C# によるお絵かきツールの試作

サイバー大学 IT 総合学部 1 年

1101015185 宍戸秀昭

2011 年 12 月 13 日

1 概要

C# を使うと、アプリケーションプログラムが簡単に作れると言われている。本稿では C# を使って簡単なお絵かきツールを実際に作ってみることで、その真偽を検証した。

2 序論

「C#（シーシャープ）はマイクロソフト社によって同社の .NET 戦略の一環として開発されたオブジェクト指向プログラミング言語である。」[1]とあるように、C# で何かプログラムを作るときは .NET Framework と組み合わせて使われることが多い。ここでは処理系として Microsoft VC# 2010 Express Edition と、それに含まれる .NET Framework 4.0 を使うことにする。

作画やその他のアプリケーションとして備える機能は、.NET Framework が提供するライブラリ関数を呼ぶことで実現する。.NET Framework は「.NET Framework は、共通言語ランタイム、階層的な統一されたクラス ライブラリのセット、および Active Server Pages のコンポーネント化されたバージョンである ASP.NET の 3 つの部分から構成されています。」[2]と言われるように複合的な側面を持つが、ここでは .NET Framework を単なる GUI のクラスライブラリとして扱う。

[1] http://ja.wikipedia.org/wiki/C_Sharp
[2] http://msdn.microsoft.com/ja-jp/library/ms973850.aspx#faq111700_concepts02

3 本論

3－1 お絵かきツールの説明

今回作成したお絵かきツールの概観を図 1 に示す。これは一般的な Windows で動作する GUI アプリケーションである[*1]。

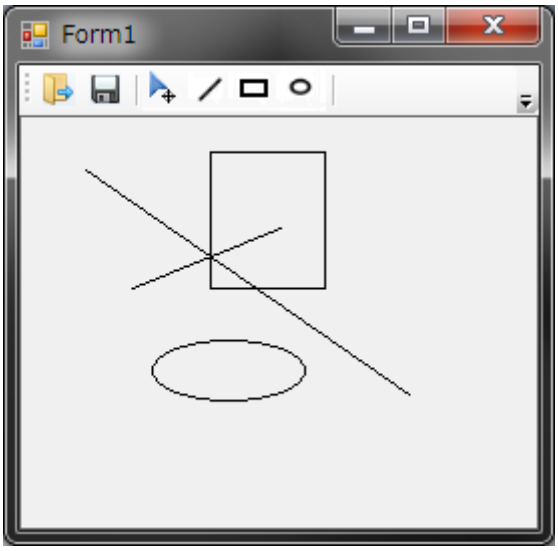

図 1 アプリケーションの外観

ユーザはアイコンで選んだ図形をマウスで描画できる。ユーザの操作はイベントとしてアプリケーションに伝達され、これをアプリケーション内のイベントハンドラで取得する。図形の描画は .NET Framework の PictureBox クラスに対して行う。ユーザが描画した図形はアプリケーションの単一のリストに保存される。

ユーザがマウスをドラッグしている間は、選択した図形を破線で表現したラバーバンドを表示する。

C# はオブジェクト指向プログラム言語なので、このアプリケーションは次のようにクラスとして定義される。

```
public partial class Form1 : Form
{
    private Shape rb;
    private List<Shape> shapes;
      :
}
```

.NET Framework で GUI アプリケーションを作る場合、アプリケーションのウィンドウを、.NET Framework が提供する Form という基底クラスから継承して作るのが一般的である[3]。上記は Form クラスを元に、お絵かきツールが備える特徴を追加した Form1 クラスを定義している。

[*1] 試作したお絵かきツールは実使用を想定したものではなく、作画に必要な最低限の構成とした。
[3] 「サンプルコードで使い方がわかる .NET Framework 実践ライブラリリファレンス」第 11 章 ウィンドウとダイアログボックス

図形の保存先であるリストには、.NET Framework が提供する List<> コレクションを使う。.NET Framework が提供するコレクションにはさまざまなものがある[4]が、ここでは簡単さから List<> を選択した。

3－2 図形クラス

同様にして、ユーザが描画する図形もクラスとして定義する。まず図形の元となる Shape クラスを下記のように定義する。

```
public abstract class Shape : ICloneable
{
    public List<Point> coodinate;

    public  Pen pen;
    public abstract void draw(Graphics g);
    public abstract void draw_rubber(Graphics g);

    public Shape()
    {
        coodinate = new List<Point>();
        pen = new Pen(Color.Black);
    }
    public Object Clone() { return MemberwiseClone(); }
}
```

ここで coodinate リストは図形を構成する座標を表す。例えば四角形の場合は、左上と右下の座標をリストとして保持する。

図形の描画は draw() メソッドが行う。ラバーバンドの描画は特別扱いとして draw_rubber() で行うようにした。

ユーザが描画するすべての図形は Shape クラスを継承して作られる。例えば四角形は下記のようになる。

```
public class Box : Shape
{
    public override void draw(Graphics g)
    {
        g.DrawRectangle(pen, Util.get_rect(coodinate));
    }
    public override void draw_rubber(Graphics g)
    {
        Pen p = new Pen(Color.Blue);
        p.DashStyle = System.Drawing.Drawing2D.DashStyle.Dash;
        g.DrawRectangle(p, Util.get_rect(coodinate));
```

[4]「絶対現場主義 C#入門」3.3.2. 組み込み型とその他大勢型

```
    }
}
```

３－３　描画コマンドと操作モード

ユーザは線分、四角形、楕円等の描画コマンドをアイコンで選択する。プログラムではこれを描画モードとして扱う。また、マウスボタンの押/離等のユーザの操作を操作モードとして扱う。これらを C# の 列挙型 で定義すると次のようになる[5]。

```
public enum Cmd { SELECT, LINE, BOX, OVAL }
public enum Mode { IDLE, BEGIN }
```

描画コマンドはユーザによる描画アイコンの選択で決定される。下記は描画アイコンの四角形が選択された場合のイベントハンドラである[*2]。

```
private void cmdButtonSelect_Click(object sender, EventArgs e)
{
    cmd  = Cmd.SELECT;
    mode = Mode.IDLE;
}
```

一方、操作モードはマウスボタンの押/離により遷移する状態であり、図 2 のようになる。

操作モードの状態遷移は単に mode 変数への代入として扱い、それぞれのイベントハンドラに書く。

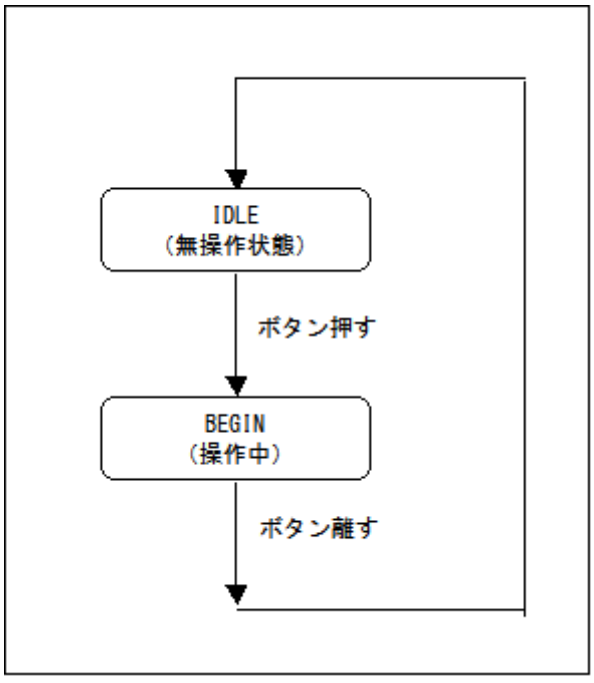

図 2　操作モードの状態遷移

[5]「独習 C#」第 9 章 インターフェイス、構造体、列挙型

[*2] ここで、操作モードを IDLE にしているのは、描画アイコンが選択されたら、それまでのユーザ操作を中止する意図がある。

3－4　マウスボタンを押す

マウスボタンが押されたときに必要な処理を次に示す。

1. 描画コマンドを取得する
2. 座標を記録する
3. 操作モードを BEGIN に変更

2と3はラバーバンドを表示するための準備である。イベントハンドラは下記のようになる。

```
private void pictureBox1_MouseDown(object sender, MouseEventArgs e)
{
    if (mode == Mode.IDLE)
    {
        rb = get_shape(cmd);
        rb.coodinate.Add(new Point(e.X, e.Y));
        rb.coodinate.Add(new Point(e.X, e.Y));
        mode = Mode.BEGIN;
        }
    }
}
```

ラバーバンドはこれからユーザが描く図形と同じ形状にしたい。`get_shape()` はそのための関数で、最後に押された描画ボタンの種類を引数に渡すとその図形オブジェクトを作って返す。

```
private Shape get_shape(Cmd cmd)
{
    switch(cmd)
    {
        case Cmd.LINE: return new Line();
        case Cmd.BOX:  return new Box();
        case Cmd.OVAL: return new Oval();
        default: return new Box();
    }
}
```

3－5　マウスをドラッグする

ドラッグ中はラバーバンドを表示する。操作モードは変更しない。

```
private void pictureBox1_MouseMove(object sender, MouseEventArgs e)
{
    if (mode == Mode.BEGIN)
    {
        pictureBox1.Refresh();
```

```
        rb.coodinate[1] = new Point(e.X, e.Y);
        rb.draw_rubber(graphic);


        draw_all_shapes();
    }
}
```

ラバーバンドはドラッグ操作に追従する必要があるため、このイベントハンドラで消去→描画を繰り返すことで表示している。ラバーバンドを消去するためには、すでに描画されている図形も含め一旦すべて消去する必要がある。その後 `draw_all_shapes()` が、それらを再描画する。

```
private void draw_all_shapes()
{
    foreach(Shape sh in shapes)
    {
        sh.draw(graphic);
    }
}
```

なお、`pictureBox1.Refresh()` は描画領域を再描画する `.NET Framework` が提供するメソッドである。

3－6 マウスボタンを離す

ユーザがマウスボタンを離したとき、以下のような処理が必要となる。

1. それまで表示していたラバーバンドを消す
2. ユーザの描画図形として登録
3. 登録した図形を再描画
4. 次のユーザ操作に備えるため、操作モードを IDLE に変更

これをイベントハンドラに記述すると次のようになる。

```
private void pictureBox1_MouseUp(object sender, MouseEventArgs e)
{
    if(mode == Mode.BEGIN)
    {
        if(cmd != Cmd.SELECT)
        {
            rb.coodinate[1] = new Point(e.X, e.Y);
            shapes.Add(rb);
        }
        mode = Mode.IDLE;
    }
    pictureBox1.Refresh();
```

```
    draw_all_shapes();
}
```

3－7 まとめ

ここまでで、お絵かきツールとしての一通りの機能を実装したことになる。実際にビルドしてみると Windows アプリケーションとして動作することが確認できる。

冒頭で述べたとおり、このお絵かきツールは .NET Framework が提供する Form クラスを元に作られており、Windows アプリケーションとして要求される仕事のほとんどが Form クラスで処理される。そのため、アプリケーションの設計者としては、Form クラスが処理しないアプリケーション固有の処理だけを記述するだけでよい。さらに、作画等のアプリケーション固有の処理も .NET Framework が提供するメソッドを効果的に呼び出すだけで済ませられる場合が多いことが分かった。

4 論議

今回は、簡単なアプリケーションを書いてみる事で C# の簡便さを評価できたと思う。実際、筆者は C# についてはほとんど知らない状態から入門書を片手に始めたが、プログラミングにかけた時間は多く見ても 8 時間足らずであった[*3]。

C# にはここでは挙げていない機能が多くあり、それらを使うことでよりシンプルなコードを記述できる。例えば、C# バージョン 3.0 で導入された Func という標準のジェネリックデリゲートを用いると、描画コマンドの選択処理と図形ごとの描画メソッドはより短く書き換えることができると思う。

C# が短い期間でバージョンアップを繰り返していることから、今後も C# の簡便さは高まっていくと思われる。

[*3] ソースコード(Form1.cs)は 250 行程度となった。この行数は Visual C# の GUI デザイナが自動生成したコードを含んでいる。

参考文献

1. 著者:匿名

タイトル:「C Sharp」

URI:http://ja.wikipedia.org/wiki/C_Sharp

アクセス日付:2010/12/10

2. 著者:Microsoft Corporation

タイトル:「.NET Framework とは何ですか?」

URI:http://msdn.microsoft.com/ja-jp/library

/ms973850.aspx#faq111700_concepts02

アクセス日付:2010/12/10

3. 著者:日向 俊二

タイトル:「サンプルコードで使い方がわかる

.NET Framework 実践ライブラリリファレンス」

出版社／版次／出版年:アスキー・メディアワークス／第 1 版第 1 刷／2009/4/28

4. 著者:丸岡 孝司

タイトル:「絶対現場主義 C#入門」

出版社／版次／出版年:ラトルズ／第 1 版第 1 刷／2011/10/12

5. 著者／訳者:ハーバート・シルト／エディフィストラーニング株式会社 矢嶋聡

タイトル:「独習 C#」

出版社／版次／出版年:翔泳社／第 3 版第 1 刷／2010/12/3