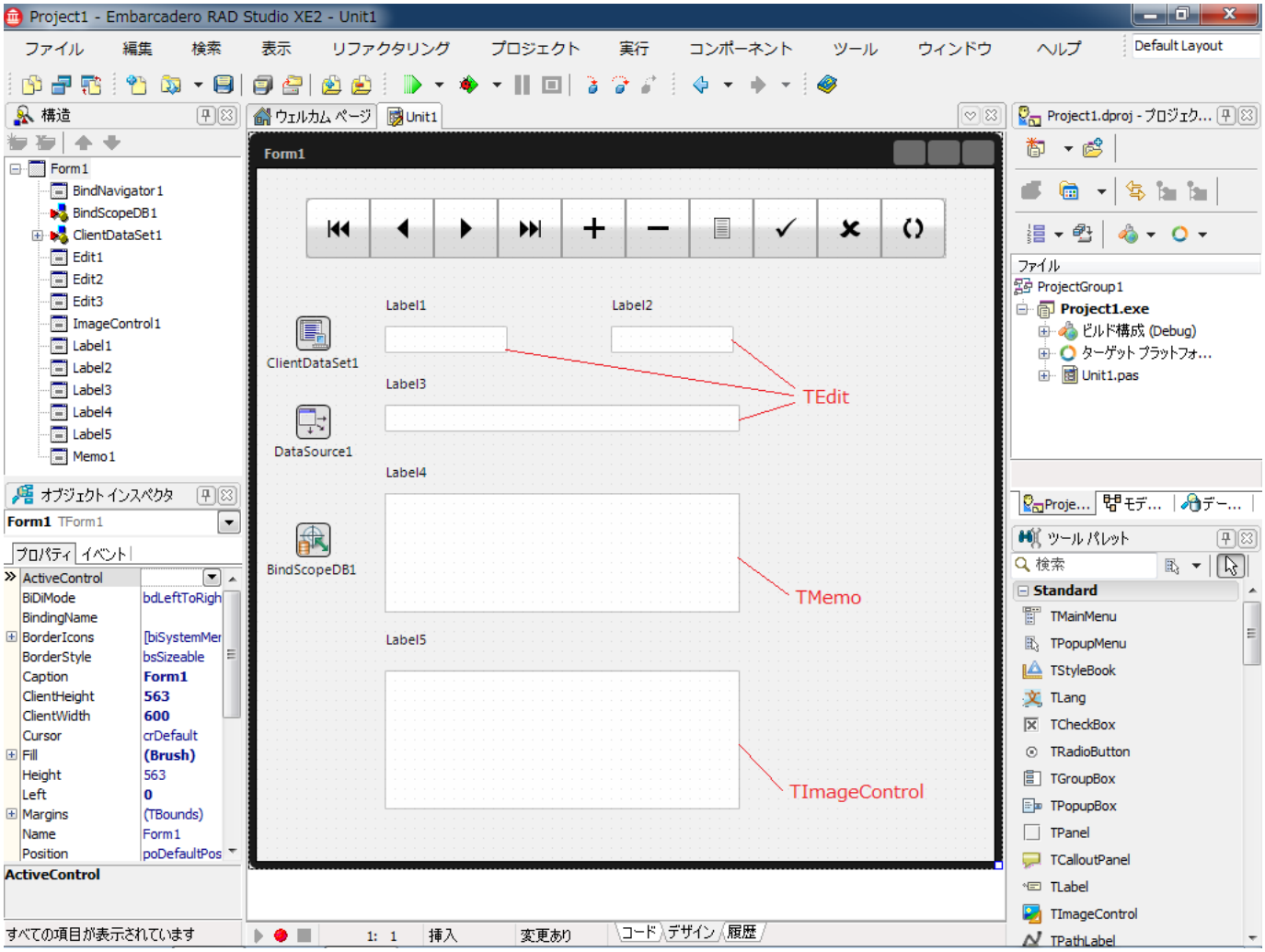
演習 6. DataBinding の使用(cds ファイルを使用)

1. [ファイル|新規作成|FireMoneky HD アプリケーション – Delphi] または [ファイル|新規作成|FireMonkey HD アプリケーション – C++Builder]で新規にアプリケーションを作成します。
2. [ファイル|すべて保存] で ¥FireMonkeyDB フォルダを作成し、生成されたファイルを保存します。
3. また、.cds ファイルをこのフォルダにコピーしておきます。
4. ツールパレットから TClientDataSet と TDataSource をフォーム上に置きます。
5. デザイナ画面上で ClientDataSet1 を選択し、マウスの右ボタンを押して表示されたポップアップメニューより「MyBase テーブルから読み込み」を選択します。
6. 表示されたダイアログで、種別を「クライアントデータセット(*.cds)」とし、表示された .cds ファイルを選択します。

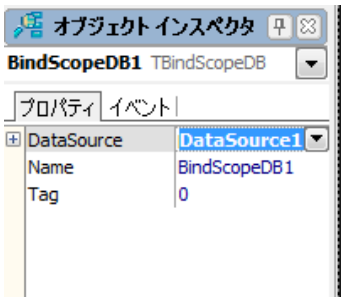


7. デザイナ画面上で、DataSource コンポーネントを選択し、オブジェクトインスペクタ上で、DataSet プロパティの値を ClientDataSet1 に設定します。
8. 次に、ツールパレットから 、以下のコンポーネントをフォーム上に置きます。
   - TBindNavigator
   - TBindScopeDB
   - TLabel x5
   - TEdit x3
   - TMemo
   - TImageControl

配置のイメージは以下を参考にしてください。



9. 次に、デザイナ画面上で BindScopeDB1 コンポーネントを選択し、オブジェクトインスペクタで、BindScopeDB1 の DataSource プロパティに DataSource1 を設定します。



10. デザイナ画面上で BindNavigator1 コンポーネントを選択し、オブジェクトインスペクタで、BindNavigator1 の BindScpe プロパティに BindScopeDB1 を設定します。

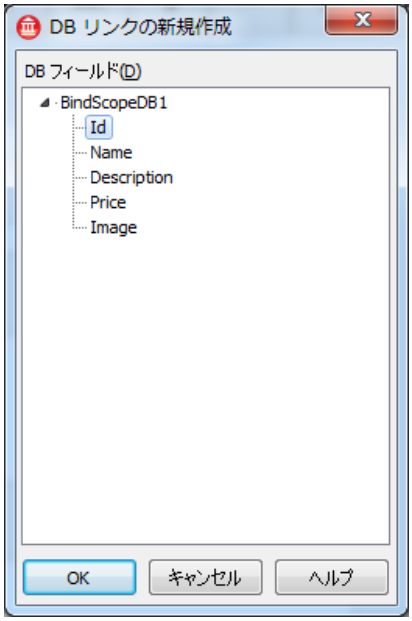
11. Label コンポーネントの Text プロパティの値を、それぞれ次のように設定します。

Label1	ID
Label2	名前
Label3	値段
Label4	説明
Label5	イメージ

値を表示する Edit, Memo, ImageControl と ClientDataSet で読み込まれたデータのフィールドとを関連付けます。

12. デザイン画面で Edit1 を選択し、オブジェクトインスペクタの「DB フィールドへのリンク」をクリックします。

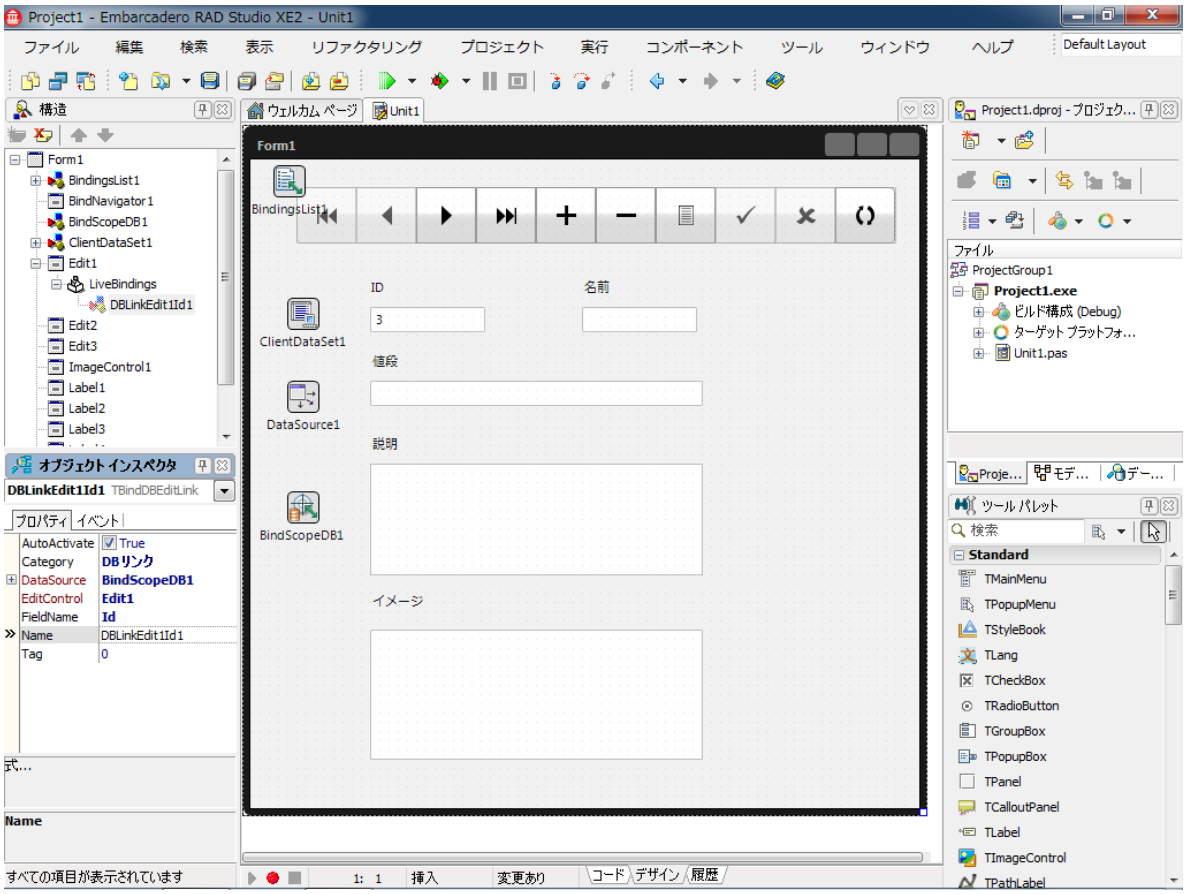
13. DB リンクの新規作成のダイアログが表示されます。



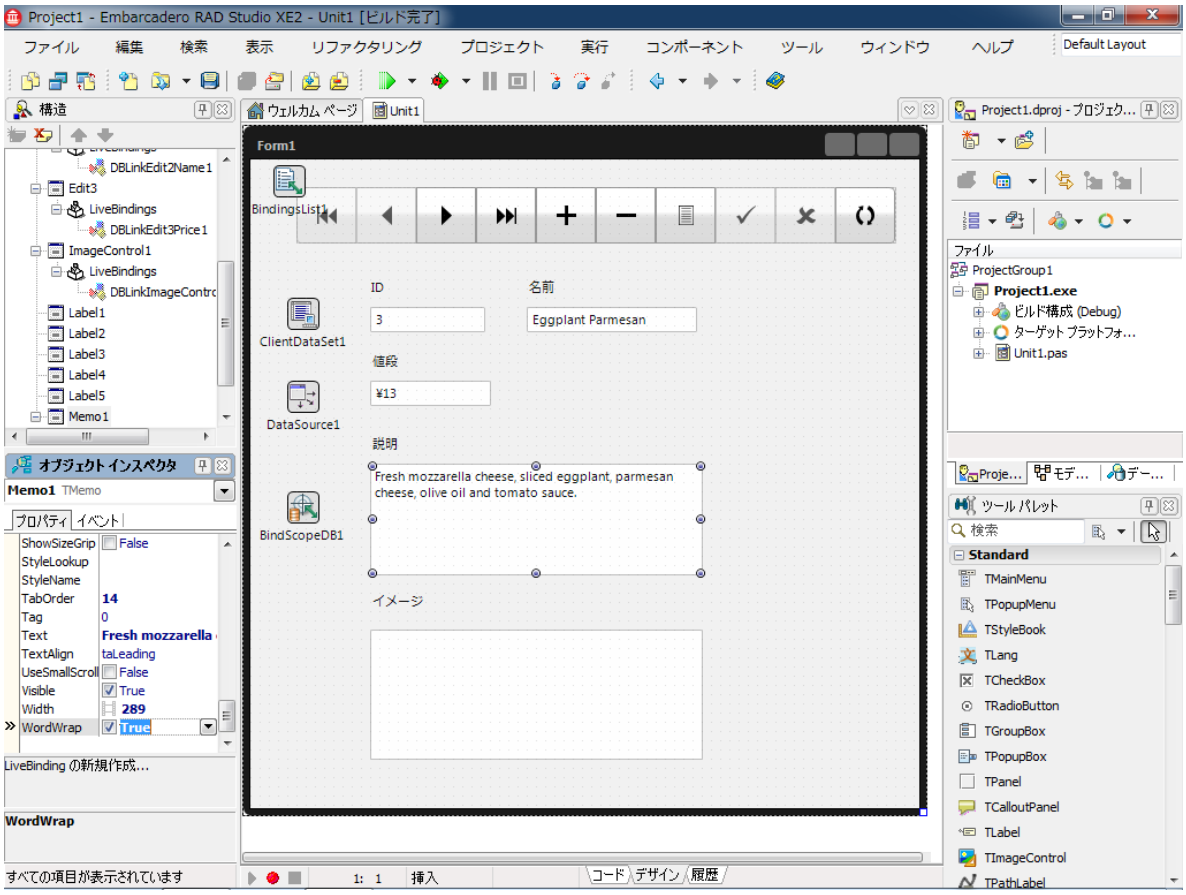
14. 表示された DB フィールドの Id を選択し [OK]ボタンを押します。

これで、ClientDataSet に読み込まれたデータの Id フィールドと Edit1 が関係付けられ、データが表示されます。

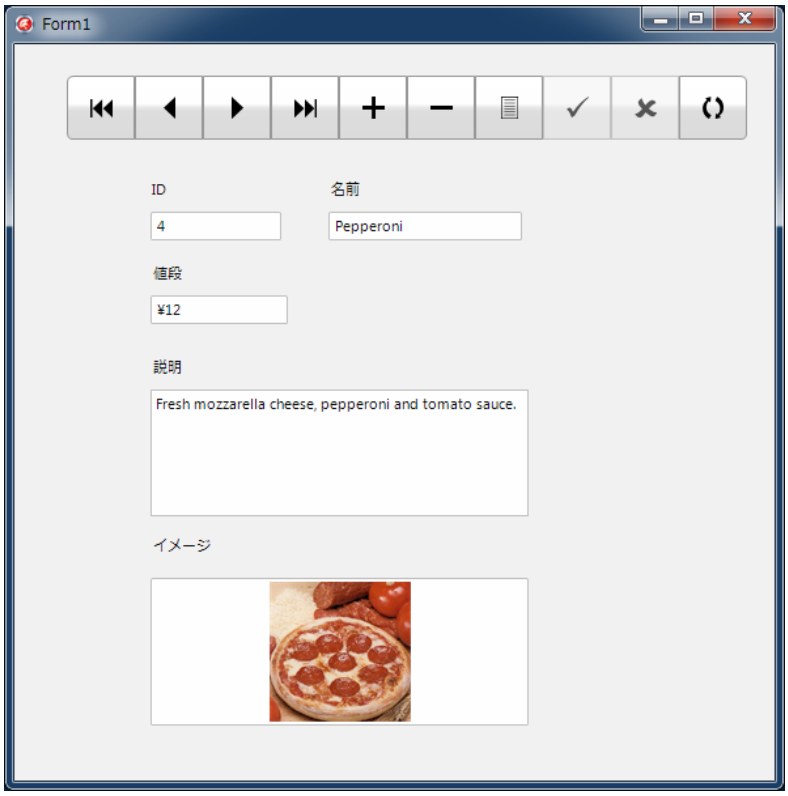
また、この際、自動的に TBindList コンポーネントが生成され、フォーム上に配置されます。



15. 同様に、Edit2 には Name フィールドを、Edit3 には Price フィールドをそれぞれ関連付けます。
16. また、Memo1 には Description を、ImageControl1 には Image フィールドをそれぞれ関連付けます。
17. Memo コンポーネントですが、デフォルトでは WordWrap が False ですので、オブジェクトインスペクタ上で True に設定し、文字が折り返されて表示されるようにします。



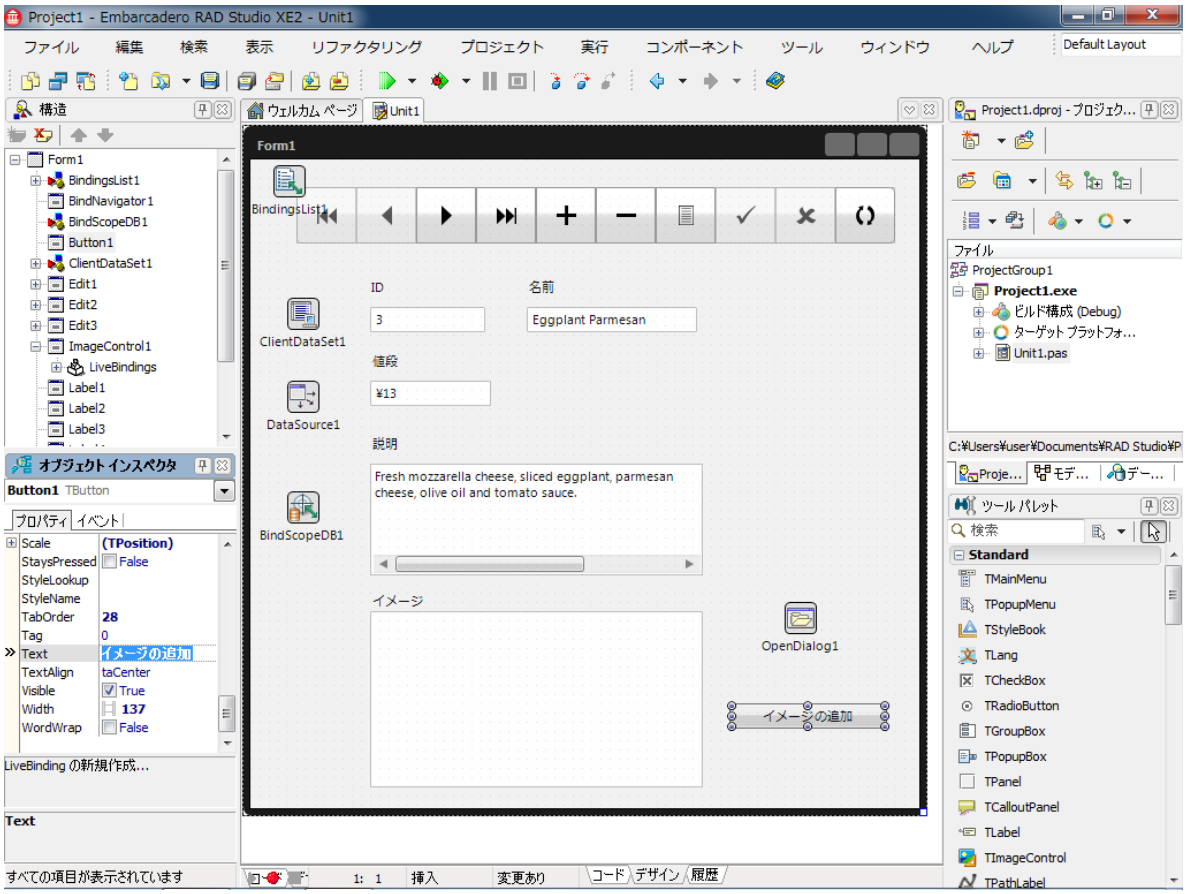
18. ここまでの内容を保存します。[ファイル|すべて保存] で保存し、[実行|実行] で動作させます。
上部のナビゲータボタンを操作し、レコードの移動を行うと、そのレコードのデータがフォーム上に表示されるのを確認することができます。



このように、特にコードを記述することなく LiveBinding を使用し、データとそれぞれのコンポーネントを関連付け、処理を行うことが可能です。

このフォームに 1 つ機能を追加します。
データベースの Image フィールドに対し、読み込んだ画像ファイルのイメージで更新するという機能です。

19. フォーム上に TButton と TOpenDialog コンポーネントを追加します。
20. デザイナ画面上で追加した Button1 を選択し、オブジェクトインスペクタ上で、Button1 の Text プロパティの値を「イメージの追加」に設定します。



21. 次にデザイナ画面上で Button1 をダブルクリックし、OnClick イベントを生成します。
22. 生成した OnClick イベントに以下のコードを記述します。

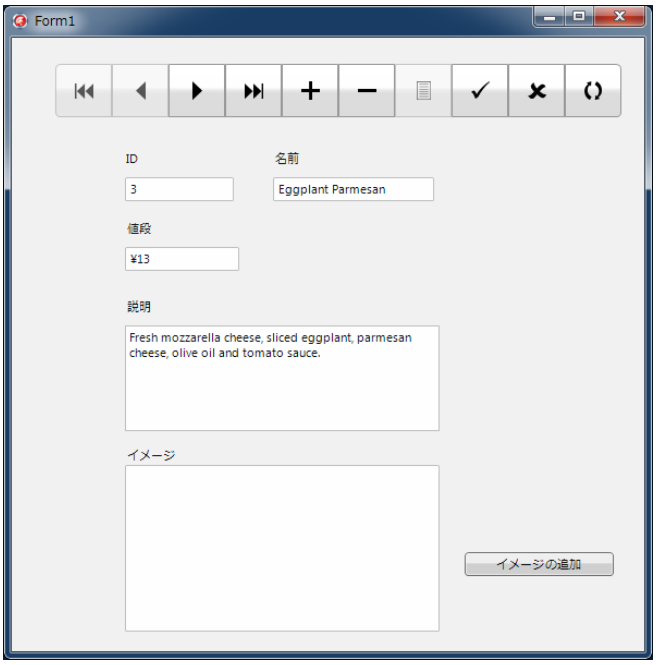
Delphi

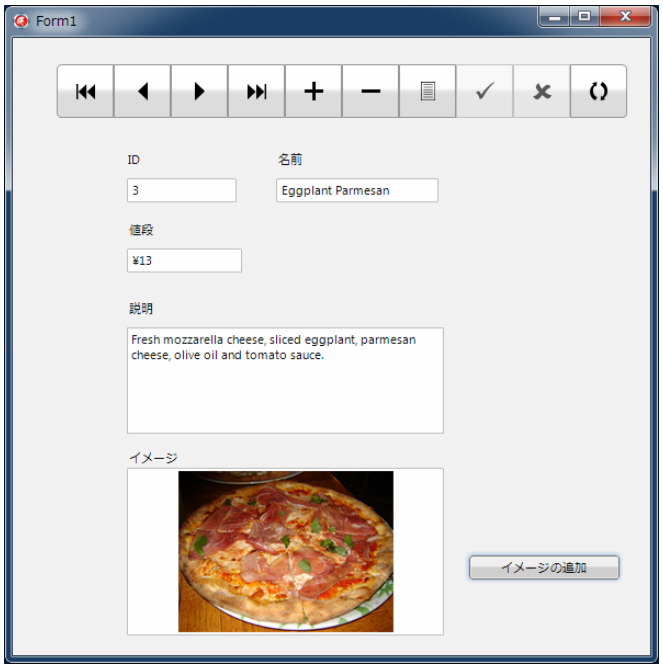
```
if OpenDialog1.Execute then
begin
  ClientDataSet1.Edit;
  (ClientDataSet1.FieldByName('Image') as TGraphicField).LoadFromFile(OpenDialog1.FileName);
  ClientDataSet1.Post;
end;
```

C++Builder

```
if (OpenDialog1->Execute()) {
  ClientDataSet1->Edit();
  ((TGraphicField*)ClientDataSet1->FieldByName("Image"))->LoadFromFile(OpenDialog1->FileName);
  ClientDataSet1->Post();
}
```

23. [ファイル|すべて保存] で保存します。
24. [実行|実行] で動作させ、何もイメージが表示されていないところで [イメージの追加] ボタンをクリックし、画像イメージのファイルを指定すると、先程まで空白だったイメージの部分に、指定した画像イメージが表示されます。


Form1
ID
3
名前
Eggplant Parmesan
値段
¥13
説明
Fresh mozzarella cheese, sliced eggplant, parmesan cheese, olive oil and tomato sauce.
イメージ
イメージの追加


Form1
ID
3
名前
Eggplant Parmesan
値段
¥13
説明
Fresh mozzarella cheese, sliced eggplant, parmesan cheese, olive oil and tomato sauce.
イメージ
イメージの追加

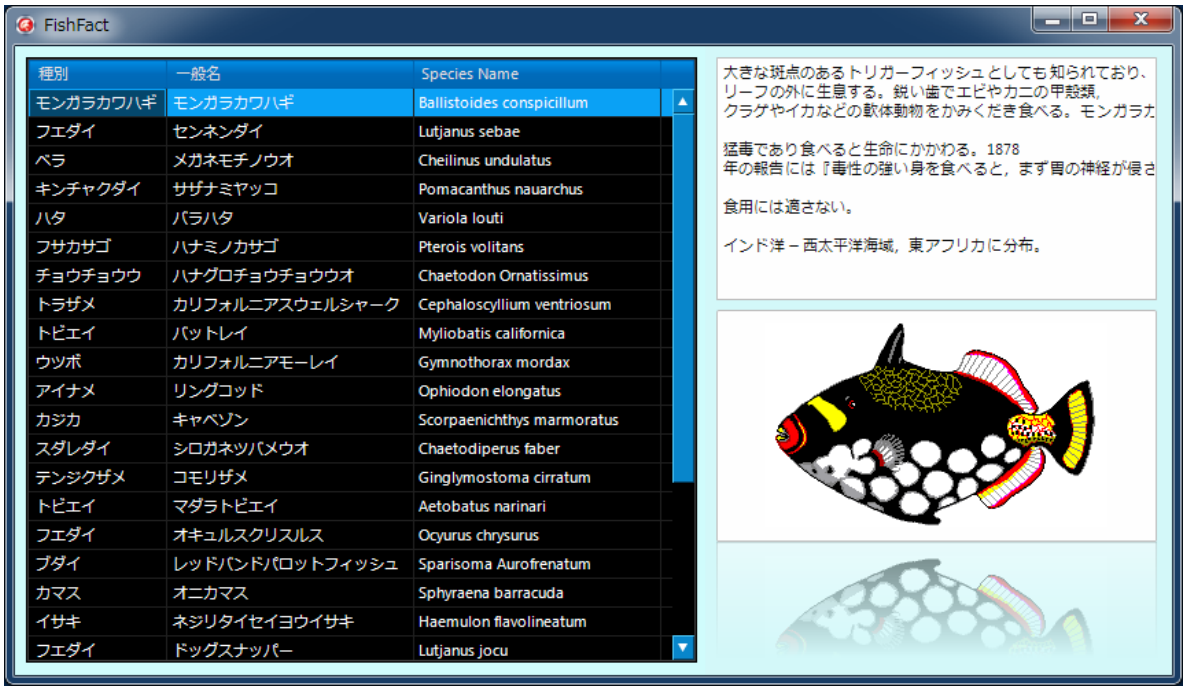
5. 課題（時間に余裕のある受講者の方へ）

今まで学んだことを使用して、演習で作成したアプリケーションの動作を拡大してみましょう。

5-1. 課題

- 演習 5 で作成した 3D のピザアプリケーションですが、左側の画像の回転中にマウスでクリックすると、画像自身は変わるのですが終点が途中になってしまうという問題があります。
  回転のアニメーションの開始と終了の際に、マウスを捕捉しないように、あるプロパティを設定するコードを追加し、この問題を解決してください。

- 演習 5 と演習 6 の応用で、フォーム上に StringGrid を使用して、魚の図鑑情報データベースのレコード(種別と一般名と Special name のみ) を表示します。マウスで StringGrid 内のレコードをクリックすると、横に表示された説明とイメージ部分が回転して該当するレコードの説明部分とイメージを表示するアプリケーションを作成してください。
  1. HD アプリケーションで作成した場合、回転する部分は TViewPort3D と TLayer3D を入れ子にして回転部分を作成します。
  2. 3D アプリケーションで作成した場合、TLayer3D を２つ用意し、一方は StringGrid、もう一方は説明とイメージを扱うように設計します。
  3. アニメーションの回転方向は、好きな方向に回転させてください。
  4. 製品付属のスタイルを使用します。
  5. イメージ部分には、表示効果を付けて鏡に映ったようにします。



5-2. 解答例

- 3D のピザアプリケーションですが、HitTest プロパティを使用します。 HitTest プロパティが True に設定されている場合、マウスの OnClick イベントおよび OnDblClick イベントを捕捉します。

  回転中は OnClick イベントを捕捉しないように、回転させるための TurnPage の先頭に

  Delphi

  ```
  Image3D1.HitTest := False;
  Image3D2.HitTest := False;
  ```

  C++Builder

  ```
  Image3D1->HitTest = False;
  Image3D2->HitTest = False;
  ```

  のコードを記入して、このコンポーネントに対するクリック動作を無効にします。

  あとは、FloatAnimation1 および FloatAnimation2 の OnFinish イベントを作成し、このイベント内で上記で False にした部分を True にセットするコードを記述します。

  Delphi

  ```
  Image3D1.HitTest := True;
  Image3D2.HitTest := True;
  ```

  C++Builder

  ```
  Image3D1->HitTest = True;
  Image3D2->HitTest = True;
  ```

- 応用のアプリケーションですが、以下に HD アプリケーションとして作成した場合の手順を記します。
  1. [ファイル|新規作成|FireMonkey HD アプリケーション – Delphi]または[ファイル|新規作成|FireMonkey HD アプリケーション – C++Builder] で設計を開始します。
  2. ツールパレットから、フォーム上に TClientDataSet, TDataSource, TBindScopeDB を置きます。
  3. ClientDataSet1 に対し「MyBase テーブルからの読み込み」を実行し、biolife_j.xml ファイルを選択して関連付けます。
  4. DataSource1 の DataSet プロパティに ClientDataSet1 を、BindScopeDB1 の DataSource プロパティに DataSource1 をそれぞれ関連付けます。
  5. フォーム上に TStringGrid, TStyleBook を置きます。
  6. StringGrid1 の「DB データソースへのリンク」で、BindScopeDB1 を選択して [OK]ボタンをクリックして関連付けを行います。
  7. 設計画面上で、StringGrid1 を選択し、マウスの右ボタンを押し「カラムエディタ」を選択します。
  8. カラムエディタ上で、マウスの右ボタンを押し「すべてのフィールドを追加」を選択します。
  9. 追加されたフィールドの内 Category, Common_name, Species name 以外のフィールドは削除します。これで StringGrid1 上には 3 つのフィールドのみ表示されるようになります。
  10. 構造ペイン上で StringGrid1|LiveBindings|DBLinkStringGrid11|Columns|Category を選択し、オブジェクトインスペクタ上で Header プロパティの値を Category から種別に変更します。
  11. また、Width プロパティの値を 100 とします。

12. 同様に構造ペイン上で StringGrid1|LiveBindings|DBLinkStringGrid11|Columns|Common_name を選択し、オブジェクトインスペクタ上で Header プロパティの値を Common_name から一般名に変更します。
13. また、Width プロパティの値を 180 とします。
14. さらに、構造ペイン上で “StringGrid1|LiveBindings|DBLinkStringGrid11|Columns|Speciaes name” を選択し、オブジェクトインスペクタ上で Width プロパティの値を 180 とします。
15. 設計画面上で、StyleBook1 をダブルクリックします。
16. [読み込み]をクリックし、Styles ディレクトリ 内の AquaGraphite を選択し、[適用して閉じる]をクリックします。
17. オブジェクトインスペクタ上で、Form1 を選択し、StyleBook プロパティに StyleBook1 を設定します。
18. フォーム上に TViewPort3D を置き、Align プロパティを alRight に、Width プロパティを 340 に設定します。
19. この時点で、フォームの大きさと StringGrid と ViewPort3D の位置関係を調節します。
20. ViewPort3D1 上に TLayer3D を置き、Projection プロパティを pjScreen に、Align プロパティを alClient に設定します。
21. Layer3D1 上に TMemo, TImageControl を置き、それぞれ「DB データソースへのリンク」で、Notes, Graphic フィールドと関連付けます。
22. Memo1 の WordWrap プロパティの値を True にします。
23. ImageControl1 に TReflectionEffect を追加し、鏡のように表示される効果を追加します。
24. Layer3D 上の Memo1 と ImageControl1 の位置と大きさを調節します。
25. オブジェクトインスペクタ上で、StringGrid1 のイベントタグを選択し、OnClick イベントをダブルクリックし、以下のコードを記述します。(回転の方向は一例です)

Delphi

```
Layer3D1.AnimateFloat('RotationAngle.Z',360,1,TAnimationType.atInOut, TInterpolationType.itSinusoidal);
Layer3D1.AnimateFloat('Position.Z',500,1,TAnimationType.atInOut, TInterpolationType.itSinusoidal);
Layer3D1.AnimateFloatDelay('Position.Z',0,1,1,TAnimationType.atInOut, TInterpolationType.itSinusoidal);
ReflectionEffect1.UpdateParentEffects;
```

C++Builder

```
Layer3D1->AnimateFloat(“RotationAngle.Z”,360,1,TAnimationType::atInOut, TInterpolationType::itSinusoidal);
Layer3D1->AnimateFloat(“Position.Z”,500,1,TAnimationType::atInOut, TInterpolationType::itSinusoidal);
Layer3D1->AnimateFloatDelay(“Position.Z”,0,1,1,TAnimationType::atInOut, TInterpolationType::itSinusoidal);
ReflectionEffect1->UpdateParentEffects();
```

26. 保存し、実行して動作を確認します。