

LẬP TRÌNH GDI+ # PHẦN 3

CÁC CÔNG CỤ NÂNG CAO (TIẾP THEO)



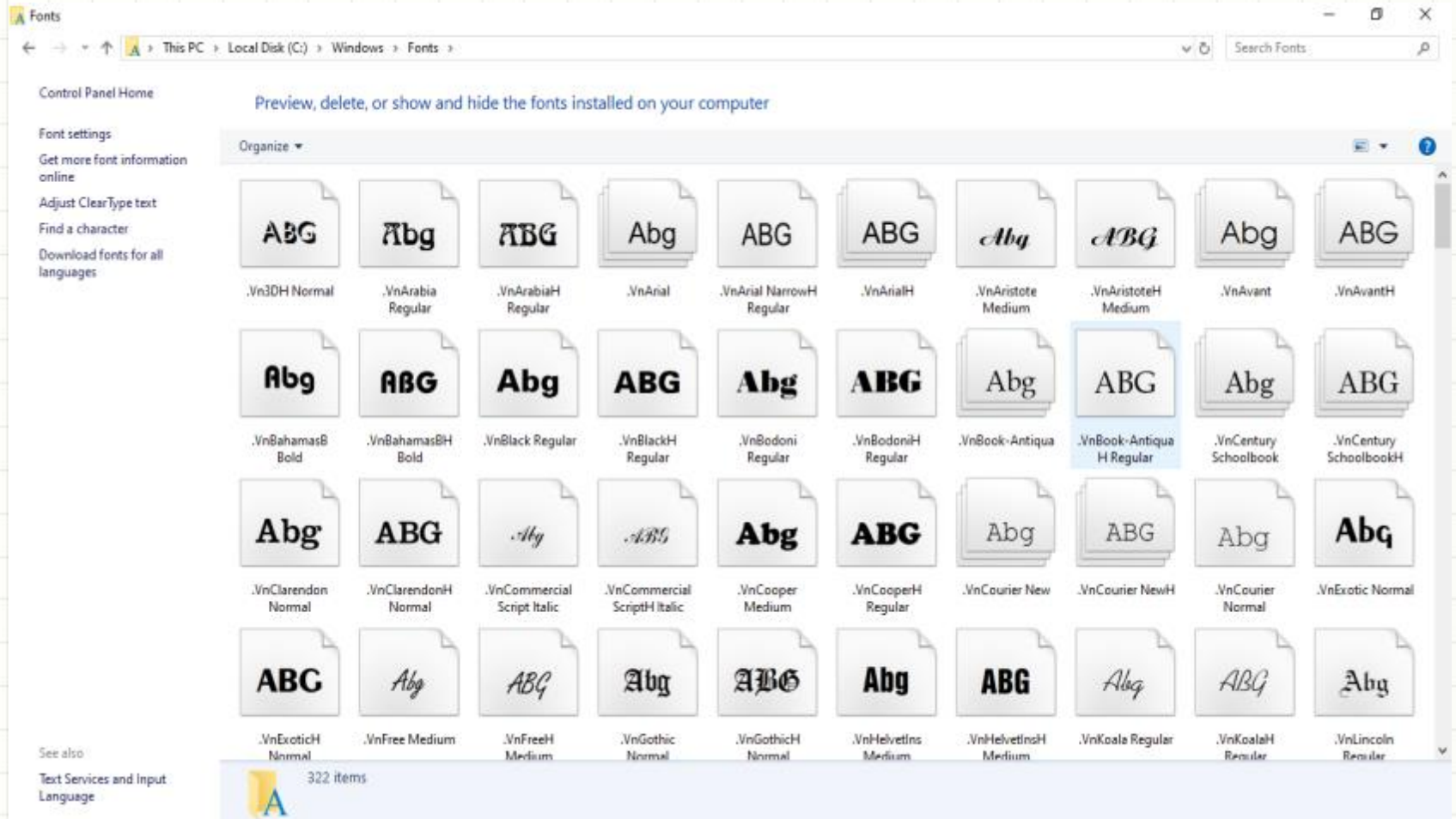
Nội dung – Phần 3

- | | |
|---|-----------------------------------|
| 1 | Fonts |
| 2 | Vẽ chữ |
| 3 | Biến đổi hệ trục (transformation) |
| 4 | Bài tập |

Fonts

- ❖ Windows hỗ trợ hai loại phong chữ: Device fonts và GDI fonts
- **Device fonts** có nguồn gốc từ các thiết bị đầu ra như màn hình, máy in,...
- **GDI fonts** được lưu trữ trong các tập tin trên hệ thống, thường chứa trong thư mục Windows\Fonts.
- ❖ Mỗi font có tập tin riêng, ví dụ: Arial, Arial Black, Arial Bold, Arial Italic, Italic Arial Black, Arial Bold Italic, Arial Narrow, Italic Arial Narrow Bold, và Arial Narrow Italic, ...

Fonts



Fonts

Thư viện GDI+ cung cấp các lớp font cho phép thiết lập font chữ khi xuất văn bản là **FontFamily** và **Font**

class FontFamily: cung cấp các loại font chữ được định nghĩa sẵn để sử dụng trong kết xuất văn bản, lớp này chứa các thuộc tính như

- **Name**: tên font chữ
- **Families**: mảng chứa tất cả các font families trong thiết bị ngữ cảnh

Fonts

class Font: là lớp cho phép tạo đối tượng font chữ với một số thuộc tính như kiểu chữ (đậm, nghiêng, gạch dưới,...), kích thước,... Một số phương thức khởi tạo thường được sử dụng:

- Font (FontFamily family, float size)
- Font (string familyName, float size)
- Font (FontFamily family, float size, FontStyle style)
- Font (string familyName, float size, FontStyle style)

Fonts

```
private void mainForm_Paint(object sender, PaintEventArgs e)
{
    Font arial = new Font("Arial", 12);
    e.Graphics.DrawString("Arial Size 12", arial, Brushes.Blue, 10, 10);
    FontStyle style = FontStyle.Bold | FontStyle.Italic;
    FontFamily family = new FontFamily("Tahoma");
    Font tahoma = new Font(family, 12, style);
    e.Graphics.DrawString("Tahoma Size 12", tahoma, Brushes.Red, 10, 30);
}
```



Vẽ chữ

Sử dụng phương thức **DrawString**

➤ Vẽ chữ xuất phát tại một điểm:

DrawString (string, Brush, Font, Point): vẽ chuỗi string, với Font, tại vị trí Point, màu Brush

DrawString (string, Brush, Font, Point, StringFormat): vẽ chuỗi string, với Font, tại vị trí Point, màu Brush và thiết lập định dạng văn bản xuất, chuyển hướng văn bản phụ thuộc vào StringFormat,...

Trong đó, **StringFormat** là đối tượng cho phép thiết lập các định dạng xuất của văn bản, dùng thuộc tính **FormatFlags**

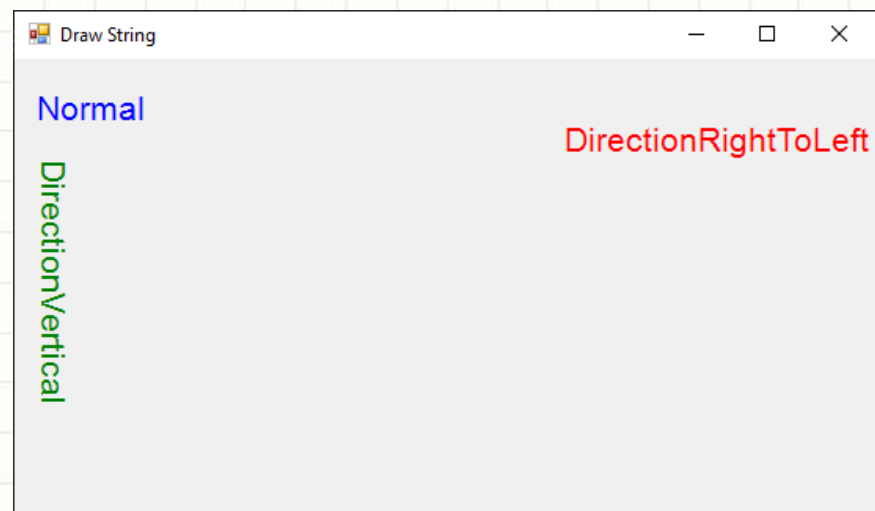
Vẽ chữ

Giá trị của **FormatFlags** là một giá trị thuộc kiểu dữ liệu liệt kê **StringFormatFlags**

- **DirectionRightToLeft**
- **DirectionVertical**
- **NoClip**
- **NoWrap**
- ...

Vẽ chữ

```
private void mainForm_Paint(object sender, PaintEventArgs e)
{
    Font arial = new Font("Arial", 16);
    e.Graphics.DrawString("Normal", arial, Brushes.Blue, 10, 20);
    StringFormat format = new StringFormat();
    format.FormatFlags = StringFormatFlags.DirectionRightToLeft;
    e.Graphics.DrawString("DirectionRightToLeft", arial, Brushes.Red
        , ClientRectangle.Width, 40, format);
    format.FormatFlags = StringFormatFlags.DirectionVertical;
    e.Graphics.DrawString("DirectionVertical", arial, Brushes.Green
        , 10, 60, format);
}
```



Vẽ chữ

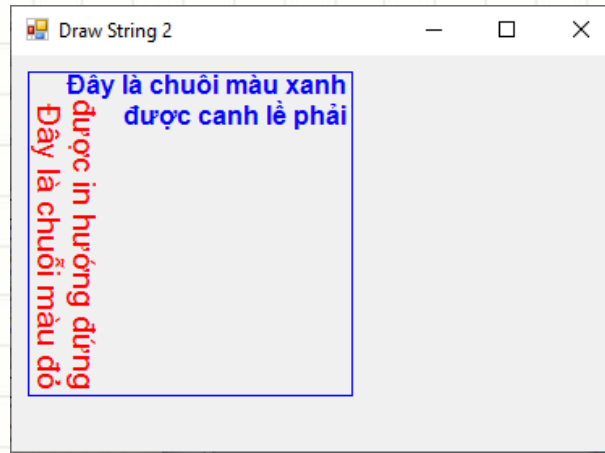
➤ Vẽ chữ trong khung chữ nhật:

DrawString (string, Brush, Font, Rectangle): vẽ chuỗi string, với Font, màu Brush, tại góc trên trái hình chữ nhật Rectangle

DrawString (string, Brush, Font, Rectangle, StringFormat): vẽ chuỗi string, với Font, màu Brush, có thiết lập định dạng văn bản xuất, canh lề, chuyển hướng văn bản phụ thuộc vào StringFormat

Vẽ chữ

```
private void mainForm_Paint(object sender, PaintEventArgs e)
{
    Rectangle rect = new Rectangle(10, 10, 200, 200);
    e.Graphics.DrawRectangle(Pens.Blue, rect);
    StringFormat format1 = new StringFormat();
    format1.FormatFlags = StringFormatFlags.DirectionRightToLeft;
    string st1 = "Đây là chuỗi màu xanh được canh lề phải";
    Font font1 = new Font("Arial", 12, FontStyle.Bold);
    e.Graphics.DrawString(st1, font1, Brushes.Blue, rect, format1);
    StringFormat format2 = new StringFormat();
    format2.FormatFlags = StringFormatFlags.DirectionVertical;
    format2.Alignment = StringAlignment.Far;
    string st2 = "Đây là chuỗi màu đỏ được in hướng đứng";
    Font font2 = new Font("Arial", 14);
    e.Graphics.DrawString(st2, font2, Brushes.Red, rect, format2);
}
```



Biến đổi hệ trục

Các hệ tọa độ trong GDI+: Khi vẽ bằng Graphics trong GDI+, một điểm đi qua **chuỗi chuyển đổi** các hệ tọa độ sau:

World Coordinates

↓ (World Transform)

Page Coordinates

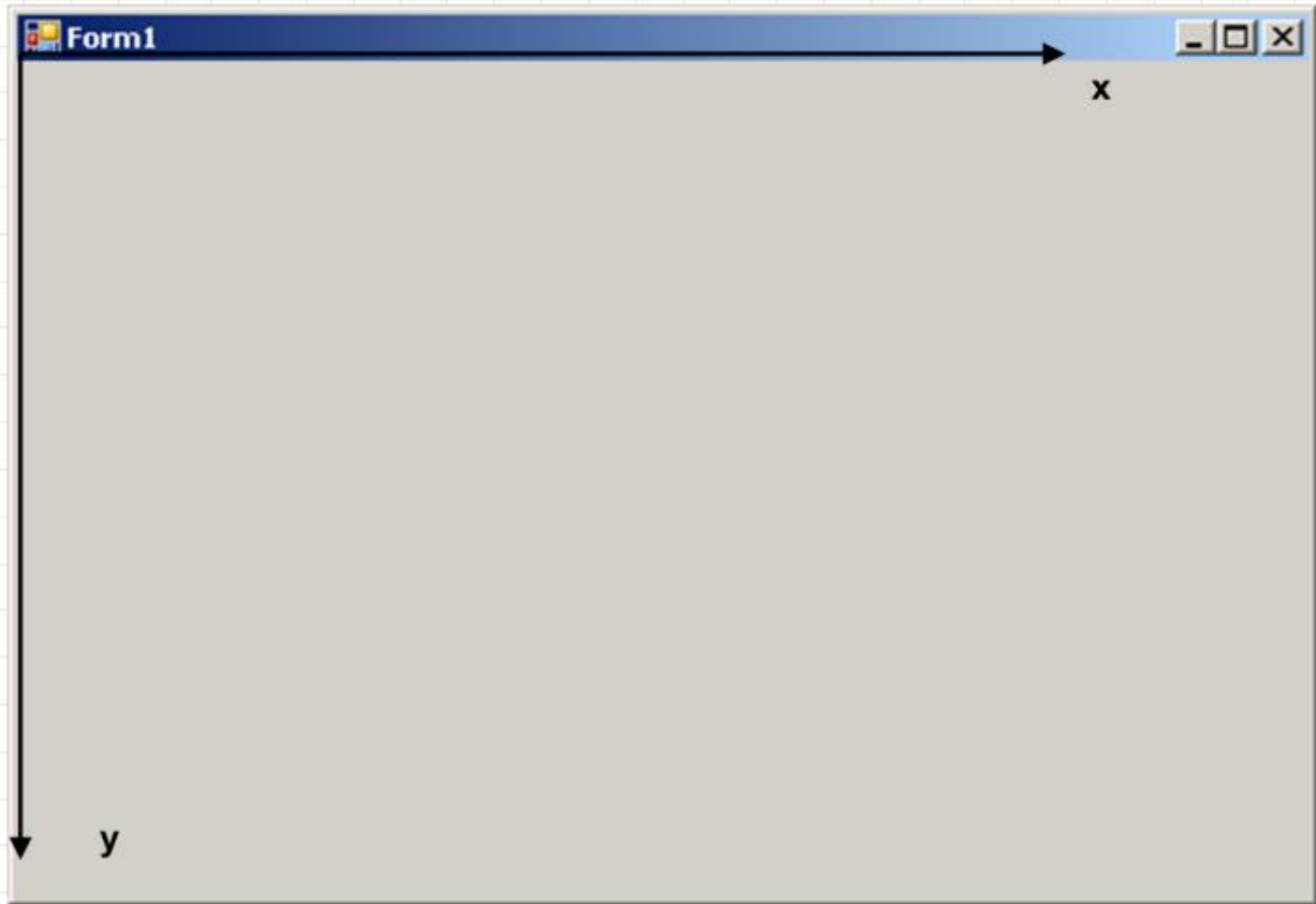
↓ (Page Transform)

Device Coordinates

- **World coordinates**: Hệ tọa độ logic mà ta dùng khi viết code (ví dụ: vẽ Rectangle tại (100, 100)).
- **Page coordinates**: Là hệ tọa độ *sau khi áp dụng các phép biến đổi (transform)* như Translate, Scale, Rotate...
- **Device coordinates**: Là tọa độ thực tế trên màn hình (pixel thật).

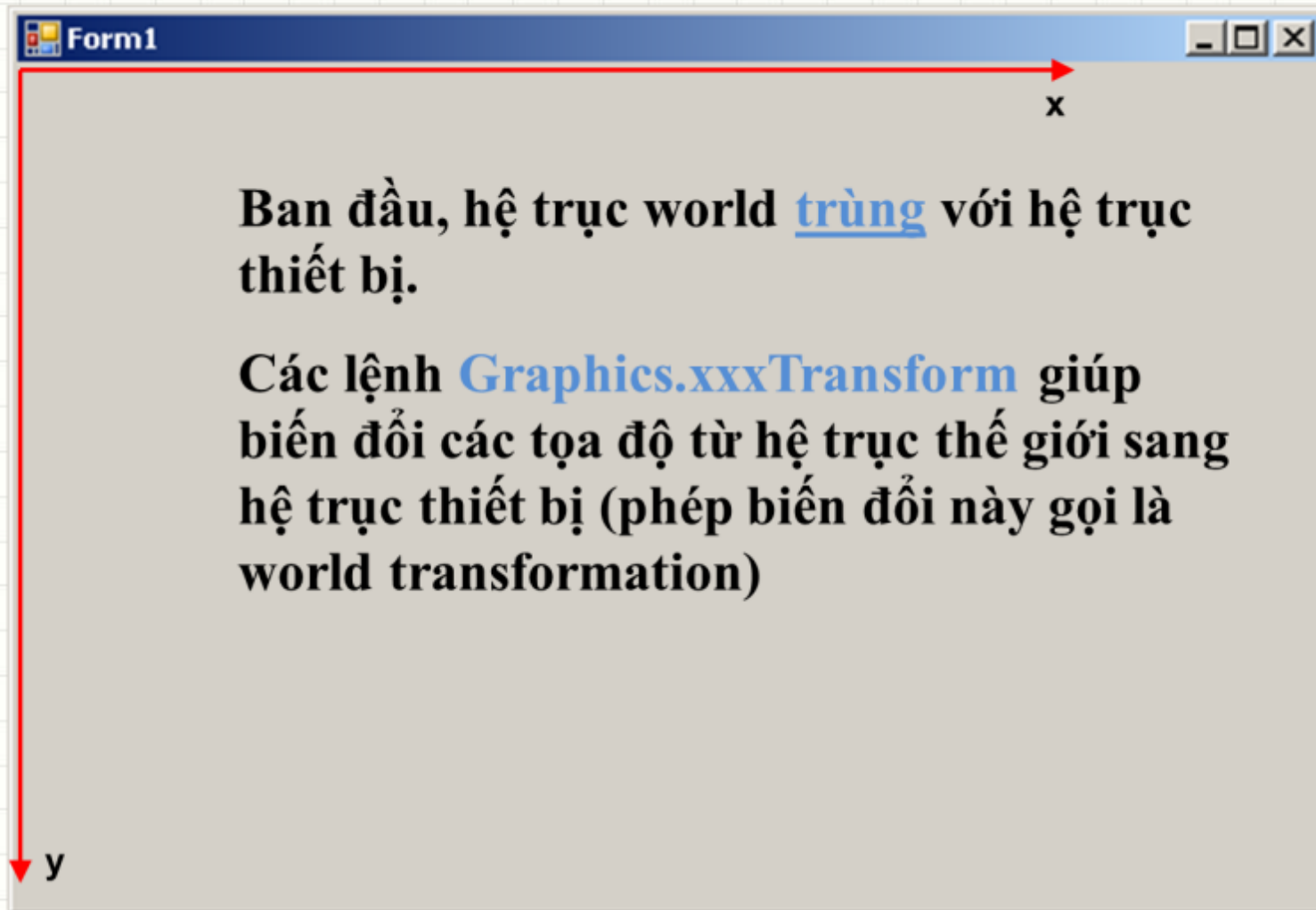
Biến đổi hệ trục

- Hệ trục thiết bị - form



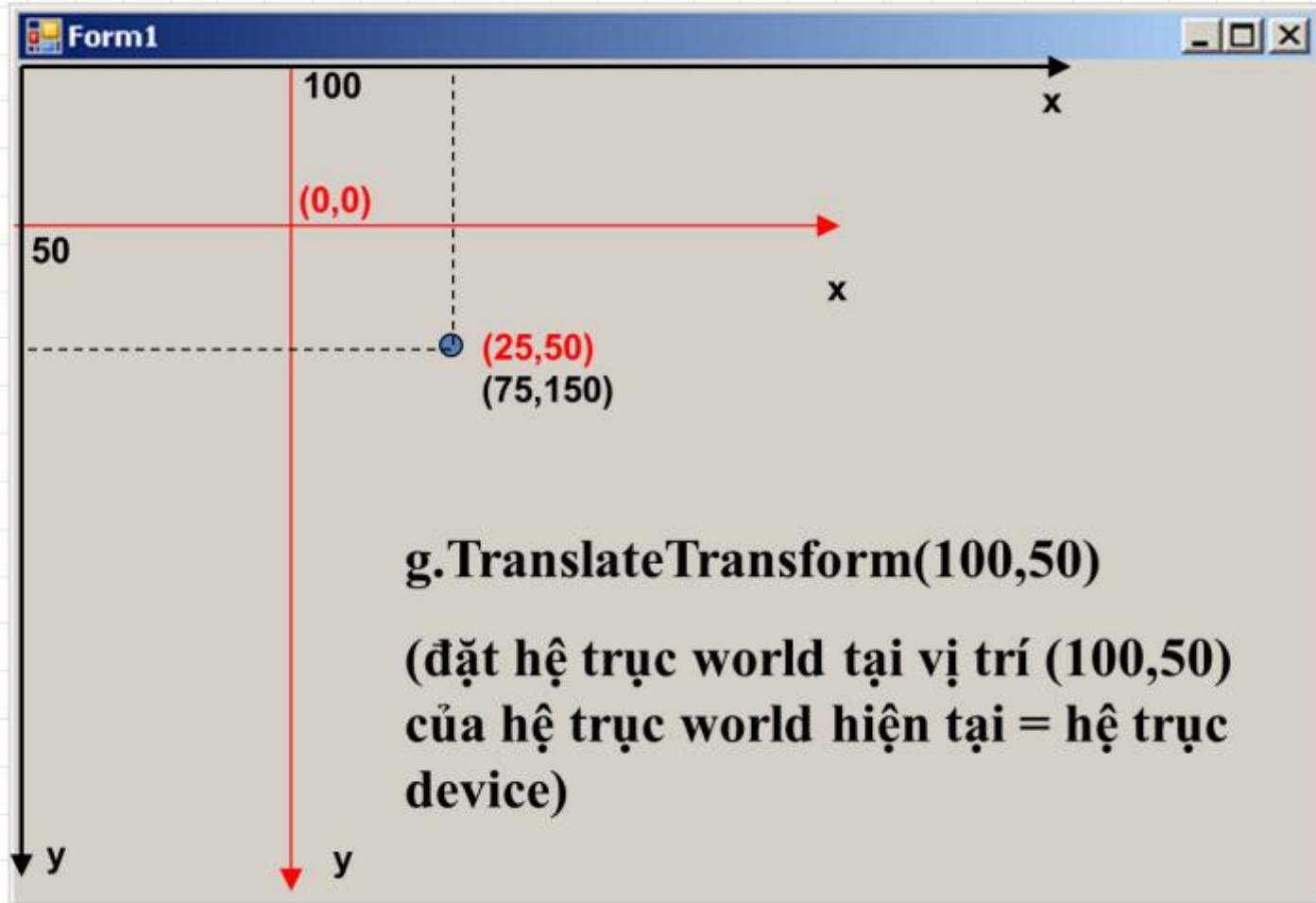
Biến đổi hệ trục

- Hệ trục thế giới (ảo – cơ sở của các lệnh Draw, Fill)



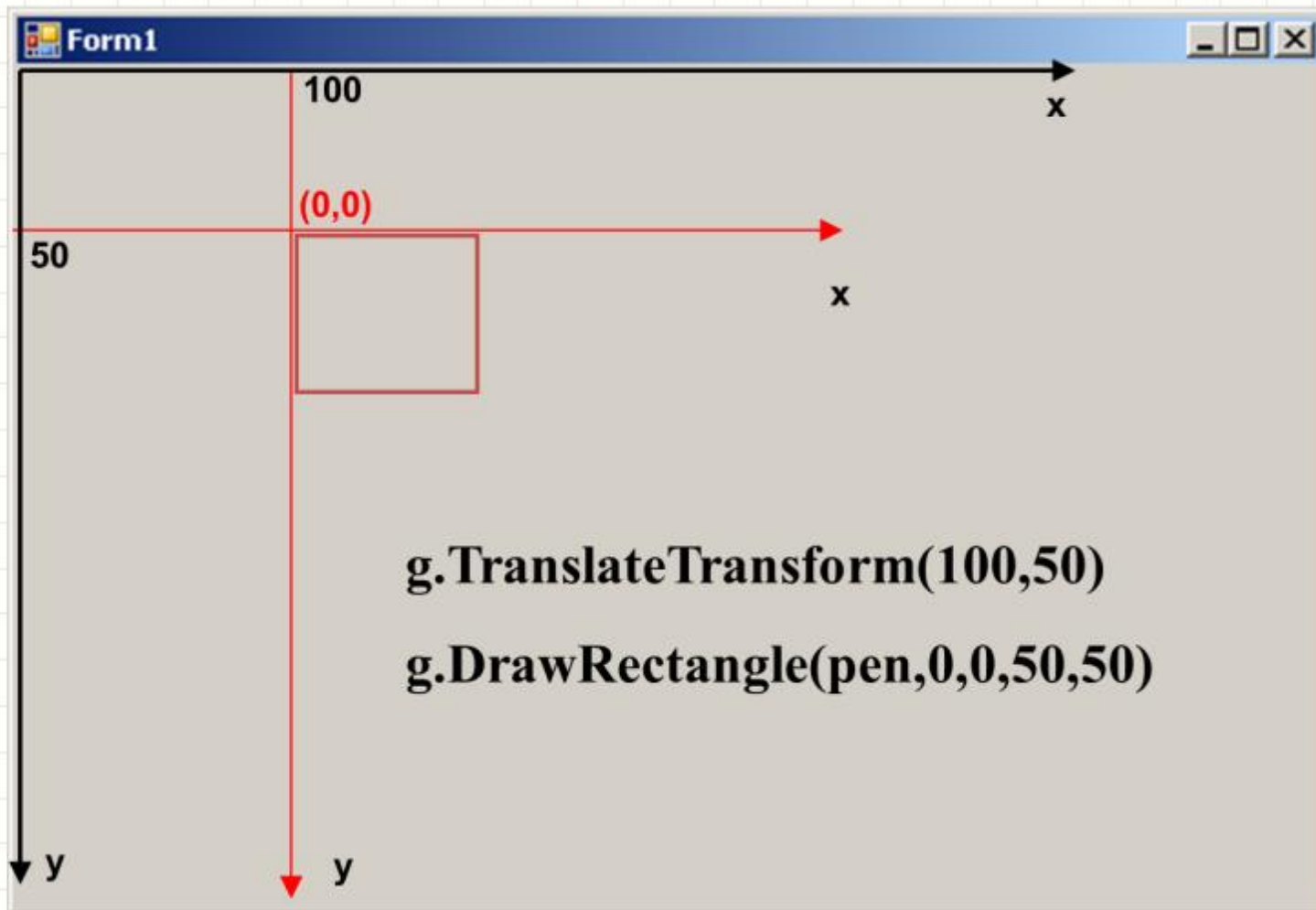
Biến đổi hệ trục

- World coordinate – hệ trục ảo – cơ sở của các lệnh Draw, Fill



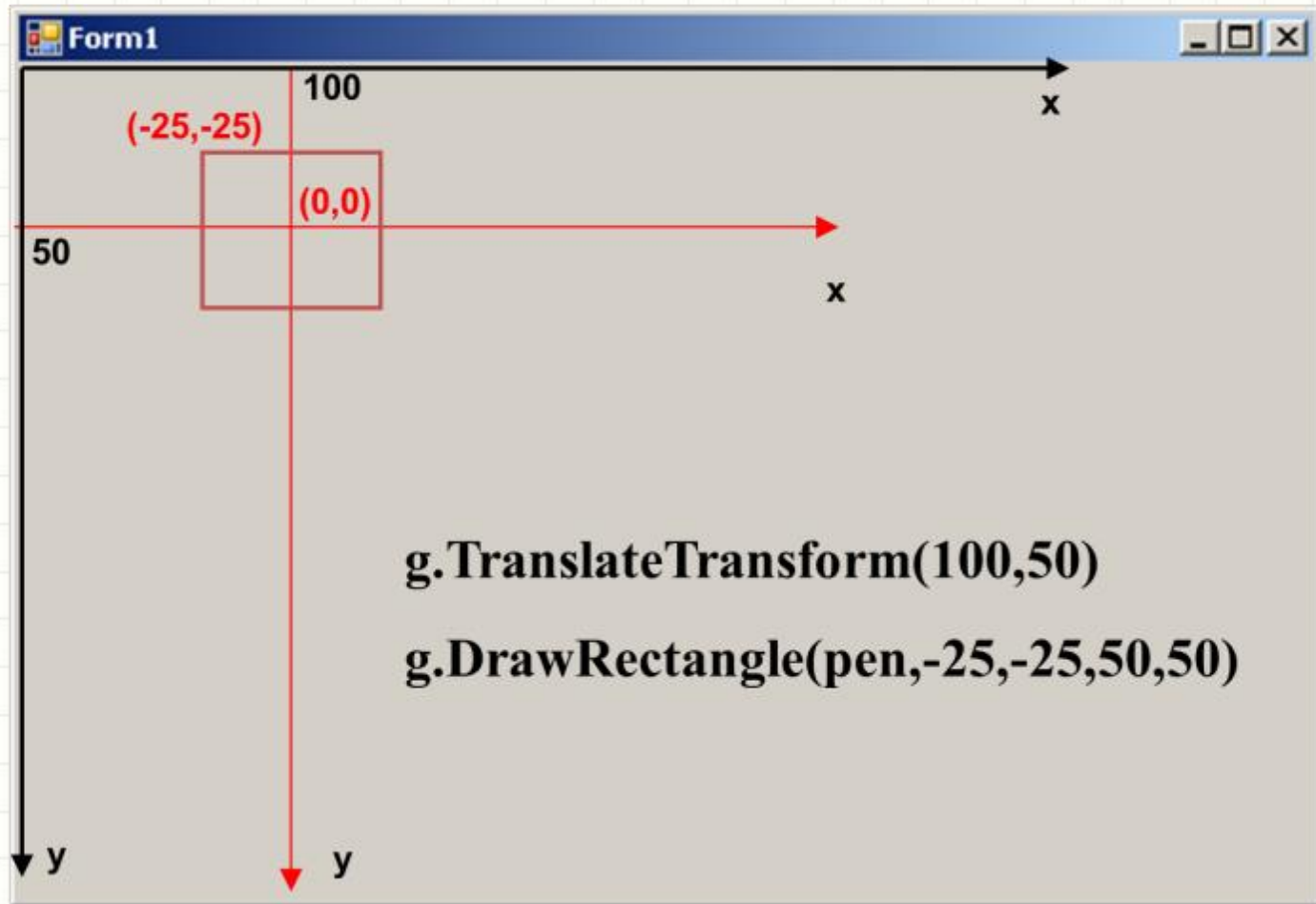
Biến đổi hệ trục

- World coordinate – hệ trục **ảo** – cơ sở của các lệnh Draw, Fill



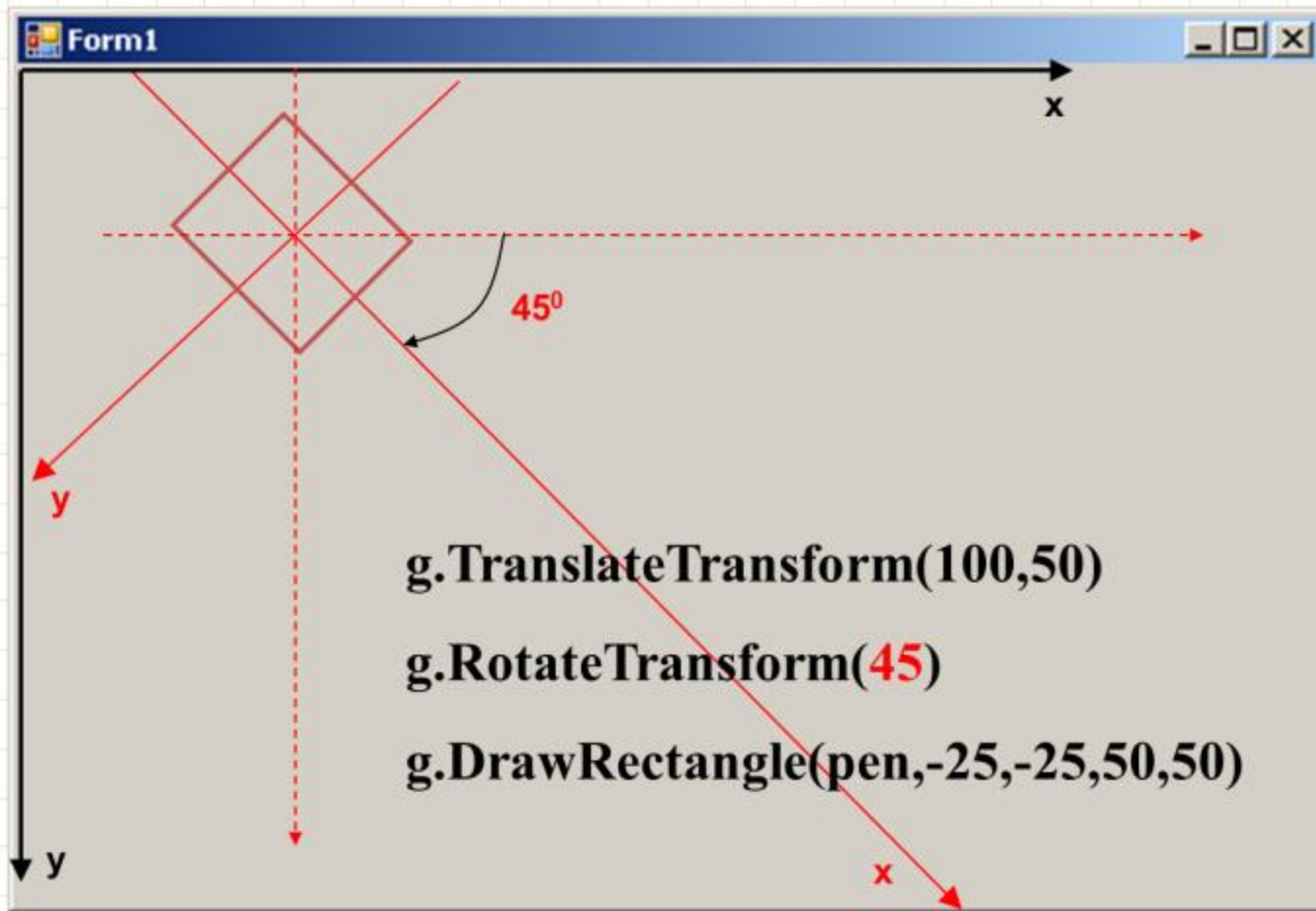
Biến đổi hệ trục

- World coordinate – hệ trục ảo – cơ sở của các lệnh Draw, Fill



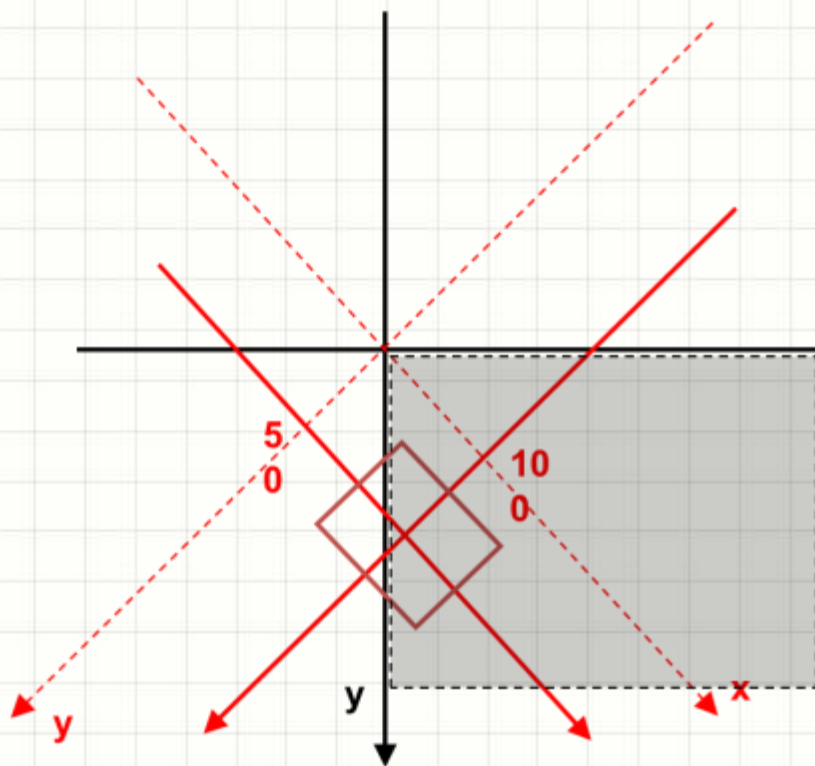
Biến đổi hệ trục

- World coordinate – hệ trục ảo – cơ sở của các lệnh Draw, Fill



Biến đổi hệ trục

- Thứ tự phép biến đổi là quan trọng, áp dụng thứ tự biến đổi khác nhau sẽ tạo ra hiệu ứng khác nhau.



`g.RotateTransform(45)`

`g.TranslateTransform(100,50)`

`g.DrawRectangle(pen,-25,-
25,50,50)`

Bài tập

Dùng các phép biến đổi trục để tạo hiệu ứng hình vuông xoay giữa form

Hướng dẫn: dùng Timer để lặp lại việc vẽ hình vuông ở những góc xoay α tăng dần lên qua các bước:

- 1) Chuyển hệ trục về tâm của form
- 2) Xoay hệ trục một góc α
- 3) Vẽ hình vuông có tâm ở tâm của form