



Embarcadero conference

embarcadero®



Efeitos e Animações no Firemonkey

[Rodrigo Mourão]

[Firemonkey para desenvolvedores VCL]

- Apresentação
- O Firemonkey
- Estrutura da FMX
- Interfaces na FMX
- RTL



[Rodrigo Mourão]

- Gestor de TI e Empreendedor Digital
- Minha missão é impactar vidas através do conhecimento
- Nos últimos 8 anos deixei o medo me impedir de fazer o que mais amo profissionalmente: ensinar
- E desde outubro/15, corro atrás para tirar o atraso
- Apaixonado pelo Delphi
 - fb.com/blogrodrigomourao
 - youtube.com/thermfactory
 - telegram.me/rmfactory
 - @rodrigocmourao



O Firemonkey

[Bibliotecas do RAD Studio]

- O RAD Studio possui 3 grandes bibliotecas
 - VCL – Visual Component Library
 - RTL – Run-Time Library
 - FMX – Firemonkey
- A RTL é compartilhada entre Delphi e C++ e suportada por todas as plataformas que o RAD Studio trabalha

[Features do FireMonkey]

- Camada de abstração Cross-platform para recursos dos sistemas operacionais
- Suporte a gráficos 2D and 3D
- Mecanismo de Vetor poderoso, similar a Adobe Flash e Microsoft WPF
- Anti-aliased eficiente em tempo real nos vetores gráficos. Resolução independente com suporte a alpha blending and gradientes
- WYSIWYG

[Features do FireMonkey]

- Controles GUI poderosos - window, button, textbox, numberbox, memo, anglebox, list box, etc
- Mecanismos de Skins baseados em vetores gráficos com dezenas de temas pré-definidos
- Formas e gráficos 2D com brushes, pens, geometries e transforms embutidos
- Animações avançadas rerenderizadas por threads em background, precisas e de fácil utilização. Uso mínimo de CPU e com correção automática de frame rate.
- Suporte a JPEG, PNG, TIFF, and GIF

[Veja bem]

- FMX e VCL não são compatíveis, são bibliotecas completamente diferentes, você não deve utilizá-las na mesma aplicação, no entanto você pode:
 - Usar bibliotecas FMX na VCL e vice e versa
 - Realizar a conversão manual (com os devidos cuidados) de suas aplicações
 - Recorrer ao Mida Converter (pago) para fazer a tarefa pesada



[Veja bem]

- Actions - Ao contrário de actions na VCL, actions no FireMonkey não suportam um Action Manager, logo você não pode usar um Action Manager para gerir as actions no FireMonkey
- Check Boxes - No FireMonkey, você deve usar o evento OnChange em vez de OnClick para interceptar as mudanças de seleção dos Check Boxes. A propriedade para obter o estado do componente é IsChecked, não Checked como na VCL
- Child-Parent Relationships - Na VCL, apenas TForms, TFrames, TPanels e Data Modules podem ser containers outros componentes. No FireMonkey, qualquer objeto pode ser container de qualquer outra coisa e você deve gerenciar isso pelo Structure View

[Veja bem]

- Colors – As cores no FireMonkey tem um componente alfa (opacidade), além dos tradicionais RGB. O FireMonkey usa constantes de cores da unit System.UIConsts. Estas constantes tem prefixados "cla" em vez de "cl". As cores são representadas pelos tipos TAlphaColor e TAlphaColorRec
- Control Positioning - Na VCL, a posição de um controle determinada pelas propriedades Left e Top, no FireMonkey os componentes são posicionados pelas propriedades aninhadas X e Y da propriedade Position do tipo TPosition. Controles 3D possuem uma instância de TPosition3D, com uma coordenada Z adicional. Para o tamanho permanece width e height (e depth para controles 3D)

[Veja bem]

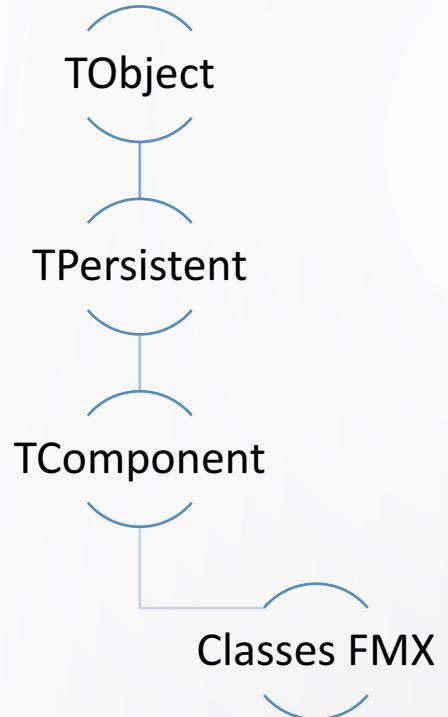
- Database Grid Events - Os eventos OnDrawDataCell e OnDrawColumnCell podem ser substituído no FireMonkey utilizando TStringGrid com LiveBindings
- Display Text - O texto do label vem da propriedade Text no FireMonkey e não caption como na VCL
- Font Size - Na VCL tamanhos de fonte são determinados em pontos na proporção de 72 pontos por polegada. No FireMonkey, as fontes são definidas em DIP, na proporção de 96 pontos pro polegada. Por isso, um texto usando o mesmo Font.Size é menor em FireMonkey
- Radio Buttons and Groups - Para converter um TRadioGroup da VCL para FireMonkey, você pode usar um TPanel ou TGroupBox. Os Radio Buttons em TGrouBox diferentes na VCL são independentes, para ter essa independência no Firemonkey você precisa especificar manualmente a propriedade GroupName para cada TRadioButton



Estrutura da FMX

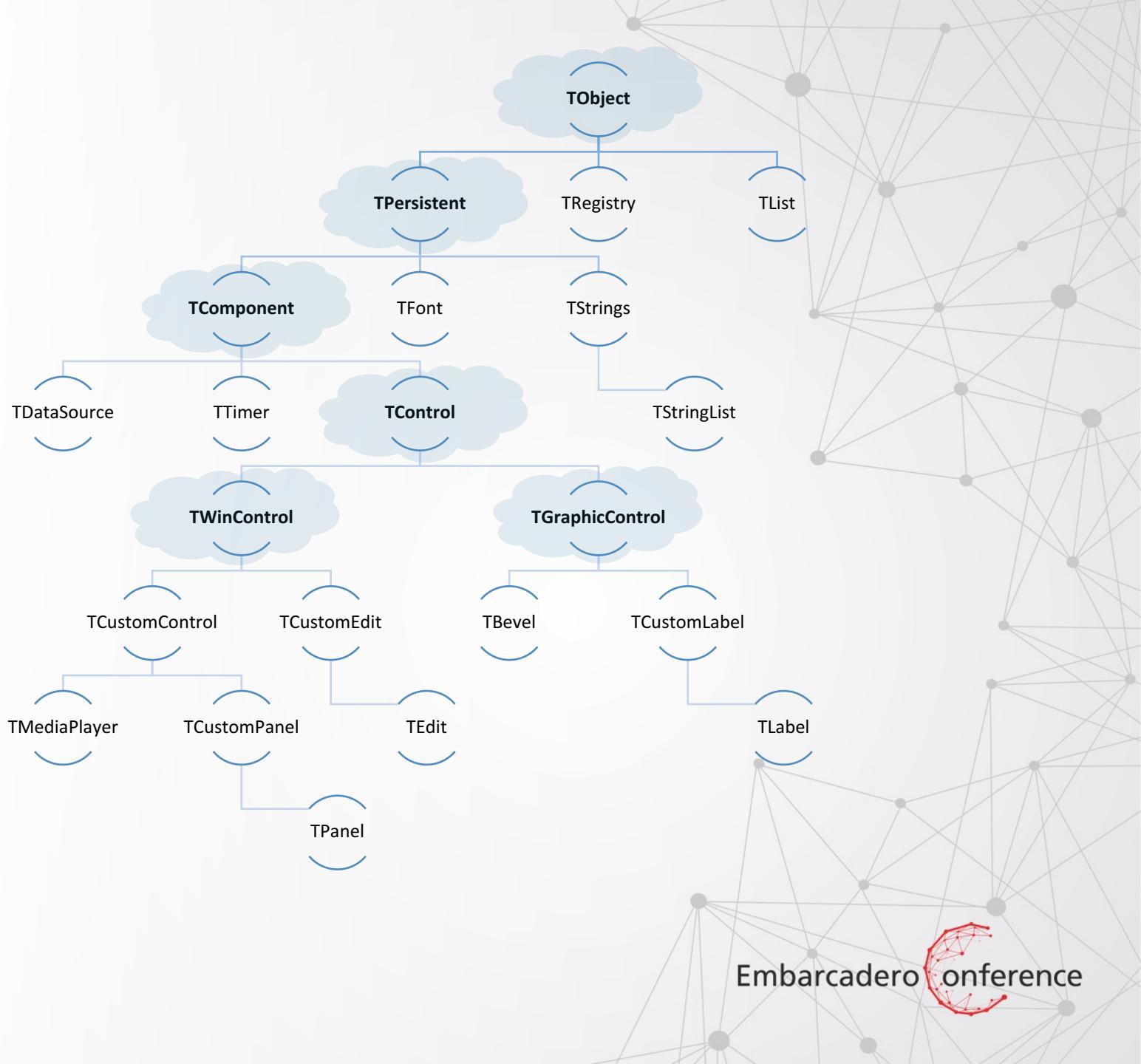
[VCL x FMX]

- A vcl e o Firemonkey possuem classes ancestrais em comum
- Como não poderia de ser, as classes do fmx descendem (indiretamente) de tobject

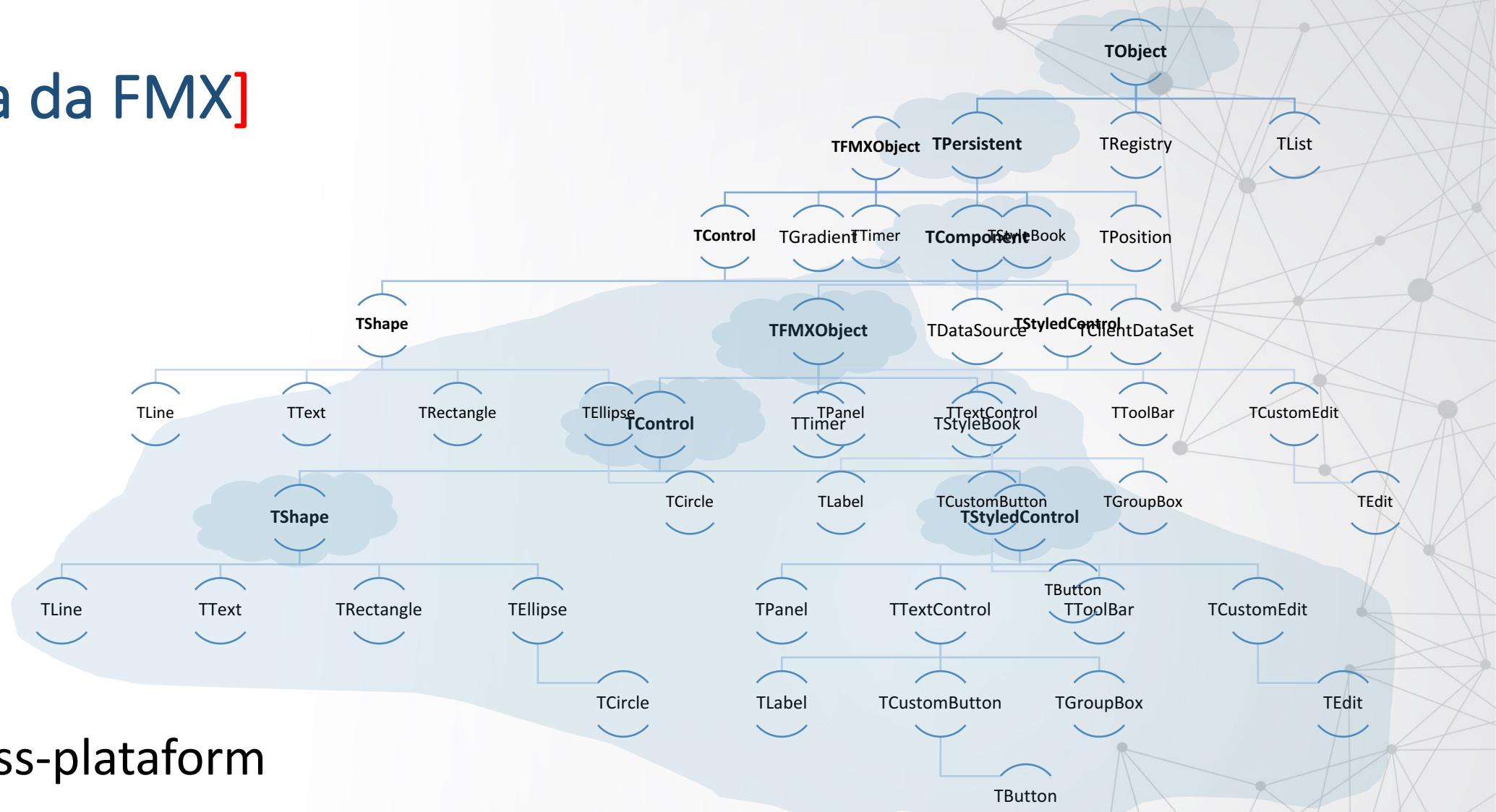


[A Estrutura da VCL]

- A VCL tem como foco ambiente Windows
- Sua estrutura é dividida com base nos recursos desse S.O.



[A Estrutura da FMX]



- A FMX: cross-plataform
 - Sua estrutura é dividida em controles primitivos e controles estilizados



Interfaces da FMX

[Principais Interfaces da FMX]

- IFreeNotification
- IFreeNotificationBehavior
- Icontrol
- ITextSettings
- IContainerObject



[IFreeNotification]

```
IFreeNotification = interface  
['{FEB50EAF-A3B9-4b37-8EDB-1EF9EE2F22D4}']  
procedure FreeNotification(AObject: TObject);  
end;
```

[IFreeNotificationBehavior]

```
IFreeNotificationBehavior = interface  
['{83F052C5-8696-4AFA-88F5-DCDFEF005480}']  
procedure AddFreeNotify(const AObject:  
IFreeNotification);  
procedure RemoveFreeNotify(const AObject:  
IFreeNotification);  
end;
```

[IControl]

```
IControl =  
interface(IFreeNotificationBehavior)  
  ['{7318D022-D048-49DE-BF55-C5C36A2AD1AC}']  
  
  function GetObject: TFmxObject;  
  procedure SetFocus;  
  function GetIsFocused: Boolean;  
  function GetCanFocus: Boolean;  
  function GetCanParentFocus: Boolean;  
  function GetEnabled: Boolean;  
  function GetAbsoluteEnabled: Boolean;  
  function GetPopupMenu: TCustomPopupMenu;  
  function EnterChildren(AObject: IControl): Boolean;  
  function ExitChildren(AObject: IControl): Boolean;  
  procedure DoEnter;  
  procedure DoExit;  
  procedure DoActivate;  
  procedure DoDeactivate;  
  procedure DoMouseEnter;  
  procedure DoMouseLeave;  
  function ShowContextMenu(const ScreenPosition: TPointF): Boolean;  
  
  function ScreenToLocal(P: TPointF): TPointF;  
  function LocalToScreen(P: TPointF): TPointF;  
  function ObjectAtPoint(P: TPointF): IControl;  
  procedure MouseDown(Button: TMouseButton; Shift: TShiftState; X, Y: Single);  
  procedureMouseMove(Shift: TShiftState; X, Y: Single);  
  procedure MouseUp(Button: TMouseButton; Shift: TShiftState; X, Y: Single);  
  procedure MouseWheel(Shift: TShiftState; WheelDelta: Integer; var Handled: Boolean);  
  procedure MouseClick(Button: TMouseButton; Shift: TShiftState; X, Y: Single);  
  procedure KeyDown(var Key: Word; var KeyChar: WideChar; Shift: TShiftState);  
  procedure KeyUp(var Key: Word; var KeyChar: WideChar; Shift: TShiftState);  
  procedure Tap(const Point: TPointF);  
  procedure DialogKey(var Key: Word; Shift: TShiftState);  
  procedure AfterDialogKey(var Key: Word; Shift: TShiftState);  
  function FindTarget(P: TPointF; const Data: TDragObject): IControl;  
  
  procedure DragEnter(const Data: TDragObject; const Point: TPointF);  
  procedure DragOver(const Data: TDragObject; const Point: TPointF; var Operation: TDragOperation);  
  procedure DragDrop(const Data: TDragObject; const Point: TPointF);  
  procedure DragLeave;  
  procedure DragEnd;  
  function CheckForAllowFocus: Boolean;  
  procedure Repaint;  
  function GetDragMode: TDragMode;  
  procedure SetDragMode(const ADragMode: TDragMode);  
  procedure BeginAutoDrag;  
  function GetParent: TFmxObject;  
  function GetLocked: Boolean;  
  function GetVisible: Boolean;  
  procedure SetVisible(const Value: Boolean);  
  function GetHitTest: Boolean;  
  function GetCursor: TCursor;  
  function GetInheritedCursor: TCursor;  
  function GetDesignInteractive: Boolean;  
  function GetAcceptsControls: Boolean;  
  
  procedure SetAcceptsControls(const Value: Boolean);  
  procedure BeginUpdate;  
  procedure EndUpdate;  
  function GetTabStopController: ITabStopController;  
  function GetTabStop: Boolean;  
  procedure SetTabStop(const TabStop: Boolean);  
  /// <summary>This method returns true if the control has an available hint to display.</summary>  
  function HasHint: Boolean;  
  /// <summary>If HasHint is true, this method is invoked in order to know if the control has an available  
  /// string to show as hint.</summary>  
  function GetHintString: string;  
  /// <summary>If HasHint is true, this method is invoked in order to know if the control has a custom hint  
  /// object to manage the hint display. This usually returns an instance of THint to allow the form to manage  
  /// it.</summary>  
  function GetHintObject: TObject;  
  { access }  
  
  property AbsoluteEnabled: Boolean read GetAbsoluteEnabled;  
  property Cursor: TCursor read GetCursor;  
  property InheritedCursor: TCursor read GetInheritedCursor;  
  property DragMode: TDragMode read GetDragMode write SetDragMode;  
  property DesignInteractive: Boolean read GetDesignInteractive;  
  property Enabled: Boolean read GetEnabled;  
  property Parent: TFmxObject read GetParent;  
  property Locked: Boolean read GetLocked;  
  property HitTest: Boolean read GetHitTest;  
  property PopupMenu: TCustomPopupMenu read GetPopupMenu;  
  property Visible: Boolean read GetVisible write SetVisible;  
  property AcceptsControls: Boolean read GetAcceptsControls write SetAcceptsControls;  
  property IsFocused: Boolean read GetIsFocused;  
  property TabStop: Boolean read GetTabStop write SetTabStop;  
end;
```

[ITextSettings]

```
ITextSettings = interface  
  ['{FD99635D-D8DB-4E26-B36F-  
  97D3AABBCCB3}']  
    function GetDefaultTextSettings:  
      TTextSettings;  
    function GetTextSettings:  
      TTextSettings;  
    procedure SetTextSettings(const  
      Value: TTextSettings);  
    function GetResultingTextSettings:  
      TTextSettings;  
    function GetStyledSettings:  
      TStyledSettings;  
    procedure SetStyledSettings(const  
      Value: TStyledSettings);
```

```
    property DefaultTextSettings:  
      TTextSettings read  
      GetDefaultTextSettings;  
    property TextSettings: TTextSettings  
      read GetTextSettings write  
      SetTextSettings;  
    property ResultingTextSettings:  
      TTextSettings read  
      GetResultingTextSettings;  
    property StyledSettings:  
      TStyledSettings read  
      GetStyledSettings write  
      SetStyledSettings;  
  end;
```

[IContainerObject]

```
IContainerObject = interface  
['{DE635E60-CB00-4741-92BB-3B8F1F29A67C}']  
function GetContainerWidth: Single;  
function GetContainerHeight: Single;  
property ContainerWidth: single read  
GetContainerWidth;  
property ContainerHeight: single read  
GetContainerHeight;  
end;
```

[Outras Interfaces da FMX]

- ICaption
- IIsChecked
- IGroupName
- IKeyShortcut
- IGlyph
- IGestureControl
- IRotatedControl
- IAlignableObject



RTL

[RTL]

- Contém rotinas globais e classes utilitárias que servem principalmente de apoio para componentes da VCL e FMX
- Boa parte da RTL se concentra na unit System (ou escopo System)
 - System.IOUtils
 - System.Generics
 - System.Win
 - System.Rtti
 - System.JSON
 - System.SysUtils
 - System.DateUtils
 - System.ZIP

[RTL]

- Além das rotinas globais e classes utilitárias eis algumas features sob a responsabilidade da RTL
 - Beacon
 - Bindings
 - Bluetooth
 - Generics
 - Tethering







Dúvidas



Animações, Efeitos, Transição e 3D

[Rodrigo Mourão]

[Animações, Efeitos, Transição e 3D]

- Apresentação
- Animações
- Efeitos
- Transição
- Efeitos Animados
- 3D



[Rodrigo Mourão]

- Gestor de TI e Empreendedor Digital
- Minha missão é impactar vidas através do conhecimento
- Nos últimos 8 anos deixei o medo me impedir de fazer o que mais amo profissionalmente: ensinar
- E desde outubro/15, corro atrás para tirar o atraso
- Apaixonado pelo Delphi
 - fb.com/blogrodrigomourao
 - youtube.com/thermfactory
 - telegram.me/rmfactory
 - @rodrigocmourao



Animações

[Classe TAnimation]

- Uma animação muda o valor de uma propriedade em função do tempo
- TAnimation é a classe base para todos os tipos de animações, porém nunca deve ser usada diretamente
- Propriedades importantes
 - Interpolation, AutoReverse, Loop, Delay, Duration, Inverse, Trigger
 - **PropertyName**
 - **StartValue**
 - **StopValue**

[Descendentes de TAnimation]

- Os descendentes de TAnimation estão divididos em 3 categorias:
- Interpolação de um valor inicial para um valor final
 - **TIntAnimation** - altera o valor de qualquer propriedade do tipo inteiro
 - **TFloatAnimation** - altera o valor de qualquer propriedade do tipo real
 - **TRectAnimation** - altera o valor das coordenadas de qualquer propriedade do tipo TBounds
 - **TColorAnimation** - altera o valor (String ou Integer) de qualquer propriedade do tipo color inclusive TAlphaColor que é um cardinal
 - **TGradientAnimation** - altera o efeito gradient de cada um dos pontos que define o efeito gradiente
 - **TBitmapAnimation** - faz a transição de uma imagem inicial para uma imagem final, simulando um efeito fade através da opacidade
- Interpolação através de uma lista de valores
 - **TFloatKeyAnimation** - transição de uma lista de valores numéricos
 - **TColorKeyAnimation** - transição de uma lista de cores
 - **TPathAnimation** - transição de uma lista de posições de um objeto 2D
- Lista sem interpolação
 - **TBitmapListAnimation** – simula um slide show com todas as imagens combinadas horizontalmente como uma única imagem

[Recursos da Animação]

- Interpolação e Tipos
- Gatilhos de Start e Stop
- Loop e Inversão de Efeito
- Path Animation





Efeitos e Transições

[Efeitos]

- Mecanismos de rerenderização de imagens baseado em GPU
- Facilmente aplicado a qualquer controle do FMX, simples como o Delphi deve ser
- Visível em design-time e em run-time
- Cada efeito possui propriedades inerentes a sua função
- Os efeitos são baseados em filtros que atuam sobre o pixel
- Não é possível aplicar dois efeitos em um mesmo controle
- Os efeitos são divididos em 5 categorias em função do tipos de filtro

[Categorias de Efeitos]





3D



Mão na Massa



Dúvidas

@ rodrigomourao@rodrigomourao.com.br

▶ <http://youtube.com/thermfactory>

f <http://fb.com/blogrodrigomourao>

[OBRIGADO]

Embarcadero conference



Embarcadero conference

embarcadero®