

MISAO with WPF

J-Z 5

2008/6/7



# Agenda

自己紹介

ニコニコメソッド

MISAO

MISAO Inside  
(WPF)



梨ズームも  
覚えて帰ってね

# 自己紹介

- J Z 5 (松江祐輔)
- @わんくま同盟
- ハードウェア技術者
- 使ってみよう！ Windows Live SDK/API (gihyo.jp)
- katamari.jp 塊 SOUL
- katamari.wankuma.com

W P F  
初心者

# 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**ことをしてみた。



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

2007/5/1

- LingrTickr - 誰でもニコニコメソッドプレゼン (nakatani @ cybozu labs)
  - Yahoo! Widgets, Lingr

2007/5/1

- AIR でニコニコメソッドプレゼン (てっく煮ブログ)
  - Adobe AIR, テキストファイル

J Z 5 調べ  
by Google

## 歴史その2

2007/9/9

- WPF でニコニコメソッド (ZOETROPE  
の日記)
  - WPF, telnet

2008/2/2

- [AIR][ActionScript]AIR でニコニコ動画  
風 RSS リーダー作った (プレゼン向け)  
(public static void main)



# 歴史その3

2008/2/23

- 第2回1000speakers DDDの概要とその可能性suztomo
  - AIR, IRC(USTREAM.TV)

2007/11? 08/2?

- 萩野・服部研究室 ニコニコプレゼン
  - スライドも含めたPHPによるシステム

# Birth of MISAO

- 2008/3/15 東京勉強会
  - USTREAM.TV 配信
- 2008/3/29 大阪勉強会
  - WPF & IRC (USTREAM.TV)

6/7

Release!

3/15

3/29





# MISAO DEMO



IRC

Twitter (予定)

USTREAM.TV

RSS/Atom Feeds (未定)

Live Messenger

配布場所 予定地

[katamari.jp/soulware/](http://katamari.jp/soulware/)



# MISAO Inside

透明ウィンドウ

アニメーション

Thread

?

ウィンドウを透明にするには？

WindowのXAML

**Background="Transparent"**



残念な結果に

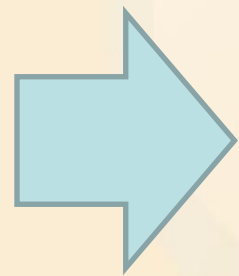
**AllowsTransparency="True"**

**WindowStyle="None"**

セットで！

# 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:

**`System.Windows.Interop.  
WindowInteropHelper(Me).  
Handle`**

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

- 以上でOK? まだです。

これ要らない



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

常に最前面に**非アクティブ**で表示  
最前面だけならXAMLでOK

**Topmost="True"**

非アクティブで表示するには、  
やっぱりWin32 API

```
SetWindowPos(handle, _  
    CType(HWND_TOPMOST, IntPtr), _  
    0, 0, 0, 0, _  
    SWP_NOMOVE Or SWP_NOSIZE Or _  
    SWP_NOACTIVATE)
```





ウィンドウ表示時に非アクティブ

- フックを使うとできます。

-SetWindowsHookEx

-UnhookWindowsHookEx

-CallNextHookEx

※最初のウィンドウは無理

参考: 「WPF Tips and Tricks: Window.Show()  
Without Activating The Window」  
(IRhetic)

おわりに

- ShowInTaskbarも忘れずに。

**ShowInTaskbar="False"**

# 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

- 依存関係プロパティ

- DependencyObjectクラス継承

- IAnimatableインタフェースを実装

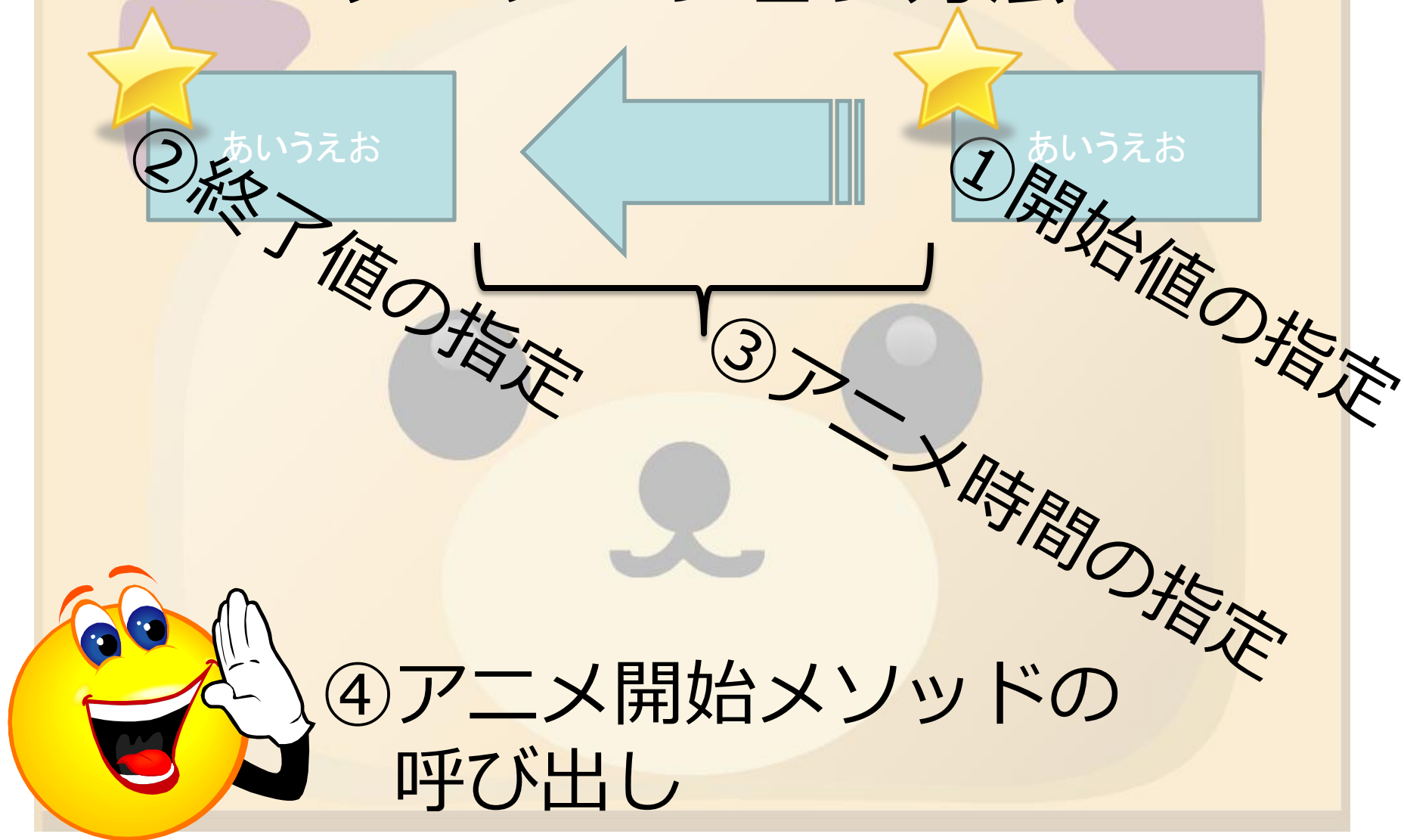
} したクラス  
に属する

- 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 ← 結論

# アニメーション方法



# 資料 1

```
<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">
  <Grid>
    <Label x:Name="BarLabel" Width="300"
      Background="Red" HorizontalAlignment="Left"
      MouseDown="BarLabel_MouseDown" />
  </Grid>
</Window>
```

```
Private Sub BarLabel_MouseDown()
  Dim a = New DoubleAnimation
  a.From = BarLabel.Width
  a.To = 0
  a.Duration = New Duration(TimeSpan.FromSeconds(10))
  BarLabel.BeginAnimation(Label.WidthProperty, a)
End Sub
```



# Storyboardを使う

- 普通は（？）Storyboardを使う
- 複数のプロパティアニメにも使える
- あとXAMLにも書ける

```
Private Sub BarLabel_MouseDown()
```

```
Dim a = New DoubleAnimation
```

```
a.From = BarLabel.Width
```

```
a.To = 0
```

```
a.Duration = New Duration(TimeSpan.FromSeconds(10))
```

```
Storyboard.SetTargetName(a, "BarLabel")
```

```
Storyboard.SetTargetProperty(a, New PropertyPath(Label.WidthProperty))
```

```
Dim s = New Storyboard
```

```
s.Children.Add(a)
```

```
BarLabel.BeginStoryboard(s)
```

```
End Sub
```

資料 2



# Thread処理 UIの操作

- UIの操作はUIのスレッドから行う！
- WPFではUIスレッド以外から操作すると例外をスロー

これまで


(System.Windows.Forms) :

**Control.Invokeメソッド**とか

# WPFでThread

WPFの方法:

- Dispatcherオブジェクトを使う。
- **Dispatcher.Invoke(DispatcherPriority, Delegate)**



優先順位が  
指定できる

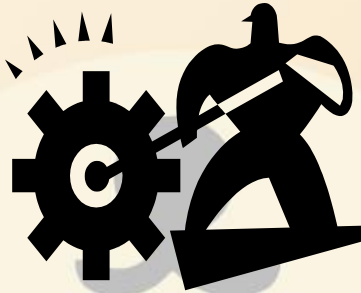


# 私がしばしば書くコード

```
Private Sub MessageReceived(ByVal sender As Object, _  
                             ByVal e As EventArgs)  
    If Dispatcher.Thread IsNot System.Threading.Thread.CurrentThread Then  
        Dispatcher.Invoke(Windows.Threading.DispatcherPriority.Normal, _  
                            New Action(Of Object, EventArgs)(AddressOf  
MessageReceived), sender, New Object() {e})  
  
        Exit Sub  
    End If  
    'ここにUI 操作  
End Sub
```

# 基本はBackgroundWorker

- .NET Framework 2.0で追加されたクラス
- DoWorkイベント: 別スレッドで動く
- ProgressChangedイベント: UIスレッドで動く
- RunWorkerCompletedイベント: UIスレッドで動く



- タイマーはDispatcherTimerを使おう。

参考: 「Dispatcher を使用して応答性の高いアプリケーションを構築する」 (MSDNマガジン)

# その他の話題

- 枠付きの文字
  - 「方法：中抜き of 文字列を作成する」  
(MSDN Library)
  - 影付きは簡単: ShadowDepth プロパティ
- メッセージソース アドイン
  - System.AddIn
- IRC
- Live Messenger
  - MSNP, パケットキャプチャ

# まとめ

- Windows Liveの記事書いてます。
- ニコメソッドツールを紹介しました。
- MISAOはkatamari.jpから。
- 透明ウィンドウはいろいろ考えるとWPF & .NETだけでは難しい。
- アニメは簡単。
- Threadはこれまでと同様な感じ。

Enjoy WPF & Presentation