MISAO with WPF

JZ5

2009/2/7

自己紹介

- ＪＺ５（松江祐輔）
- プログラマーですか？
  - –違います。Verilog書いてます。
- @jz5 Twitter
- katamari.jp
- katamari.wankuma.com

Agenda

ニコニコメソッド＆
Katamari.MISAO

WPFプログラミング

What's ニコニコメソッド

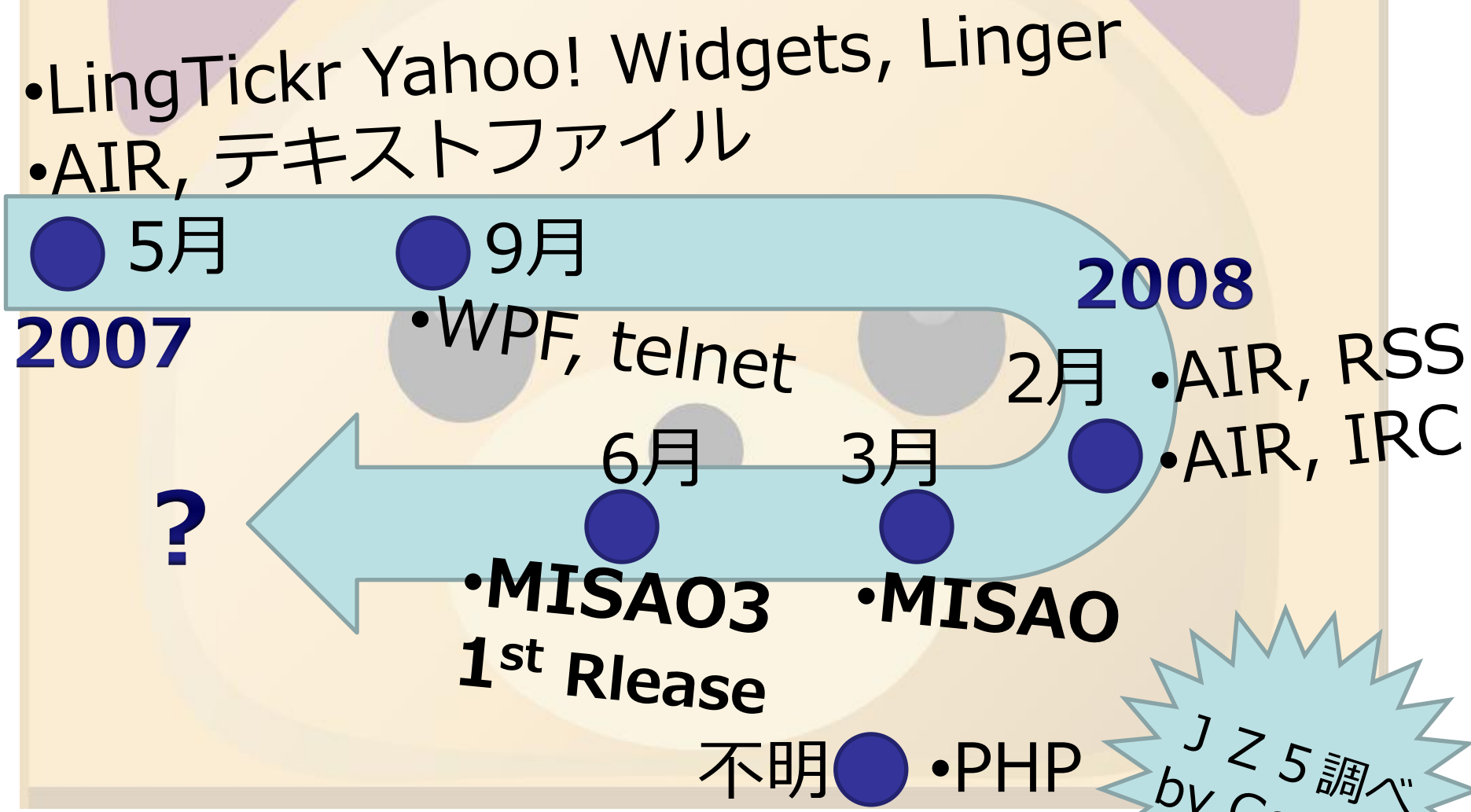
2007/4/25 ニコニコ動画勉強会

- プレゼン中に参加者がケータイからコメントし**スライド上に**ニコニコ動画風に**コメントが流れる**ことをしたみたい。

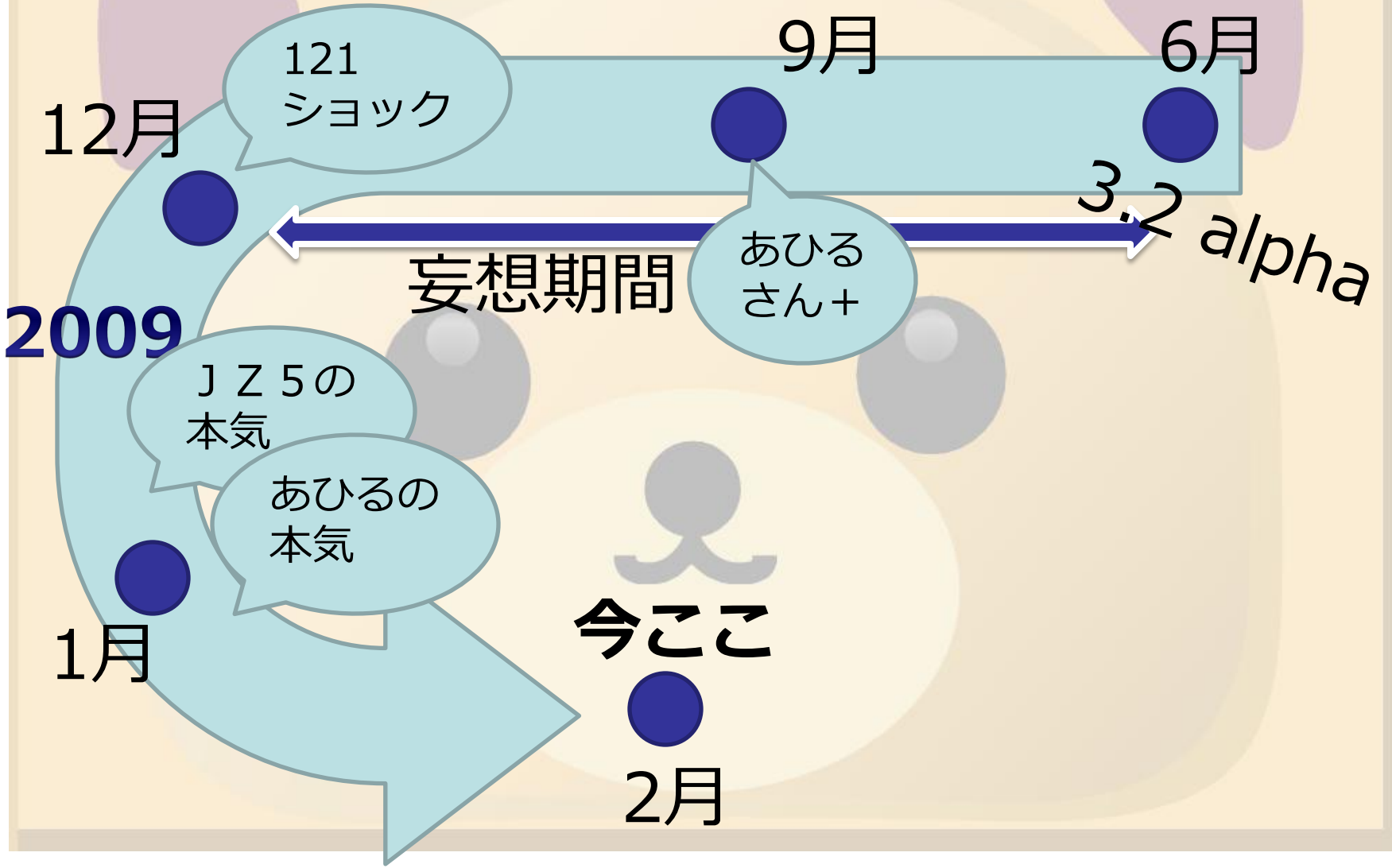
ニコニコ動画勉強会に行ってきました（TAKESAKO @ Yet another Cybozu Labs）

- ニコニコプレゼンや
  ニコニコメソッドと呼ばれる。

History of ニコニコメソッドツール



MISAO after first release



MISAOの外面的な特徴

- メッセージソース
  - Ustream（実質これだけ）
  - Twitter
  - Live Messenger
- （たぶん一番）ニコっぽい
- わんくま勉強会
- 重い

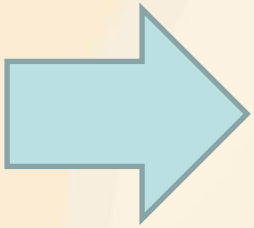
MISAOの内面的な特徴

- WPF
- System.AddIn なんでもアドイン
- 隠された拡張性
- キャラクター志向モデリングではない
- Etc.

実演

Why WPF？

- アニメーションを実装したくなかった



PowerPointのアドイン

無理！？

WPF

- 新しいWPF+VB.NET

Programming Menu

アニメーション

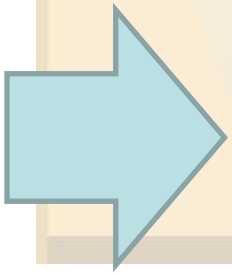
透明ウィンドウ

アプリケーション（おまけ）

WPFのアニメーション

- WPFには簡単に使えるアニメ機能がある
- プロパティを変化させてアニメーション
- 条件（とりあえずどうでもいい）
  - 依存関係プロパティ
  - DependencyObjectクラス継承
  - IAnimatbaleインタフェースを実装

  } したクラスに属する
  - 互換性のあるアニメ種類が利用できる状態



ウィンドウにのるコントロールならなんでもアニメ可

アニメーション方法

②終了値の指定

①開始値の指定

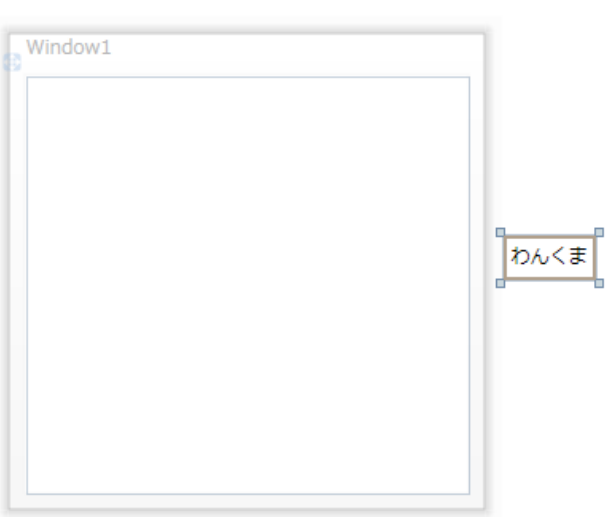
③アニメ時間の指定

④アニメ開始メソッドの
呼び出し

ウィンドウ作成（10行）

```
<Window x:Class="Window1"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    Title="Window1" Height="300" Width="300">
    <Canvas Name="KumaCanvas">
        <Label Content="わんくま"
               Name="KumaLabel"
               Canvas.Left="300" Canvas.Top="100" />
    </Canvas>
</Window>
```

WPFアプリケーションを作成してここだけ変更

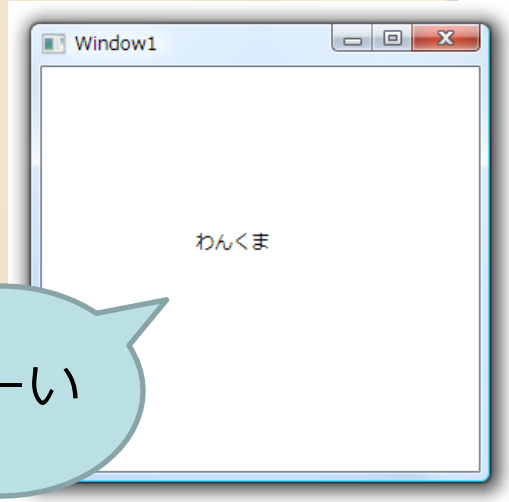


アニメーション（10行）

```
Imports System.Windows.Media.Animation
Class Window1
    Private Sub Window1_Loaded() Handles Me.Loaded
        Dim a = New DoubleAnimation With { _
            .From = Canvas.GetLeft(KumaLabel), _
            .To = -KumaLabel.ActualWidth, _
            .Duration = New Duration(TimeSpan.FromSeconds(10))}
        KumaLabel.BeginAnimation(Canvas.LeftProperty, a)
    End Sub
End Class
```

コード
ビハインド

実行

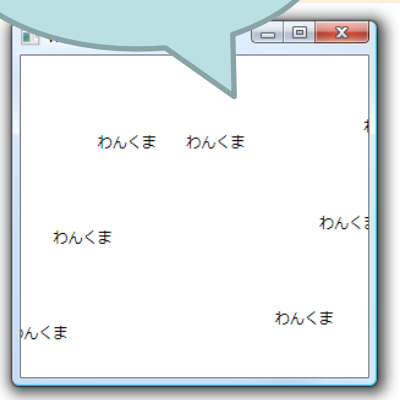


動的にラベル生成（20行ぐらい）

```
Private Timer As New System.Windows.Threading.DispatcherTimer
Private Sub Window1_Loaded() Handles Me.Loaded
    AddHandler Timer.Tick, AddressOf Timer_Tick
    Timer.Interval = New TimeSpan(0, 0, 1)
    Timer.Start()
End Sub
```

```
Private Sub Timer_Tick()
    Dim l = New Label
    l.Content = "わんくま"
    KumaCanvas.Children.Add(l) ' Canvas追加
    KumaCanvas.UpdateLayout()
    Canvas.SetLeft(l, Me.Width) ' 座標設定
    Canvas.SetTop(l, New Random().Next(Me.Height))
    Dim a = New DoubleAnimation With { _
        .From = Canvas.GetLeft(l), _
        .To = -l.ActualWidth, _
        .Duration = New Duration(TimeSpan.FromSeconds(10))}
    l.BeginAnimation(Canvas.LeftProperty, a)
End Sub
```

わらわら

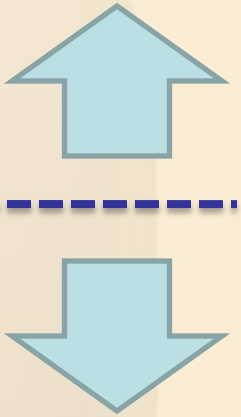


破棄…

必要なウィンドウ

- 透明なウィンドウ（枠なし）
- タスクバー非表示
- 常に最前面
- 非アクティブ
- クリック透過
- Alt+Tab切り替えで非表示

簡単



Win32

もろもろプロパティ

```
<Window x:Class="Window1"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    Title="Window1" Height="300" Width="300"
        Background="Transparent"
        AllowsTransparency="True"
        WindowStyle="None"
        ShowInTaskbar="False"
        Topmost="True"
        ShowActivated="False"
    >
```

セットで

3.5 SP1

すこし脱線

```
<Window x:Class="Window1"
 （もろもろプロパティ）
>
   <Grid>
      <Image Source="http://www.wankuma.com/images/logo3.png"
           MouseLeftButtonDown="Image_MouseLeftButtonDown"/>
   </Grid>
</Window>
```

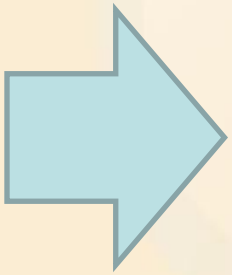
&

```
Private Sub Image_MouseLeftButtonDown()
    DragMove()
End Sub
```

！？

クリックを透過するには？

- Background=Transparentだけではウィンドウ上のコントロールがクリックできる。
- たぶんWPFだけじゃできないので……。



Windows API（Win32 API）
SetWindowLong関数

Win32 APIを使うには

- ウィンドウハンドルの取得

これまで（Windows.Forms）：　Me.Handle

WPFアプリでの方法:

```
Dim handle = New
System.Windows.Interop.
WindowInteropHelper(Me).
Handle
```

とりあえずWindow1_Loaded内に入れよう

コンストラクタ内では取得できない

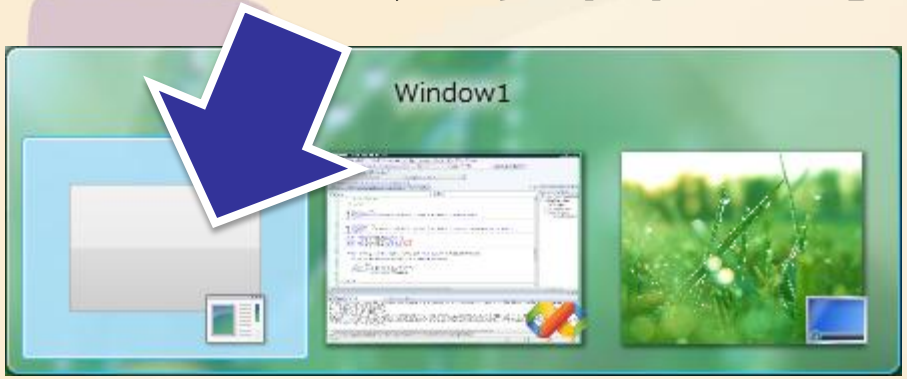
SetWindowLongでクリック透過

- 拡張ウィンドウスタイル(GWL_EXSTYLE)ってのを書き換えます。
- スタイル**WS_EX_TRANSPARENT**を付ける。

```
Dim style = GetWindowLong(handle, GWL_EXSTYLE)
SetWindowLong(handle, GWL_EXSTYLE, _
                      style Or  WS_EX_TRANSPARENT)
```

クリックが透過するのはWS_EX_LAYEREDスタイルも付いているときだけ！
透明ウィンドウにはWS_EX_LAYEREDスタイルは付いてる。

タスク切り替え時 非表示にする



SetWindowLongを使って
拡張ウィンドウスタイルから
（WS_EX_APPWINDOWを削除）
WS_EX_TOOLWINDOWを追加

参考： 非アクティブ

```
' 非アクティブ（参考）
SetWindowPos(handle, _
        New IntPtr(HWND_TOPMOST), _
        0, 0, 0, 0, _
        SWP_NOMOVE Or SWP_NOSIZE Or _
        SWP_NOACTIVATE)
```

まとめ

- ニコメソッドツール＆MISAO
- アニメ簡単
- 凝ったことをしだすとWin32…

Enjoy WPF & Presentation