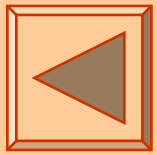
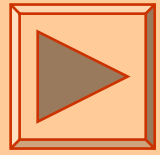
松山大学 経営学部

# 情報処理論（応用）



## 第16回　グラフィックス処理（5）



講師　檀 裕也

http://www.cc.matsuyama-u.ac.jp/~dan/application/

2006年11月21日

# 出席確認

- 出席確認フォームから学籍番号および氏名を送信せよ。

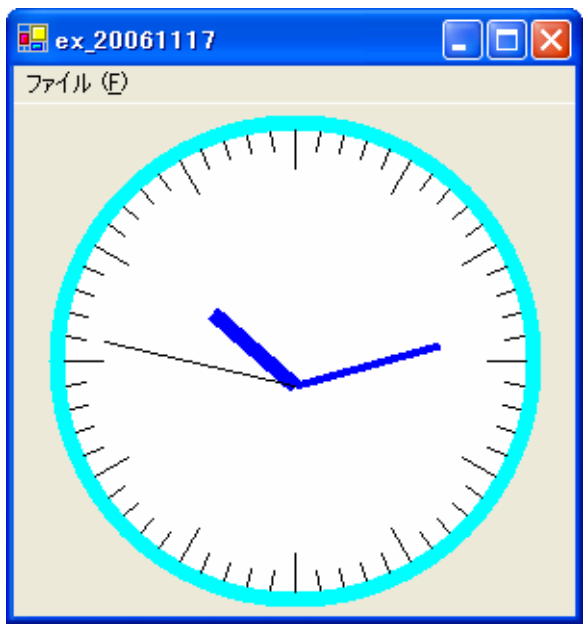
出席確認フォーム

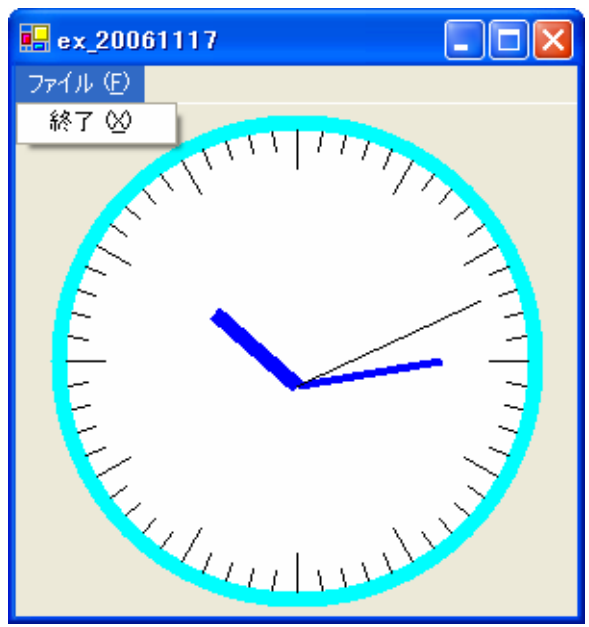
http://www.cc.matsuyama-u.ac.jp/~dan/application/attendance.html

# 前回の課題

アナログ時計プログラム（ex_20061117）

- アナログ時計の時針、分針および秒針を実装し、外観をデザインせよ。



# 解答例（コード）

- 使用する変数を宣言し、現在時刻を取得する

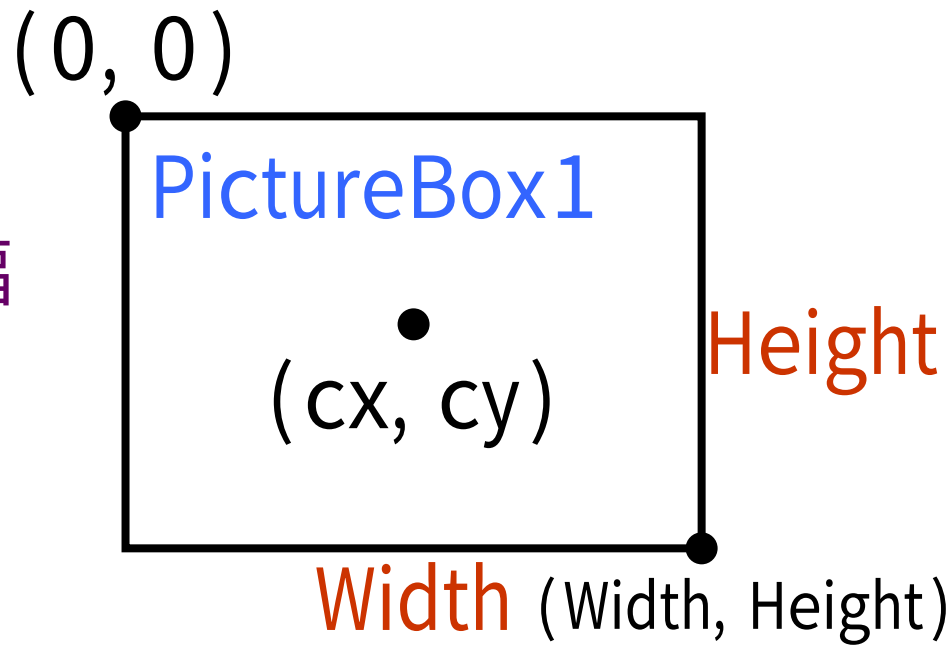
```
Public Class Form1
    Inherits System.Windows.Forms.Form

 Windows フォーム デザイナで生成されたコード 

    Private Sub PictureBox1_Paint(ByVal sender As Object, ByVal e As System.Windows.Forms.PaintEventArgs) Handles PictureBox1.Paint
        Dim t As DateTime
        Dim h, m, s As Integer
        Dim cx, cy As Integer
        Dim theta As Double
        Dim x1, y1, x2, y2 As Integer
        Dim x, y As Integer
        Dim i As Integer

        ' 現在時刻を取得する
        t = DateTime.Now
        h = t.Hour
        m = t.Minute
        s = t.Second

        ' 中心座標の取得
        cx = PictureBox1.Width / 2
        cy = PictureBox1.Height / 2

        ' 画面を消去
        e.Graphics.Clear(PictureBox1.BackColor)

        ' 時計の外枠を描画
        e.Graphics.FillEllipse(New SolidBrush(Color.Aqua), cx - 128, cy - 128, 256, 256)
        e.Graphics.FillEllipse(New SolidBrush(Color.White), cx - 120, cy - 120, 240, 240)
```

# コードの解説

- PictureBox の幅と高さから中心座標を求める

```
cx = PictureBox1.Width / 2
cy = PictureBox1.Height / 2
```

  - Width プロパティ
    オブジェクトの横幅
  - Height プロパティ
    オブジェクトの高さ



# 解答例（コードつづき）

```
        ' 時計の目盛りを描画
        For i = 0 To 59
            theta = 2 * Math.PI * i / 60
            If i Mod 5 = 0 Then
                x1 = cx + 100 * Math.Cos(theta)
                y1 = cy + 100 * Math.Sin(theta)
            Else
                x1 = cx + 110 * Math.Cos(theta)
                y1 = cy + 110 * Math.Sin(theta)
            End If
            x2 = cx + 120 * Math.Cos(theta)
            y2 = cy + 120 * Math.Sin(theta)
            e.Graphics.DrawLine(Pens.Black, x1, y1, x2, y2)
        Next

        ' 時針の表示
        theta = 2 * Math.PI * h / 12 - Math.PI / 2
        x = cx + 50 * Math.Cos(theta)
        y = cy + 50 * Math.Sin(theta)
        e.Graphics.DrawLine(New Pen(Color.Blue, 8), cx, cy, x, y)

        ' 分針の表示
        theta = 2 * Math.PI * m / 60 - Math.PI / 2
        x = cx + 75 * Math.Cos(theta)
        y = cy + 75 * Math.Sin(theta)
        e.Graphics.DrawLine(New Pen(Color.Blue, 4), cx, cy, x, y)

        ' 秒針の表示
        theta = 2 * Math.PI * s / 60 - Math.PI / 2
        x = cx + 100 * Math.Cos(theta)
        y = cy + 100 * Math.Sin(theta)
        e.Graphics.DrawLine(New Pen(Color.Black, 1), cx, cy, x, y)

    End Sub

    Private Sub MenuItem2_Click(ByVal sender As Object, ByVal e As System.EventArgs) Handles MenuItem2.Click
        Timer1.Stop()
        Me.Dispose()
    End Sub
```

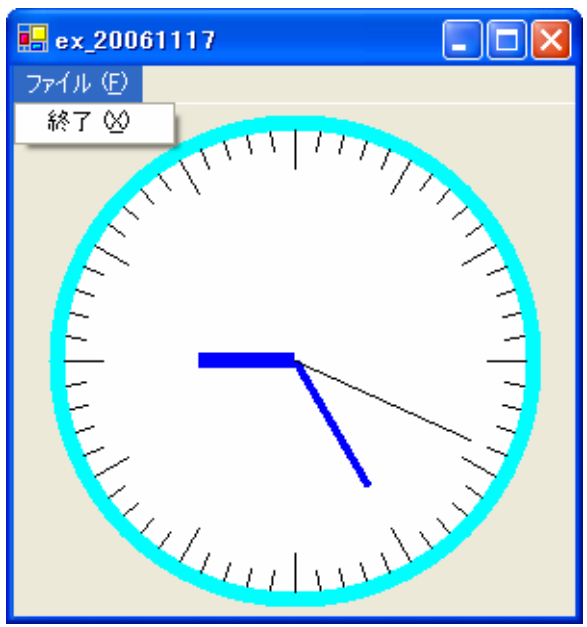
# 今回の予定

- アニメーション
  - アナログ時計の完成
  - 転がるボールのアニメーション

- 到達目標
  - Windows における GDI+ の機能と Timer コンポーネントを使って、アニメーションを実現できる。

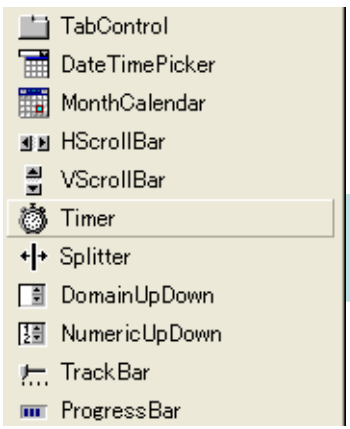
# 例題14

- 前回の課題プロジェクトに、アニメーションの処理を加えて、時計の針がリアルタイムに動くように機能拡張する。



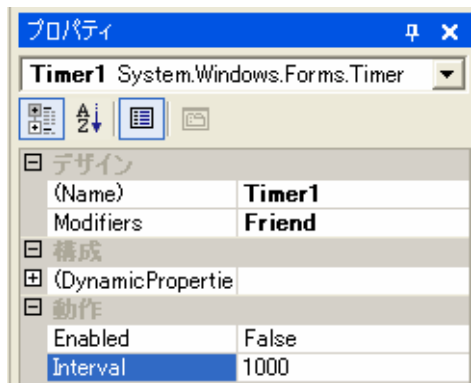
# Timer コンポーネントの追加

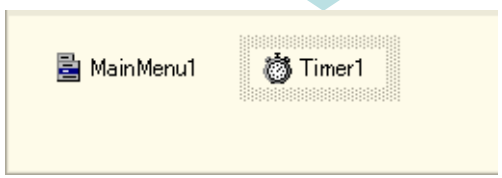
- ツールボックスから Timer を選択し、フォームに追加する。(画面下部のトレイに表示される)




①Timerを追加する

②Timer1の Interval プロパティの値を 1000 に設定する




③ダブルクリックしてコード記述へ

# Timer コンポーネントの機能

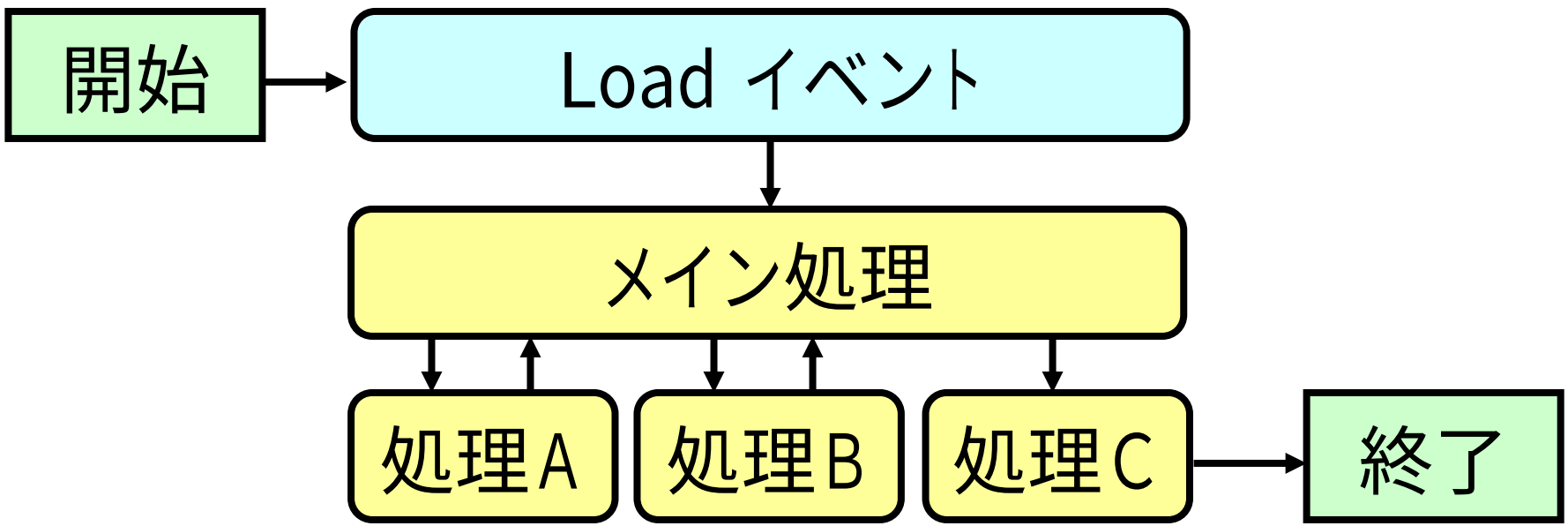
- Interval プロパティによって指定された時間の間隔でイベントハンドラを呼び出す。
  - 単位はミリ秒（1000ミリ秒＝１秒）
  - Timer_Tick プロシージャにイベント処理を記述
- 呼び出しを開始するとき Start メソッドを使う。

  Timer1.Start( )
- 終了時に Stop メソッドを使う。

  Timer1.Stop( )

# フォームの Load イベント

- プログラム起動時に発生するイベント
  - 通常、初期設定などのコードを記述する
  - デザイン画面で、フォームをダブルクリックして Form1_Load のプロシージャ内に記述する



# コードの追加

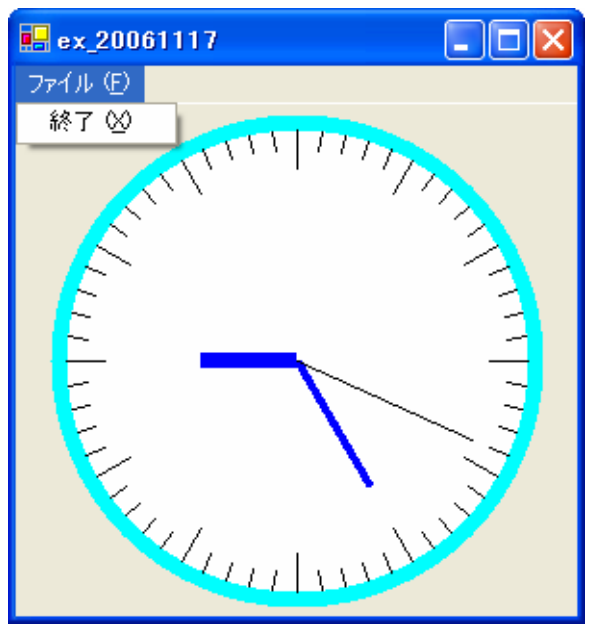
- プログラムに次のコードを追加する
  - プログラム起動時に Timer を開始する
  - １秒に１回の頻度で PictureBox を書き換える
  - プログラム終了時に Timer を終了する

PictureBox1.Refresh()

```
        ' 分針の表示
        theta = 2 * Math.PI * m / 60 - Math.PI / 2
        x = cx + 75 * Math.Cos(theta)
        y = cy + 75 * Math.Sin(theta)
        e.Graphics.DrawLine(New Pen(Color.Blue, 4), cx, cy, x, y)

        ' 秒針の表示
        theta = 2 * Math.PI * s / 60 - Math.PI / 2
        x = cx + 100 * Math.Cos(theta)
        y = cy + 100 * Math.Sin(theta)
        e.Graphics.DrawLine(New Pen(Color.Black, 1), cx, cy, x, y)

    End Sub

    Private Sub MenuItem2_Click(ByVal sender As Object, ByVal e As System.EventArgs) Handles MenuItem2.Click
        Timer1.Stop()
        Me.Dispose()
    End Sub

    Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load
        Timer1.Start()
    End Sub

    Private Sub Timer1_Tick(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Timer1.Tick
        PictureBox1.Refresh()
    End Sub
End Class
```

# 実行（デバッグ）

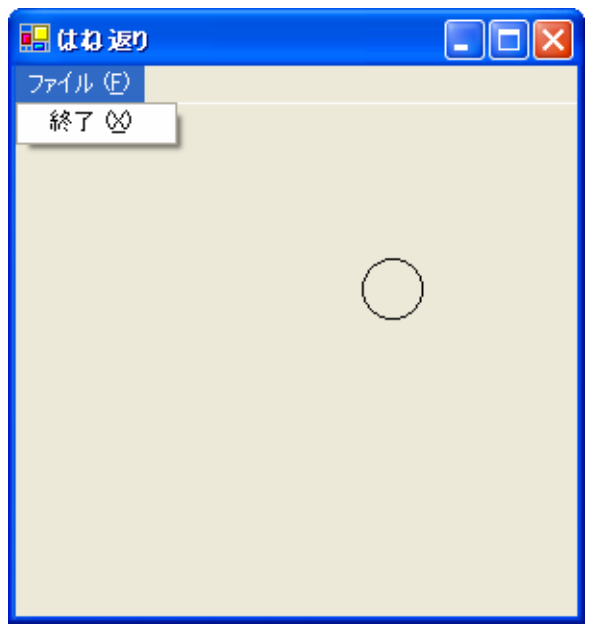

- [デバッグ]→[開始]でプログラムを実行する
- ツールバーの開始ボタンをクリックしてもよい

# 例題15

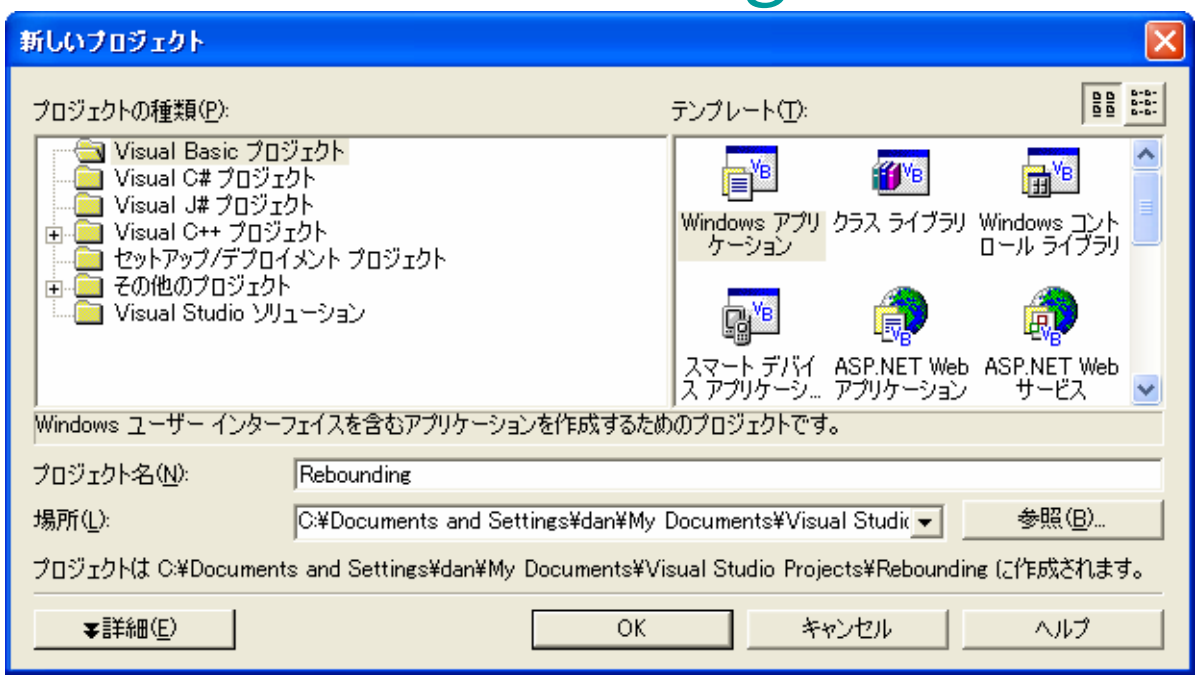
- 壁に当たるとはね返る図形のアニメーション

- 手順:
  - プロジェクトの新規作成
    (Rebounding)
  - 画面レイアウト
  - イベントハンドラの記述
  - 実行および動作確認

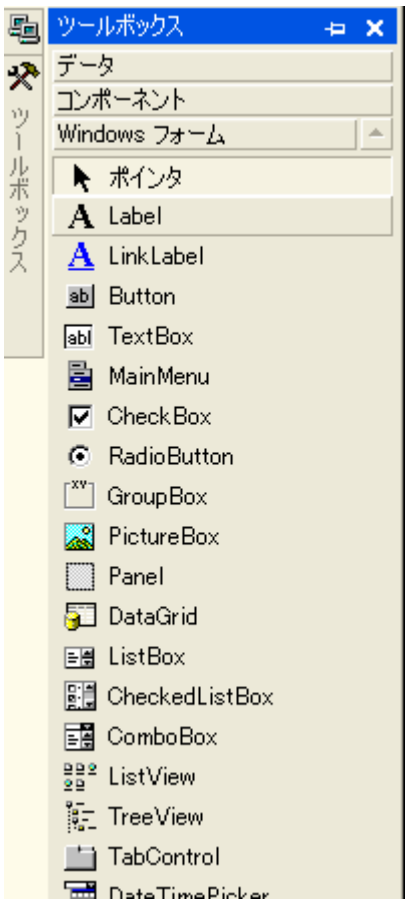


# 新しいプロジェクト

- プロジェクトの種類「Visual Basicプロジェクト」
- テンプレート「Windowsアプリケーション」
- プロジェクト名「Rebounding」



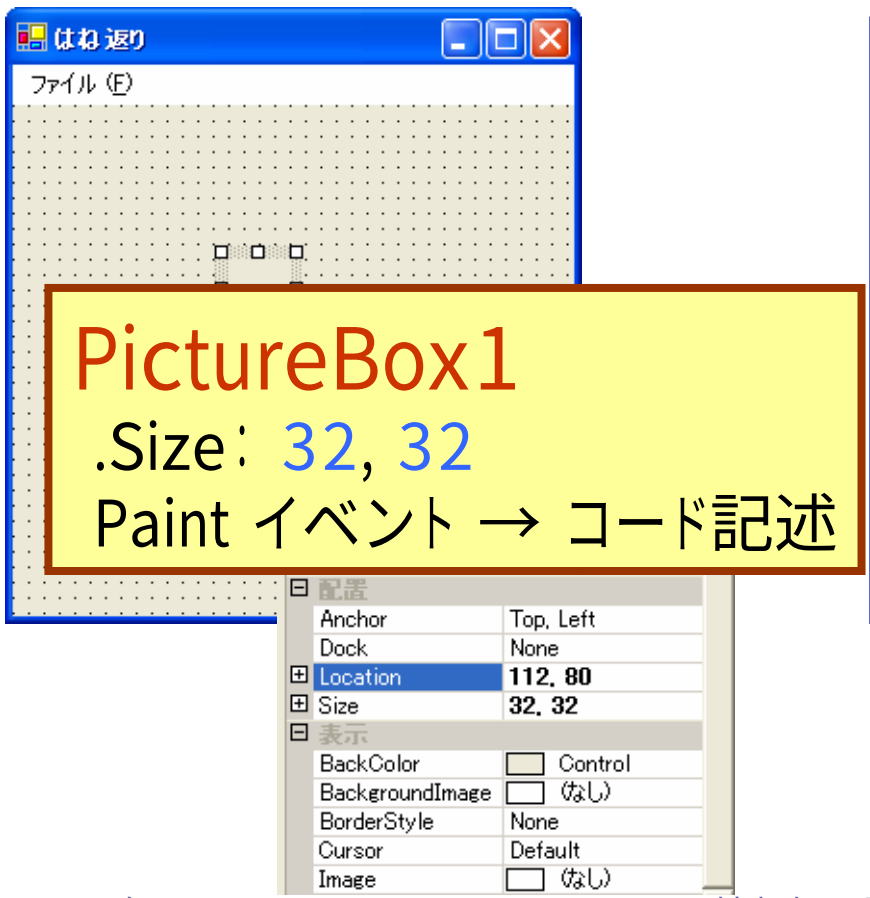
# 画面の構成要素



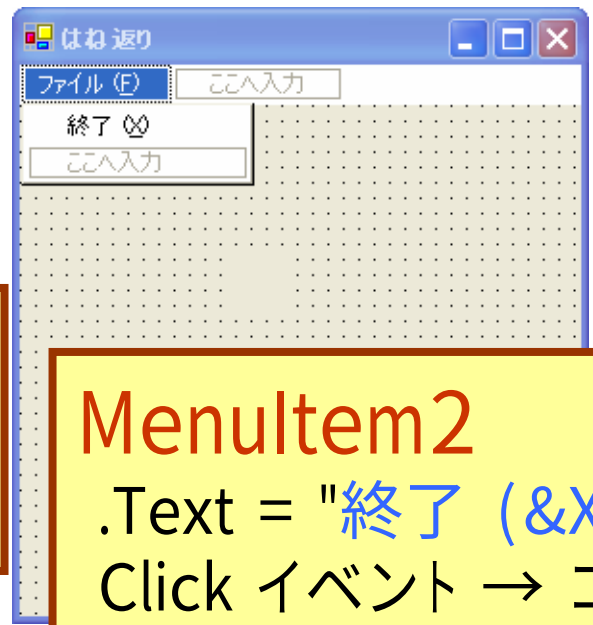

- ツールボックスの中から必要なコントロールを選択し、フォームに貼り付ける。
  - コントロールをダブルクリック
  - コントロールをフォーム上にドラッグ＆ドロップ
- MainMenu および PictureBox

# フォームの構成要素

ピクチャボックスを配置し、メニューを設定する



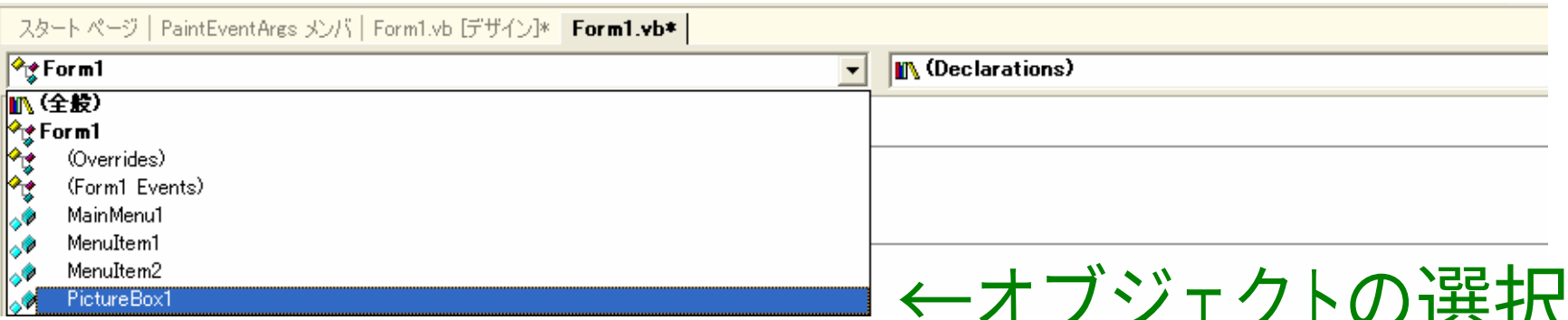

PictureBox1
.Size：32, 32
Paint イベント → コード記述



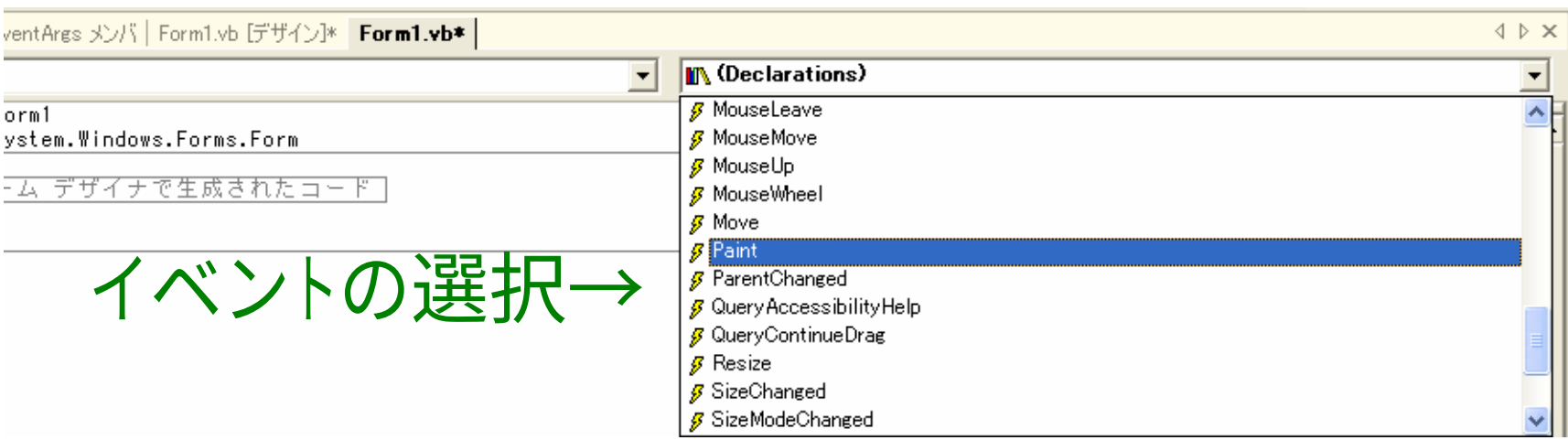

MenuItem2
.Text = "終了(&X)"
Click イベント → コード記述

# Paint イベントの処理コード

- コード画面上部のプルダウンメニューから PictureBox および Paint イベントを選択する。




←オブジェクトの選択




イベントの選択→

# コードの記述

各イベントハンドラを記述する

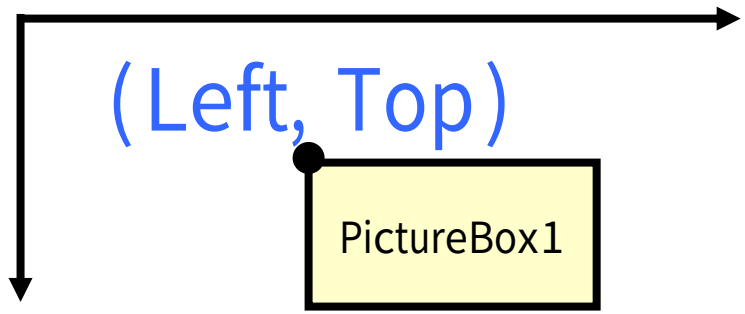
- フォームをクリックして Form1_Load を記述する
- Paint イベント発生時の処理
- Timer の Tick イベント発生時の処理

```
Public Class Form1
    Inherits System.Windows.Forms.Form

Windows フォーム デザイナで生成されたコード

    Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load
        Timer1.Start()
    End Sub

    Private Sub PictureBox1_Paint(ByVal sender As Object, ByVal e As System.Windows.Forms.PaintEventArgs) Handles PictureBox1.Paint
        e.Graphics.DrawEllipse(Pens.Black, 0, 0, 31, 31)
    End Sub

    Private Sub Timer1_Tick(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Timer1.Tick
        PictureBox1.Left += 4
    End Sub

    Private Sub MenuItem2_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MenuItem2.Click
        Timer1.Stop()
    End Sub
End Class
```

# コードの解説

- PictureBox1 のｘ座標を４ずつ増やしていく

  PictureBox1.Left += 4



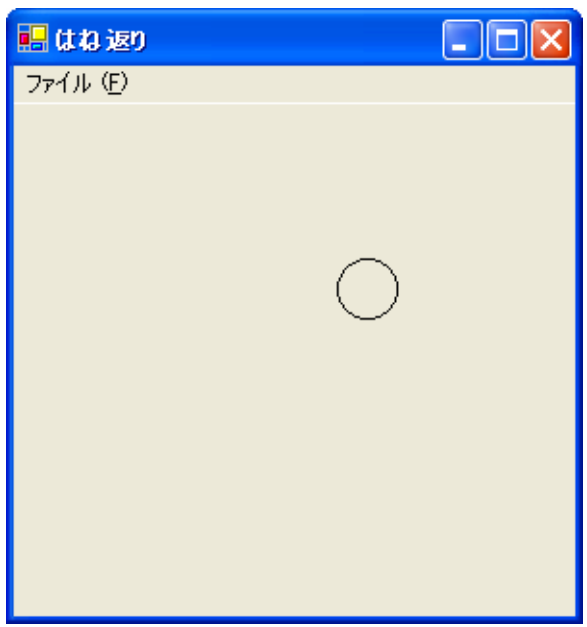

  - PictureBox1.Left

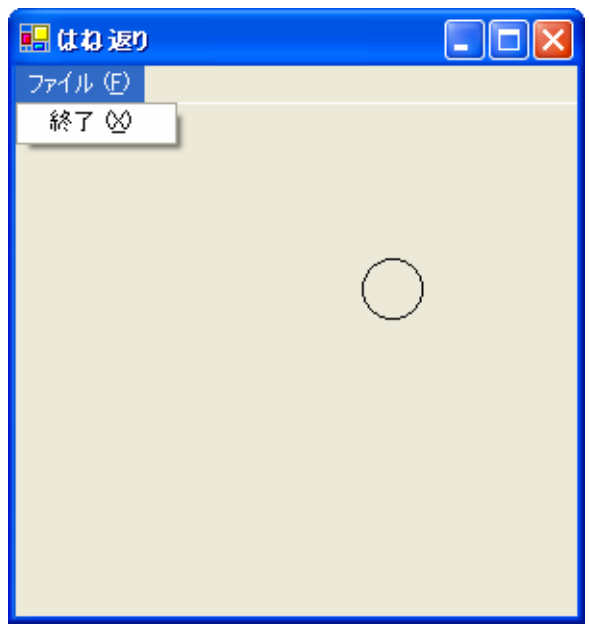
    フォーム座標系におけるオブジェクトのｘ座標
  - PictureBox1.Top

    フォーム座標系におけるオブジェクトのｙ座標

# 実行（デバッグ）


- [デバッグ]→[開始]でプログラムを実行する
- ツールバーの開始ボタンをクリックしてもよい



# グローバル変数

- プログラムのどこからでもアクセスできる変数
  (⇔ローカル変数)
- Form1 クラスの先頭で変数を宣言すると、グローバル変数として任意のプロシージャから読み書きできる。

```
Dim  vx  As Integer        'x方向の速度
```

- フォームの Load イベントプロシージャで初期値を与える。

# コードの追加･変更

- 壁に当たったときに、はね返る処理を加える

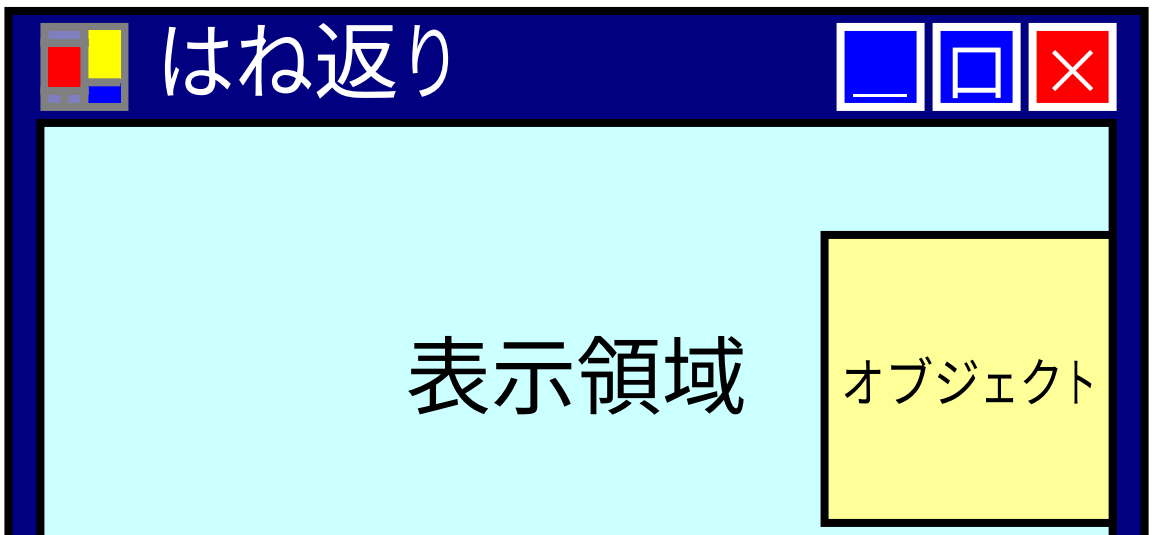
```
Public Class Form1
    Inherits System.Windows.Forms.Form
    Dim vx As Integer '×方向の速度

 Windows フォーム デザイナで生成されたコード 

    Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load
        vx = 4 '×方向の速度の初期値
        Timer1.Start()
    End Sub

    Private Sub PictureBox1_Paint(ByVal sender As Object, ByVal e As System.Windows.Forms.PaintEventArgs) Handles PictureBox1.Paint
        e.Graphics.DrawEllipse(Pens.Black, 0, 0, 31, 31)
    End Sub

    Private Sub Timer1_Tick(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Timer1.Tick

        'フォームの壁に当たったらはね返る処理
        If PictureBox1.Left + 32 >= Me.Width - 8 Or PictureBox1.Left <= 0 Then
            vx = -vx
        End If

        PictureBox1.Left += vx

    End Sub

    Private Sub MenuItem2_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MenuItem2.Click
        Timer1.Stop()
    End Sub
End Class
```

# コードの解説

- ボールが壁に当たると速度の向きを変える

```
If PictureBox1.Left + 32 >= Me.Width - 4 Then
    vx = -vx
End If
```

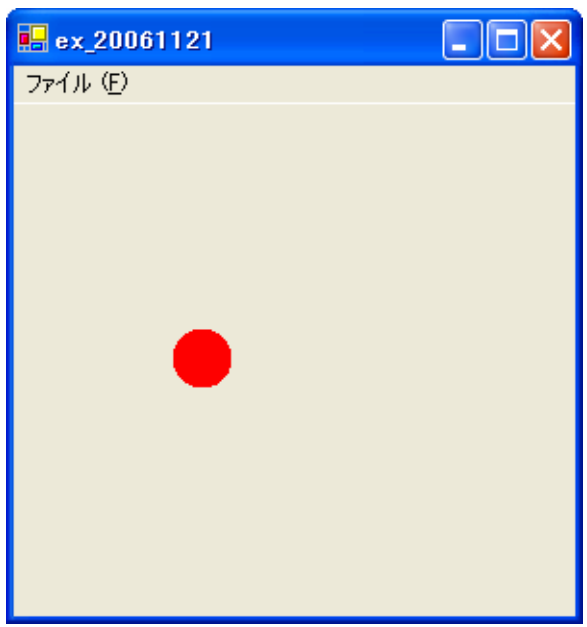
  - オブジェクトの右端の座標が表示領域にあるか？

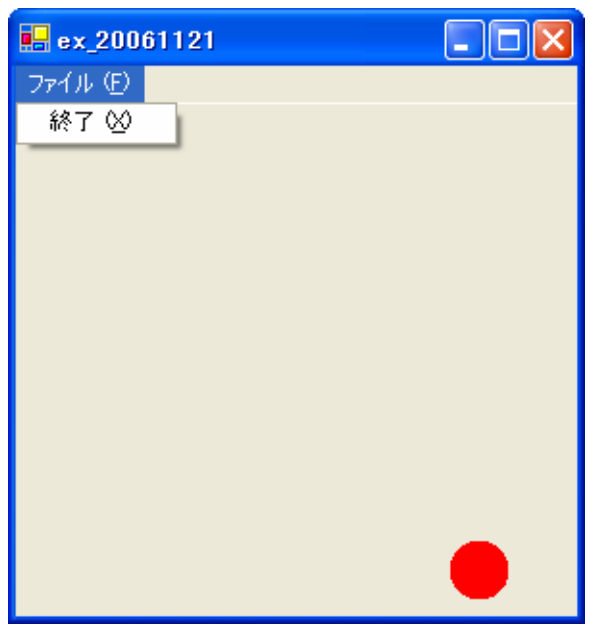


# 課題

動くボールのプログラム（ex_20061121）

- 2次元平面でボールが転がる様子を実現せよ。ただし、壁に当たるとはね返るものとする。



# まとめ

- Windows におけるグラフィックスの機能とTimer コンポーネントを使って、アニメーションを実現した。
  - Timer コンポーネント

# 次回予定

- グラフィックス（6）
  - アニメーションの処理など（予定）