GDI+ (Graphic Device Interface)

Tổng quan

- Thư viện giúp "vẽ" lên màn hình hoặc máy in mà không cần quan tâm đến cấu trúc phần cứng → **độc lập thiết bị**
- GDI+ bao gồm 3 nhóm "dịch vụ" chính:
  - 2D vector graphics: cho phép tạo hình từ các hình cơ bản (primitive): đường thẳng, tròn, eclipse, đường cong,…
  - Imaging: làm việc với các tập tin hình ảnh (bitmap, metafile)
  - Typography: vẽ chữ

GDI+ namespace

- System.Drawing
- System.Drawing. Drawing2D
- System.Drawing.Imaging
- System.Drawing.Printing
- System.Drawing.Text

Các khái niệm

Bề mặt vẽ: Graphics (System.Drawing)

- Lấy từ **Paint** event (form)
- CreateGraphics (trong **control**)

```
protected override void OnPaint(PaintEventArgs e)
{
   Graphics g  = e.Graphics;
   Pen      pen   = new Pen(Color.Red);
   g.DrawLine(pen,0,0,100,100);
}
```

Các khái niệm

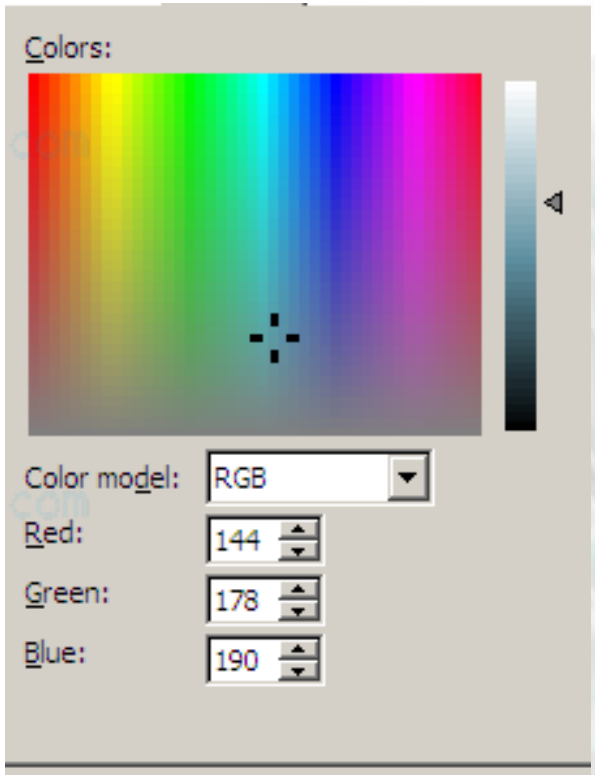
```
private void button1_click(Object o, EventArgs e)
{
   Graphics g  = this.CreateGraphics();
   Pen      pen   = new Pen(Color.Red,15);
   g.DrawLine(pen,0,0,100,100);
    g. Dispose();
}
```

Invalidate();

Invalidate(myRect);

Một số cấu trúc

- Color
- Point, PointF
- Rectangle, RectangleF
- Size, SizeF



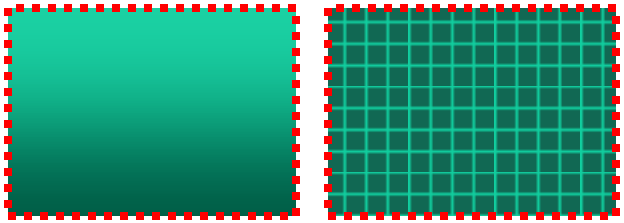
Point, PointF	X,Y +, -, ==, !=, IsEmpty
Rectangle, RectangleF	X,Y Top, Left, Botton, Right Height, Width Inflate(), IntersSec,() Union() Contain()
Size, SizeF	+, -, ==, != Height, Width
Region	“phần ruột” của khuôn hình học Rectangle rect=new Rectangle(0,0,100,100) Region rgn= new Region(rect)

Một số enumeration

- ContentAlignment
- FontStyle
- GraphicsUnit
- KnowColor
- RotateFlipType
- StringAlignment
- …..

2D vector graphics

Pen & brush



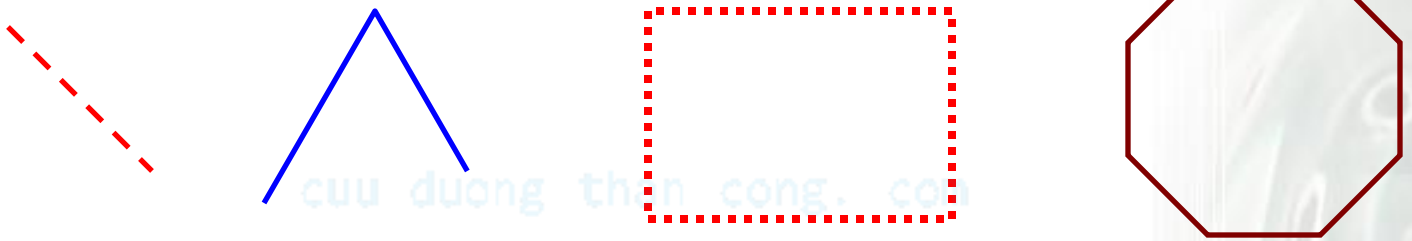
Pen, Pens, SystemPens

Brush, Brushes, SystemBrushes, SolidBrushes, TextureBrushes,

(System.Drawing.Drawing2D)

HatchBrush, LinearGradientBrush, PathGradientBrush

Lines, rectangle, polygon



DrawLine

DrawLines

DrawRectangle

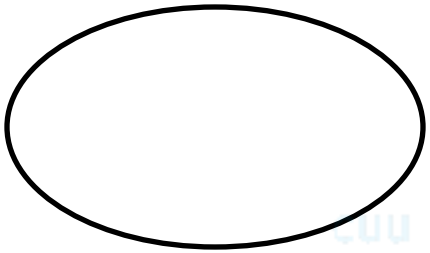
FillRectangle

DrawPolygon

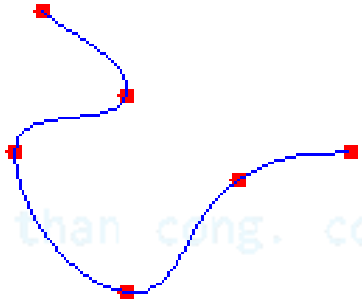
FillPolygon

2D vector graphics

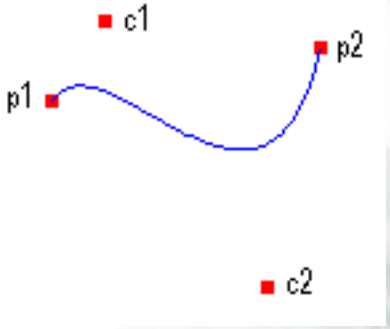
ellipse, arc, cardinal spline, bezier spline



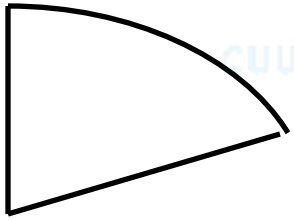
DrawEllipse
FillEllipse



DrawCurve
DrawClosedCurve
FillClosedCurve



DrawBezier
DrawBeziers

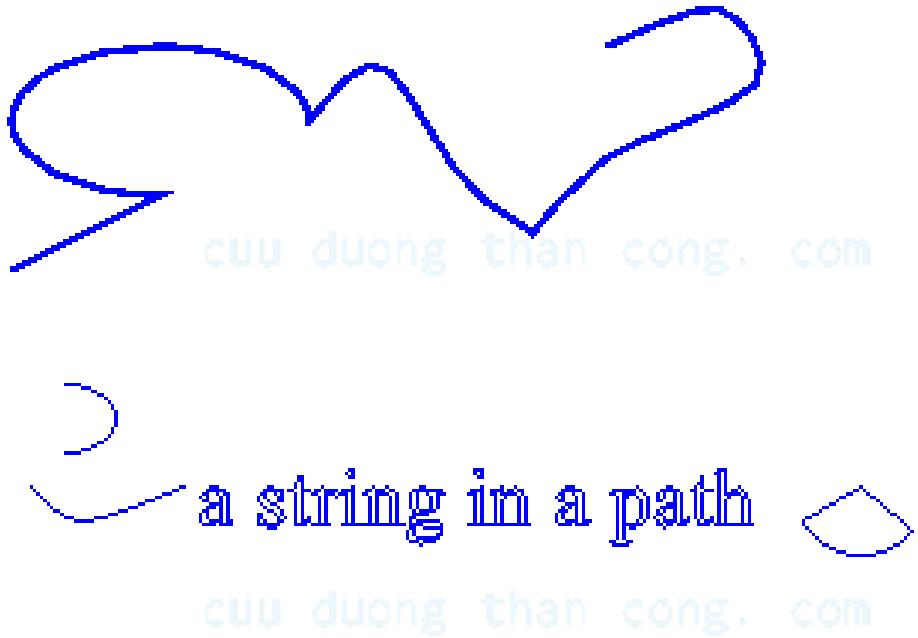


DrawPie
FillPie

DrawArc

2D vector graphics

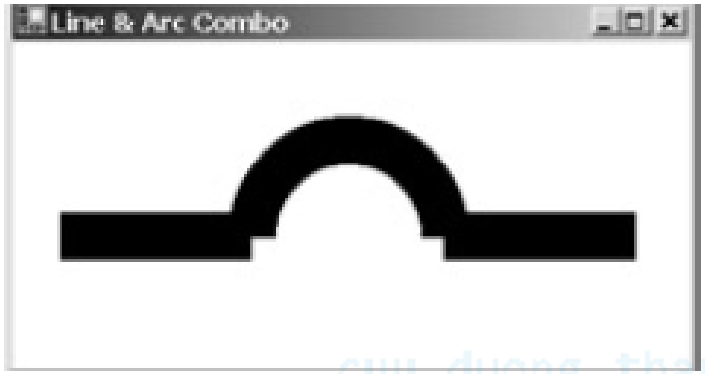
Path: kết hợp nhiều loại đường nét thành một đối tượng duy nhất. Các "nét" không nhất thiết phải liền nhau.

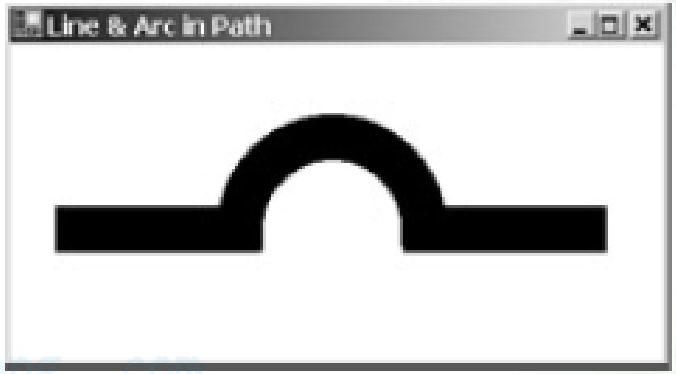


GraphicsPath (AddLine, AddCurve, …)

Graphics.DrawPath

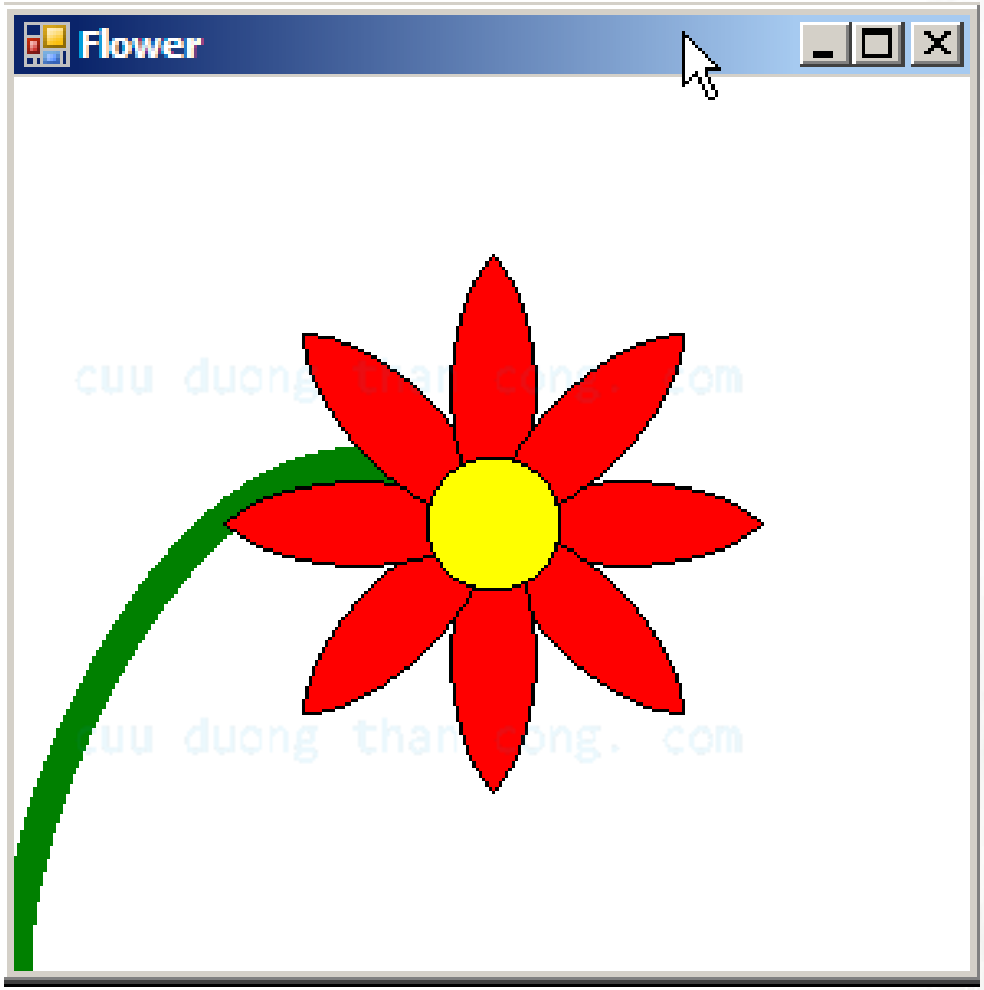
Graphics.FillPath





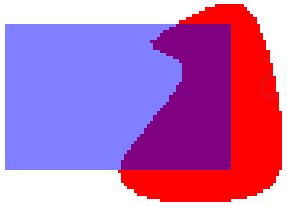

```
grfx.DrawLine(pen,  25, 100, 125, 100);
grfx.DrawArc (pen, 125,  50, 100, 100, -180, 180);
grfx.DrawLine(pen, 225, 100, 325, 100);
```

```
    GraphicsPath path = new GraphicsPath();
     path.AddLine( 25, 100, 125, 100);
     path.AddArc (125,  50, 100, 100, -180, 180);
     path.AddLine(225, 100, 325, 100);
       Pen          pen  = new Pen(clr, 25);
       grfx.DrawPath(pen, path);
```



2D vector graphics

- **Region:** một **vùng** được tạo ra bằng các phép kết giữa các hình chữ nhật hoặc **path**. Region thường được dùng cho "hit-test" hoặc "clipping"



Intersection

Union

System.Drawing.Drawing2D

Region.Intersect, Union, Xor, Exclude, Complement

Xor

The curved region excluded from the rectangular region

2D vector graphics

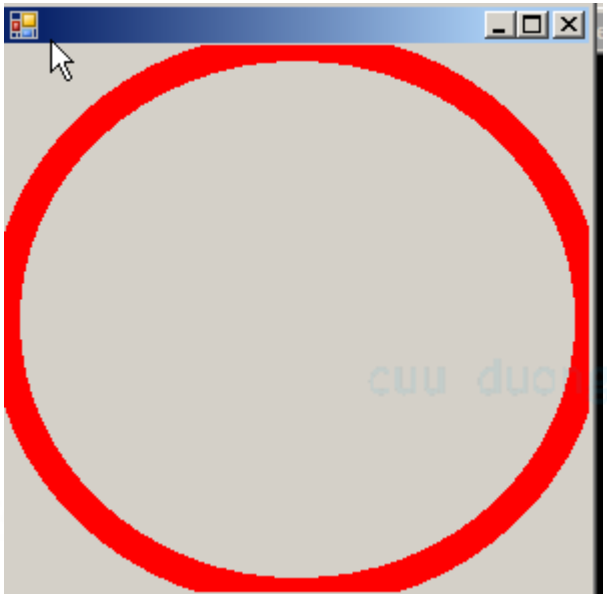
Clipping: giới hạn các hình vẽ vào trong một **region, path** hoặc **rectangle**

Graphics.SetClip(<region>)

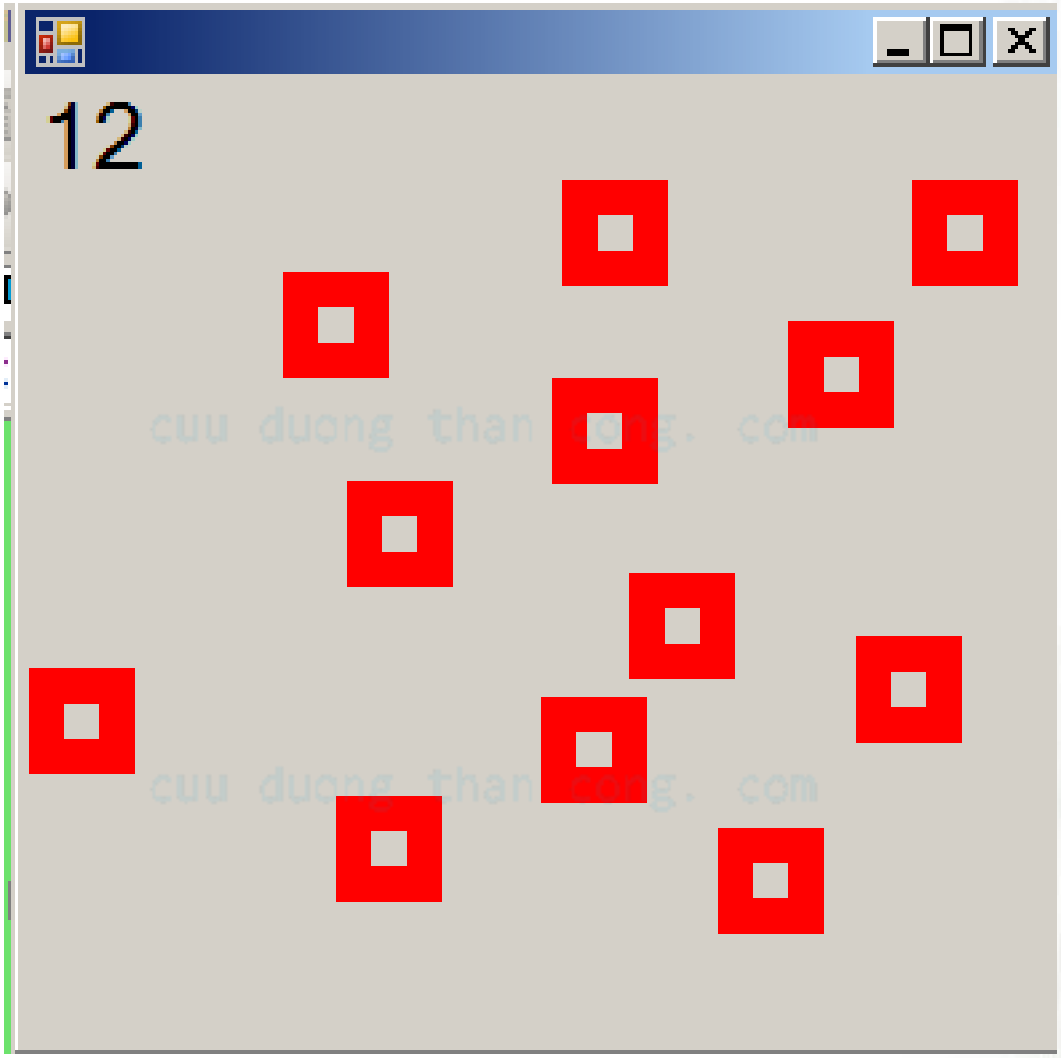
Graphics.SetClip(<path>)

Graphics.SetClip(<rectangle>)

Ví dụ



Ví dụ



Image

- **Cho phép vẽ các hình ảnh.**
  - Tạo các hình ảnh thông qua class Image (Bitmap, Metafile, Icon, …)
  - Class Bitmap hỗ trợ các định dạng chuẩn GIF, JPG, BMP, PNG, TIFF.
  - Dùng Graphics.DrawImage, DrawIcon, DrawIconUnstretched, DrawImageUnscaled
- Bitmap
  - **Bitmap bmp = new Bitmap(filename, …)**
  - RotateFlip: xoay lật, hình
  - MakeTransparent: đặt màu trong suốt.
  - GetPixel, SetPixel: vẽ bằng cách chấm từng điểm!

Vẽ chữ

- **Cho phép vẽ các câu chữ trên Graphics**
  - Tạo các đối tượng Font chỉ định các thuộc tính chữ (như font, style, …) *(chương 5)*
  - Tạo pen và brush
  - **Graphics.DrawString**
  - Để "đo" kích thước chuỗi (dài,rộng) , dùng **Graphics.MeasureString**

System.Drawing namespace

Bitmap	Pixel image (GIF, JPEG, PNG, BMP, TIFF)
Brush	Abstract base class.
Brushes	Brushes for all basic colors
Color	
Font	Defines a format for text, including font face, and sizeEncapsulates a typeface, size, style, and effects.
FontFamily	Group of type faces with the same basic design.
Graphics	
Icon	Transparent bitmaps used for Windows icons.
Image	Abstract base class common to the Bitmap, Icon, and Metafile classes.
Pen	Defines an object used to draw lines and curves.
Pens	Provides static Pen definitions for all the standard colors.

System.Drawing namespace

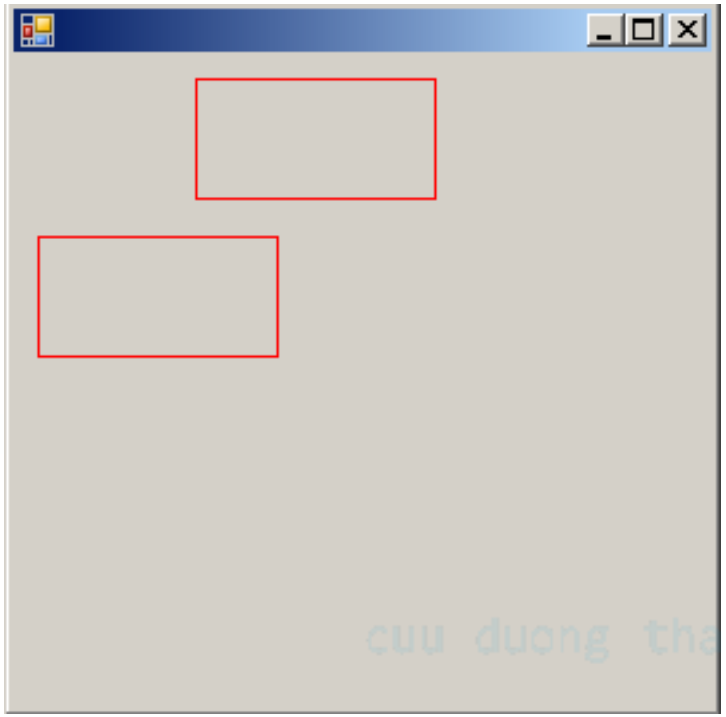
Point	Structure used to represent an ordered pair of integers. Typically used to specify two-dimensional Cartesian coordinates.
PointF	Same as Point, but uses a floating-point number (float in C#, Single in VB.NET) rather than an integer.
Rectangle	Structure that represents the location and size of a rectangular region.
RectangleF	Same as Rectangle, but uses floating-point values (float in C#, single in VB.NET) rather than integers.
Size	Structure that represents the size of a rectangular region as an order pair (Point) representing width and height.
SizeF	Same as Size, but uses PointF rather than Point.
SystemBrushes	A utility class with 21 static, read-only properties that return objects of type Brush (each of a different color).
SystemPens	A utility class with 15 static, read-only properties that return objects of type Pen (each of a different color).

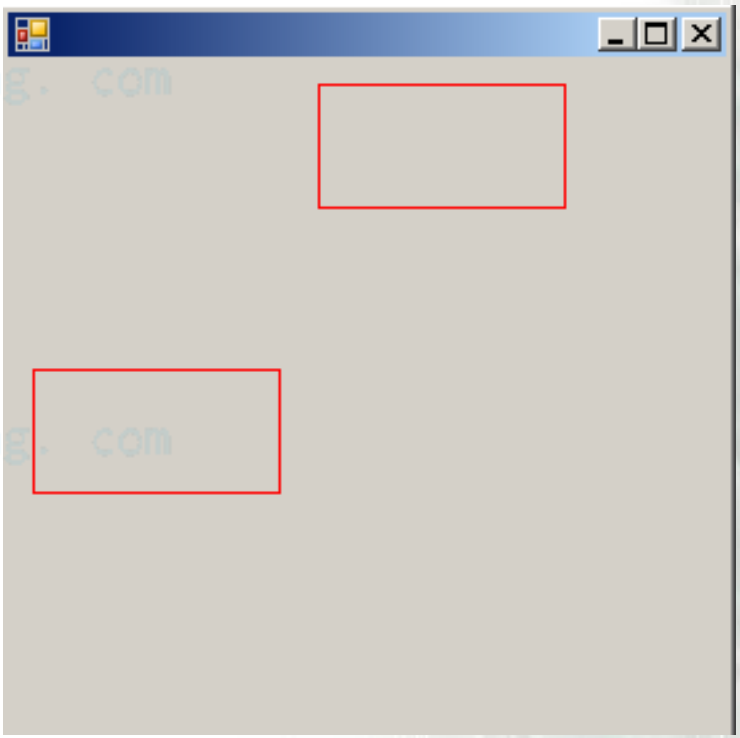
Ý tưởng tạo animation với GDI+

- **Xóa cũ - vẽ mới là sai lầm!**
- **Frame-based animation:** vẽ lại toàn bộ form theo tốc độ nhất định. Kiểm soát bằng các biến trạng thái.

```
protected int x=0;
protected int y=0;
…
private void Form1_Paint(object sender, PaintEventArgs e) {
      Graphics g = e.Graphics;
      Pen pen = new Pen(Color.Red);
      g.DrawRectangle(pen, x, 10, 100, 50);
      g.DrawRectangle(pen, 10, y, 100, 50);
}

private void timer1_Tick(object sender, EventArgs e) {
      x = (x + 1) % 200; y = (y+1) % 200;
      Refresh();
}
```





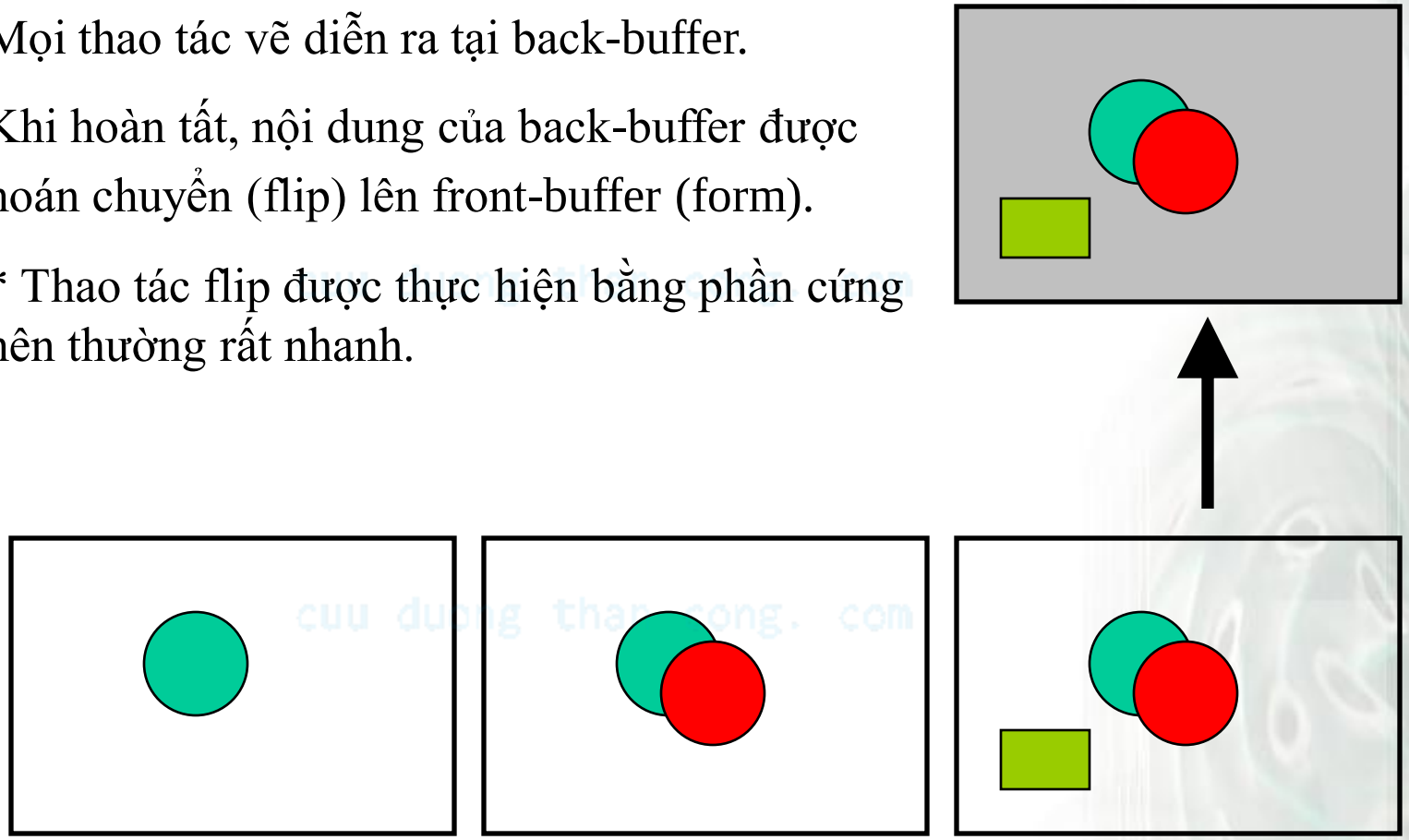
Giải quyết nháng hình bằng double-buffer

Form.DoubleBuffered = true

Mọi thao tác vẽ diễn ra tại back-buffer.

Khi hoàn tất, nội dung của back-buffer được hoán chuyển (flip) lên front-buffer (form).

* Thao tác flip được thực hiện bằng phần cứng nên thường rất nhanh.



Sprites

- Mỗi bitmap một frame => nạp hình nhiều lần, khó quản lý.
- Dùng một hình lớn chứa nhiều hình nhỏ kích thước bằng nhau (sprites)

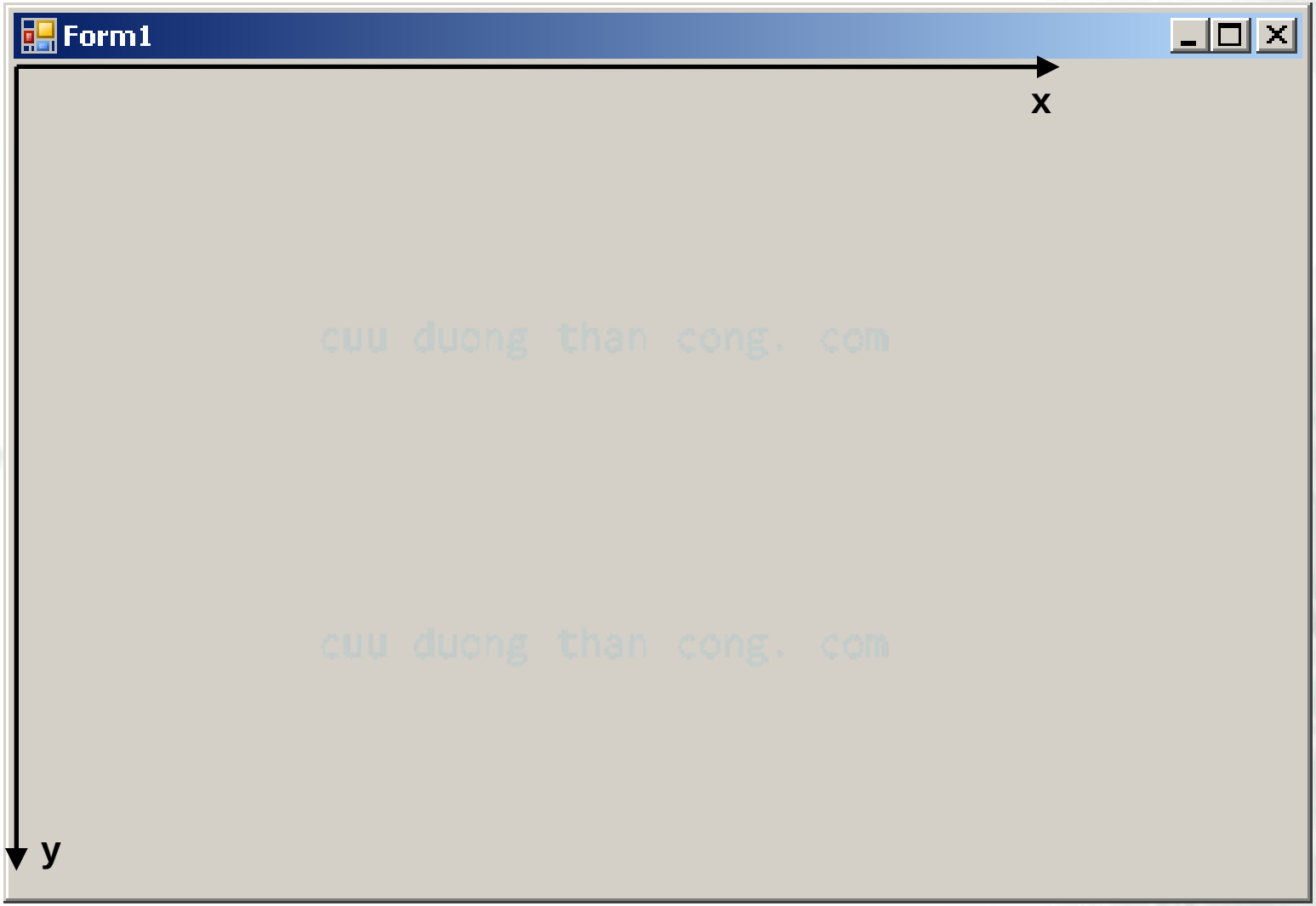
- Hàm DrawImage cho phép vẽ một phần hình chữ nhật của image lên Graphic
- Xem http://www.codeproject.com/vcpp/gdiplus/imageexgdi.asp
- để biết cách extract frames (sprites) từ animated GIF files

Transformation – biến đổi hệ trục

- Hệ trục (coordinate system)
  - Hệ trục thế giới (world coordinate system)
  - *Hệ trục trang (page coordinate system)*
  - Hệ trục thiết bị (device coordinate system)

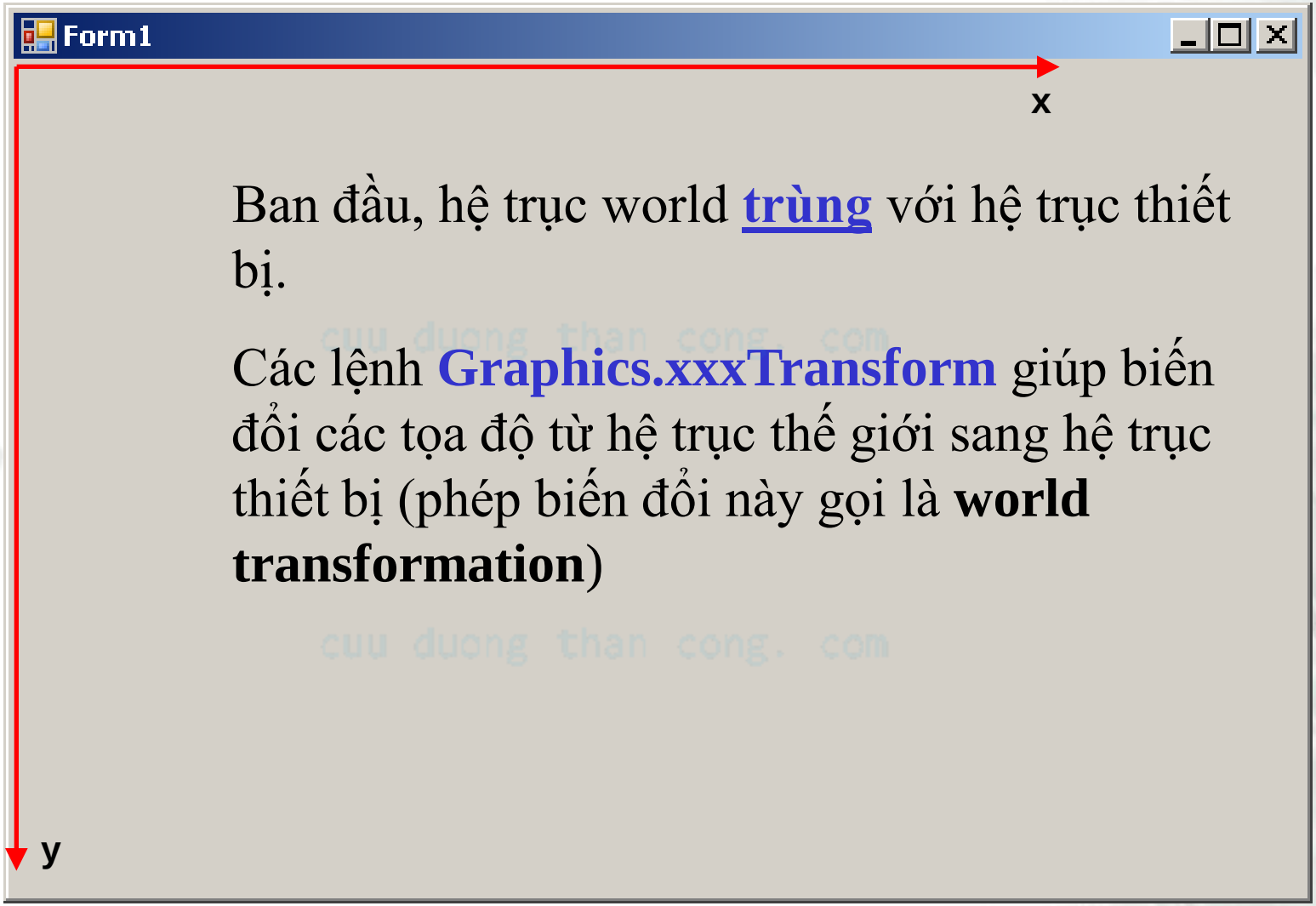
Transformation –biến đổi hệ trục

Hệ trục thiết bị - form



Transformation –biến đổi hệ trục

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

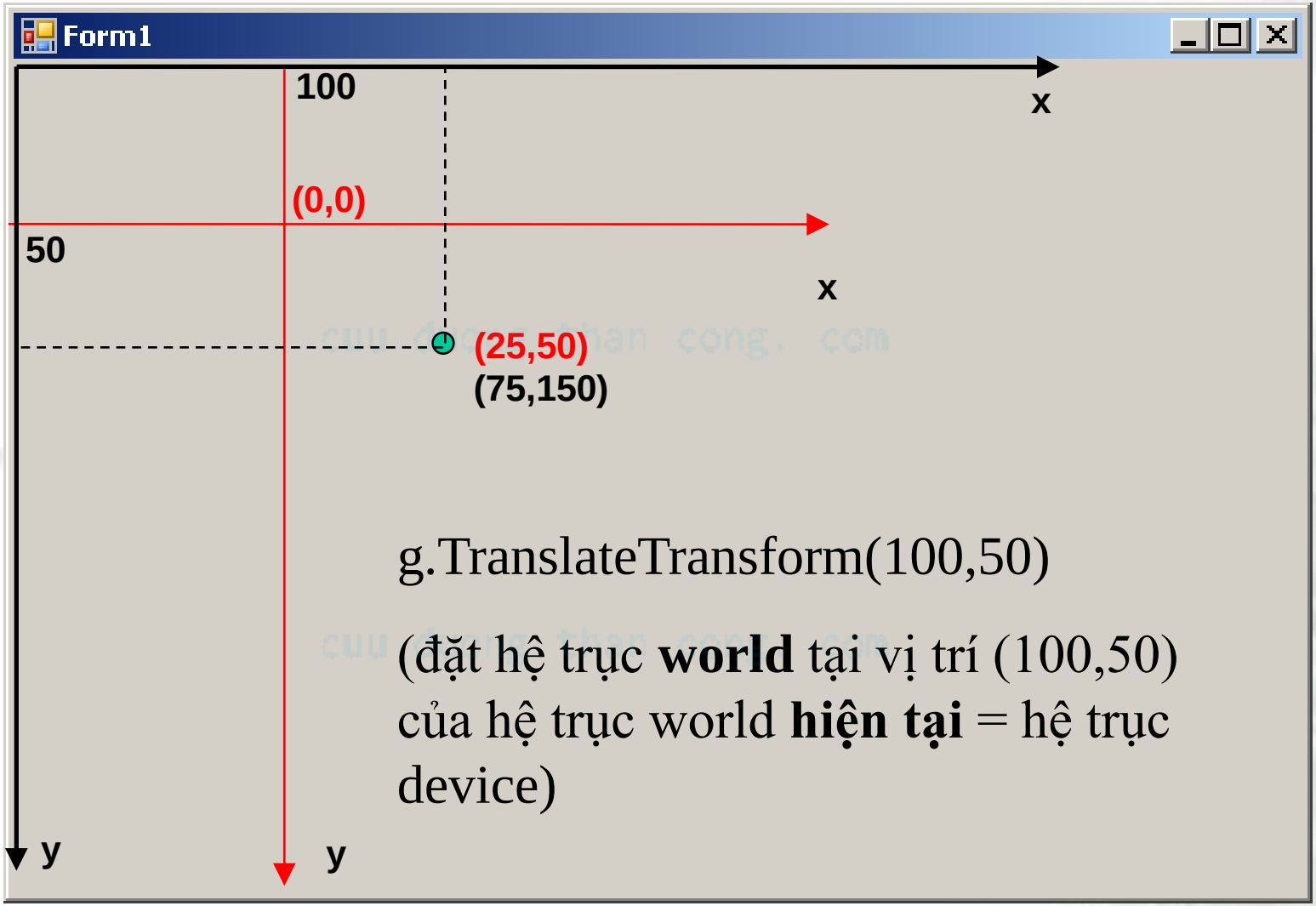


Ban đầu, hệ trục world **trùng** với hệ trục thiết bị.

Các lệnh **Graphics.xxxTransform** giúp biến đổi các tọa độ từ hệ trục thế giới sang hệ trục thiết bị (phép biến đổi này gọi là **world transformation**)

Transformation –biến đổi hệ trục

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

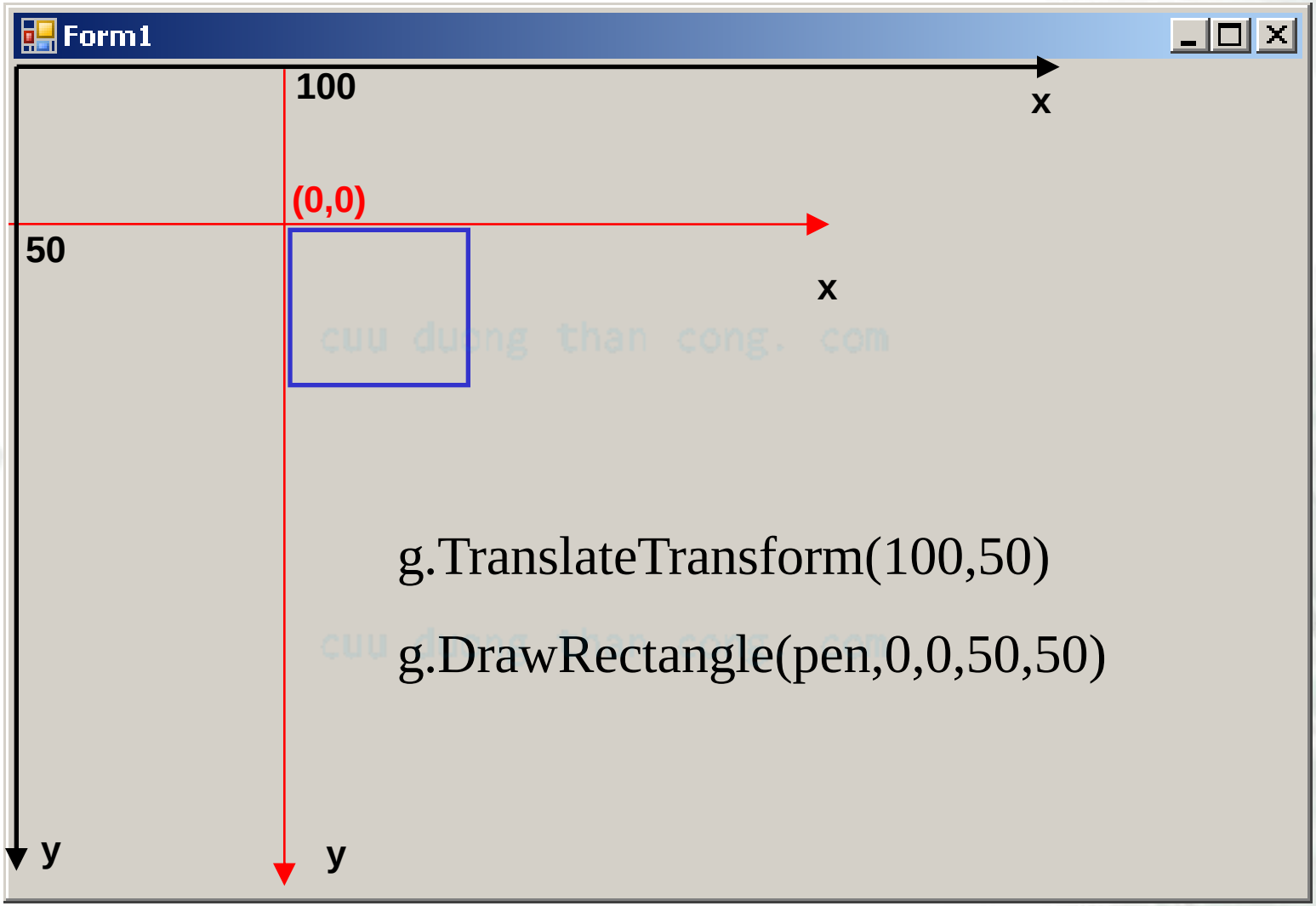


g.TranslateTransform(100,50)

(đặt hệ trục **world** tại vị trí (100,50) của hệ trục world **hiện tại** = hệ trục device)

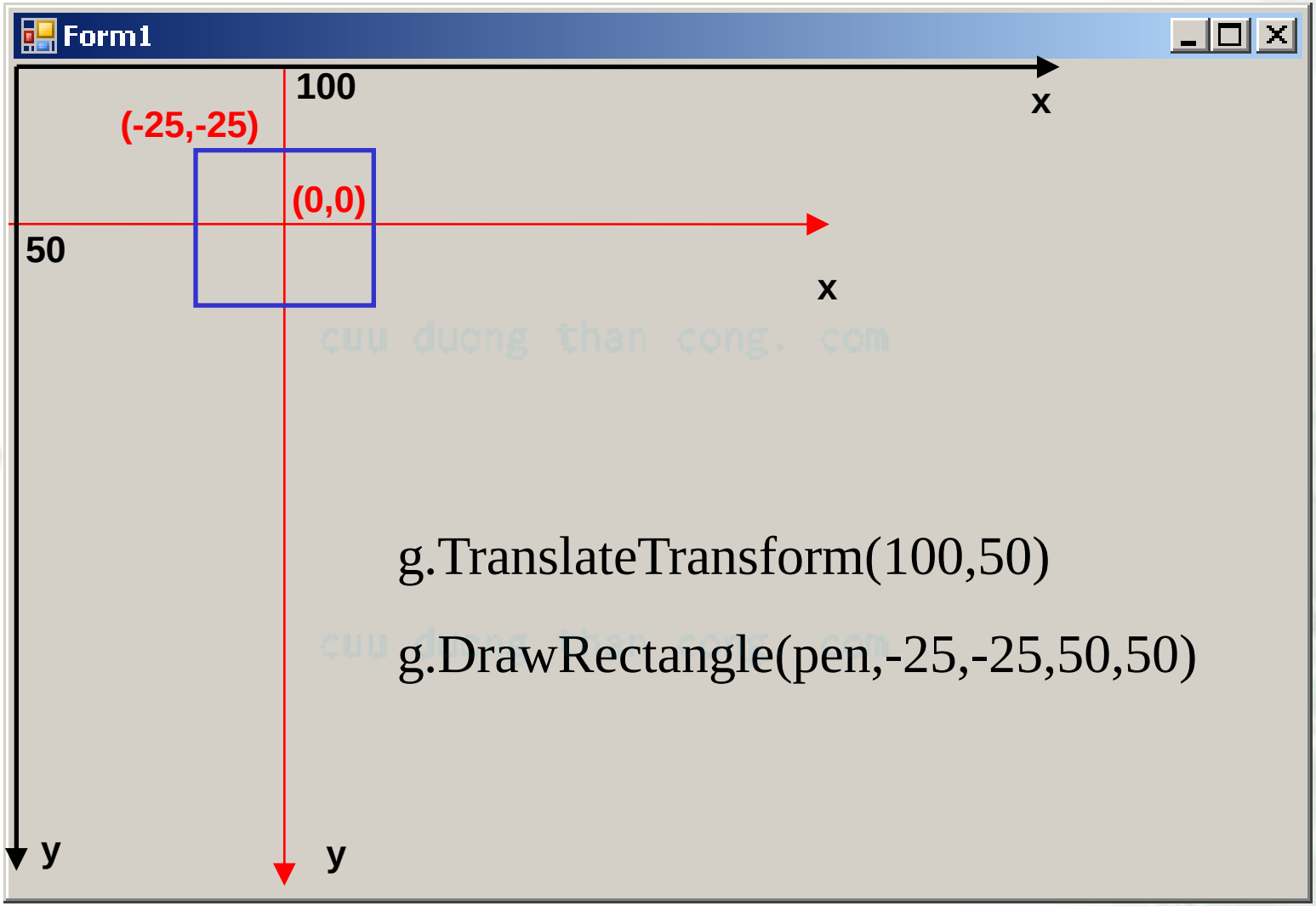
Transformation – biến đổi hệ trục

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



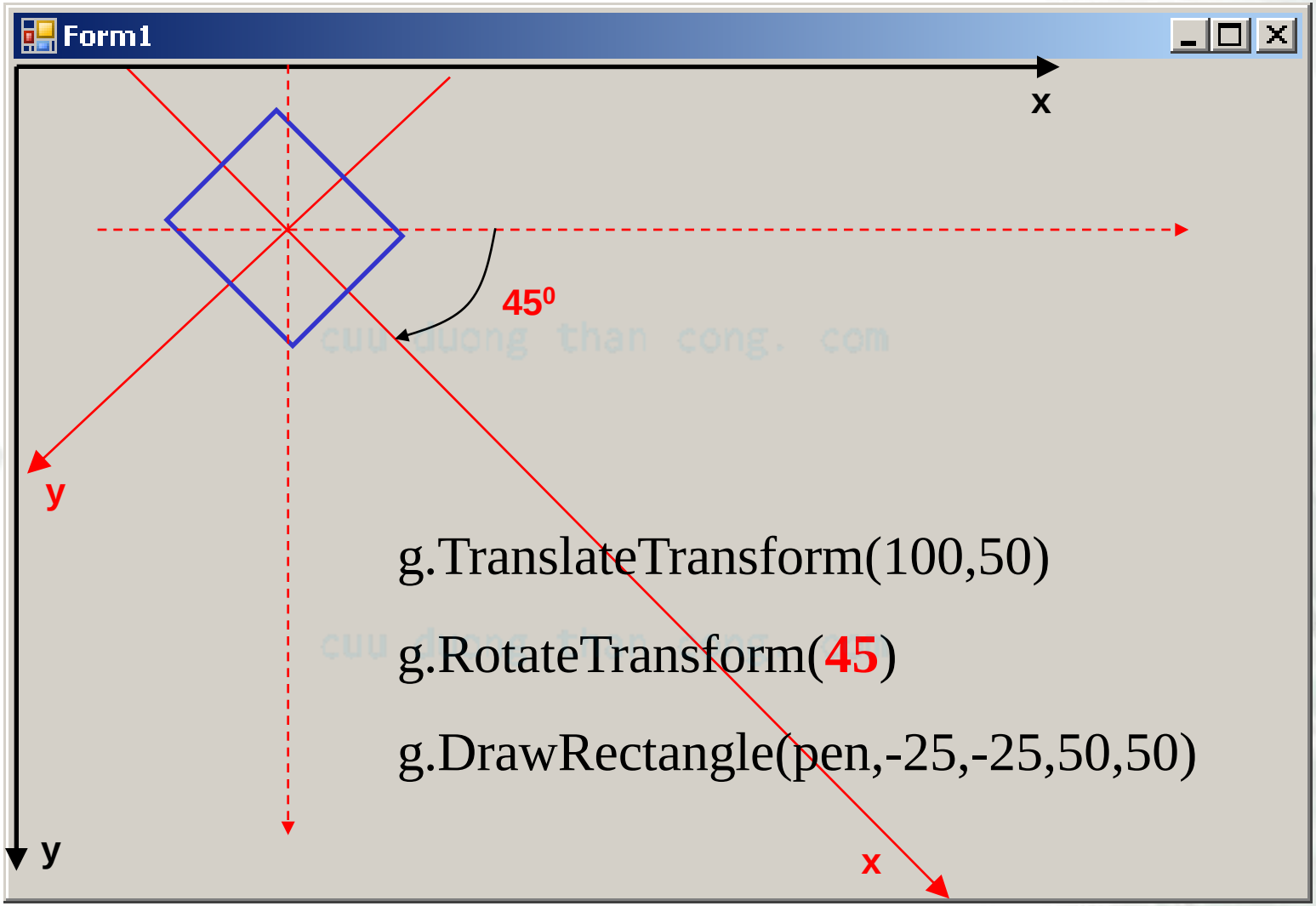
Transformation –biến đổi hệ trục

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



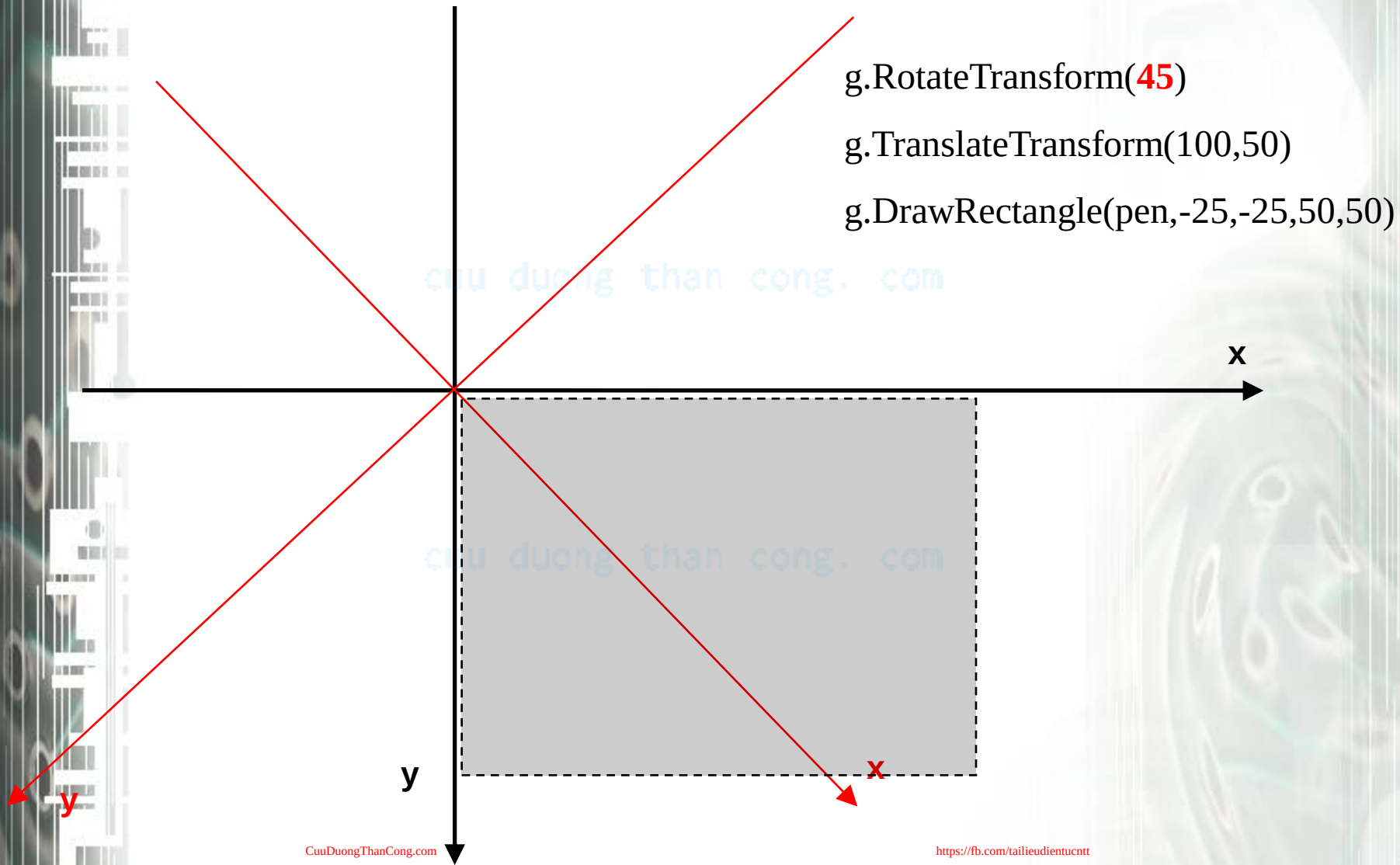
Transformation –biến đổi hệ trục

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



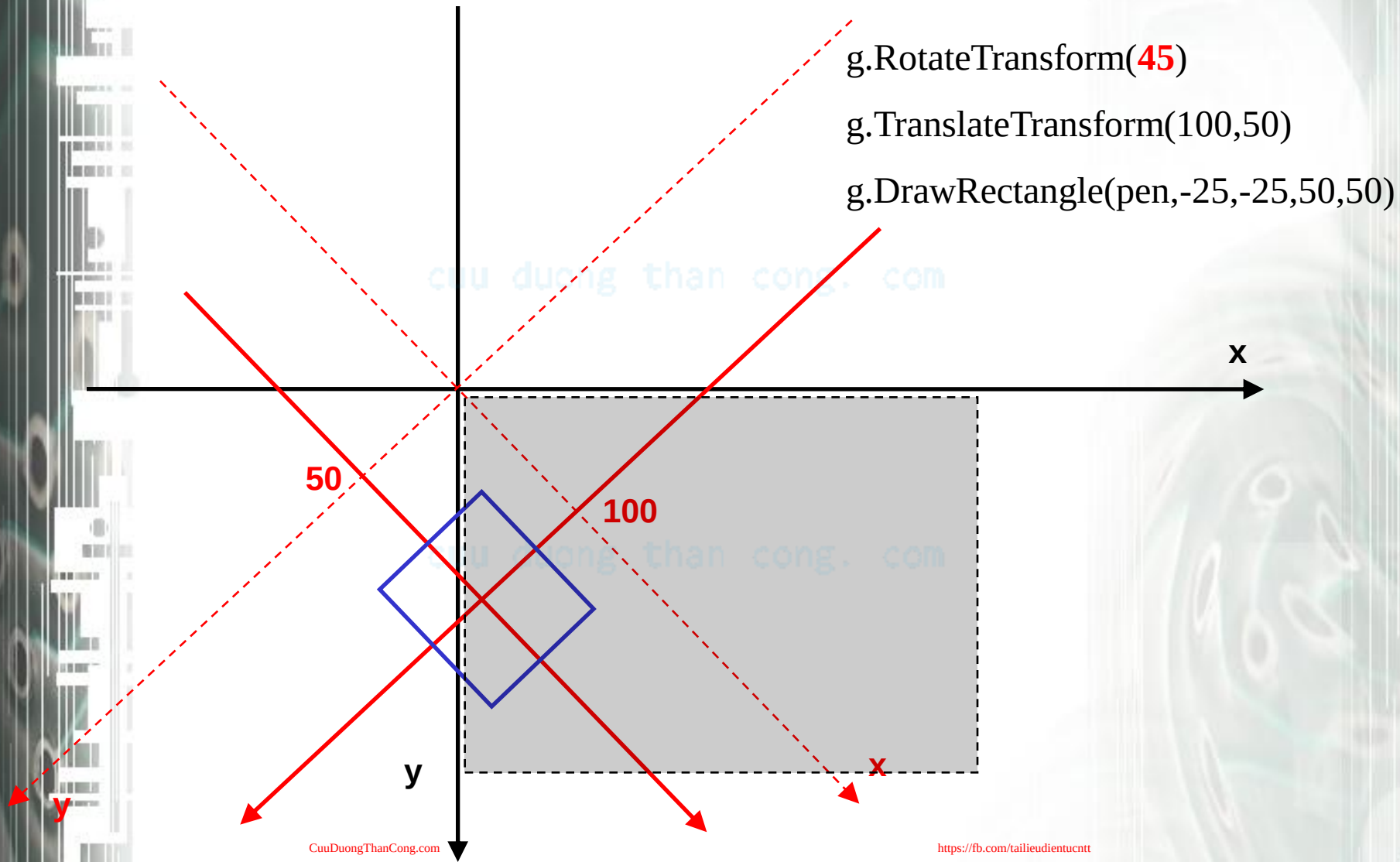
Thứ tự phép biến đổi

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



Thứ tự phép biến đổi

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



Biến đổi hệ trục bằng ma trận

- Tất cả các phép biến đổi đều được thực hiện "bên dưới" bằng ma trận.
- **Tư tưởng chính:** mọi hình đều được tạo ra từ các điểm => mọi thao tác biến đổi (dù phức tạp đến mấy) đều quy về việc chuyển đổi tọa độ của các điểm.
- Ma trận: là một bảng 2 chiều, mỗi ô là một số thực.

Đại cương về ma trận

- Cộng ma trận, cộng từng phần tử tương ứng

2	3
4	5
6	7

+

1	2
3	4
5	6

=

3	5
7	9
11	13

Đại cương về ma trận

- Nhân ma trận: (m×**n**) nhân với (**n**×p) →m × p
- Giá trị ở (i,j) = **tích vô hướng** của (hàng i của A) và (cột j của B)
- Tích vô hướng của:
  - $(a_1,a_2,a_3,\ldots,a_n) \bullet (b_1,b_2,b_3,\ldots,b_n) = (a_1*b_1 + a_2*b_2 + a_3*b_3 + \ldots + a_nb_n)$

2	3
4	5
6	7

×

1	2
3	4

=

11	

C(1,1) = (2,3) • (1,3) = 2*1 + 3*3 = 11

Đại cương về ma trận

- Nhân ma trận: (m×**n**) nhân với (**n**×p) →m × p
- Giá trị ở (i,j) = **tích vô hướng** của (hàng i của A) và (cột j của B)
- Tích vô hướng của:
  - $(a_1,a_2,a_3,\ldots,a_n) \bullet (b_1,b_2,b_3,\ldots,b_n) = (a_1*b_1 + a_2*b_2 + a_3*b_3 + \ldots + a_nb_n)$

2	3
4	5
6	7

×

1	2
3	4

=

11	16

C(1,2) = (2,3) • (2,4) = 2*2 + 3*4 = 16

Đại cương về ma trận

- Nhân ma trận: (m×**n**) nhân với (**n**×p) →m × p
- Giá trị ở (i,j) = **tích vô hướng** của (hàng i của A) và (cột j của B)
- Tích vô hướng của:
  - $(a_1,a_2,a_3,\ldots,a_n) \bullet (b_1,b_2,b_3,\ldots,b_n) = (a_1*b_1 + a_2*b_2 + a_3*b_3 + \ldots + a_nb_n)$

2	3
4	5
6	7

×

1	2
3	4

=

11	16
19	

C(2,1) = (4,5) • (1,3) = 4*1 + 5*3 = 19

Biến đổi tọa độ điểm bằng ma trận

- Một điểm là ma trận 1×2 (P)
- Phép biến đổi là ma trận: 2×2 (T)
- Biến đổi: P’ = P × T

5	10

×

1	0
0	-1

=

5	-10

Lật (flip) theo chiều đứng

Biến đổi tọa độ điểm bằng ma trận

- Một điểm là ma trận 1×2 (P)
- Phép biến đổi là ma trận: 2×2 (T)
- Biến đổi: P’ = P × T

5	10

×

0	1
-1	0

=

-10	5

Xoay 90^0

Biến đổi tọa độ điểm bằng ma trận

- Một điểm là ma trận 1×2 (P)
- Phép biến đổi là ma trận: 2×2 (T)
- Biến đổi: P’ = P × T

3	2

×

3	0
0	1

=

9	2

Dãn trục x 3 lần

Biến đổi tọa độ điểm bằng ma trận

- **Vấn đề:** không thể biểu diễn phép translate (dịch chuyển) điểm bằng nhân ma trận.
- Giải quyết:
  - Mở rộng ma trận biến đổi thành 3×3, thêm một hàng để chứa ma trận transalate, thêm một cột dummy (0,0,1).
  - Thêm một thành phần "rỗng" (dummy) có giá trị 1 vào tọa độ điểm.

x	y	1

×

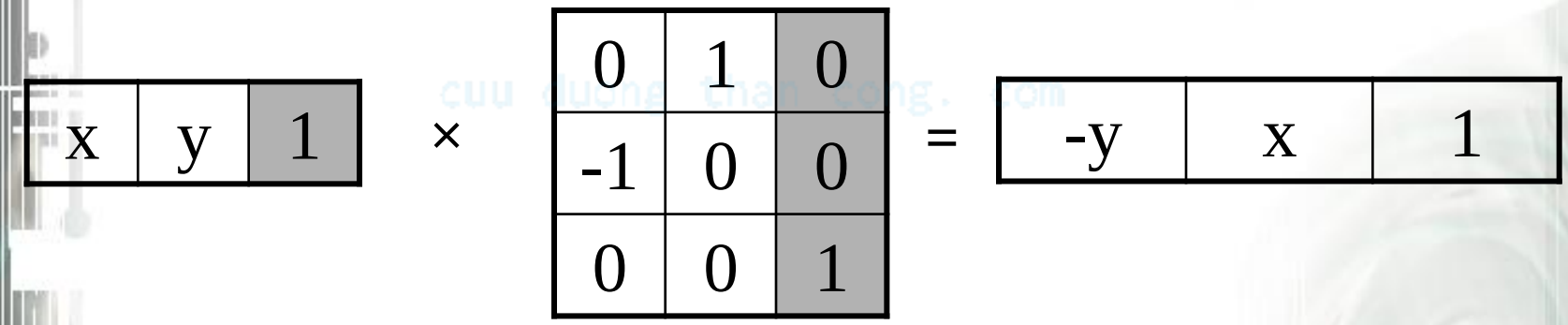
0	0	0
0	0	0
10	20	1

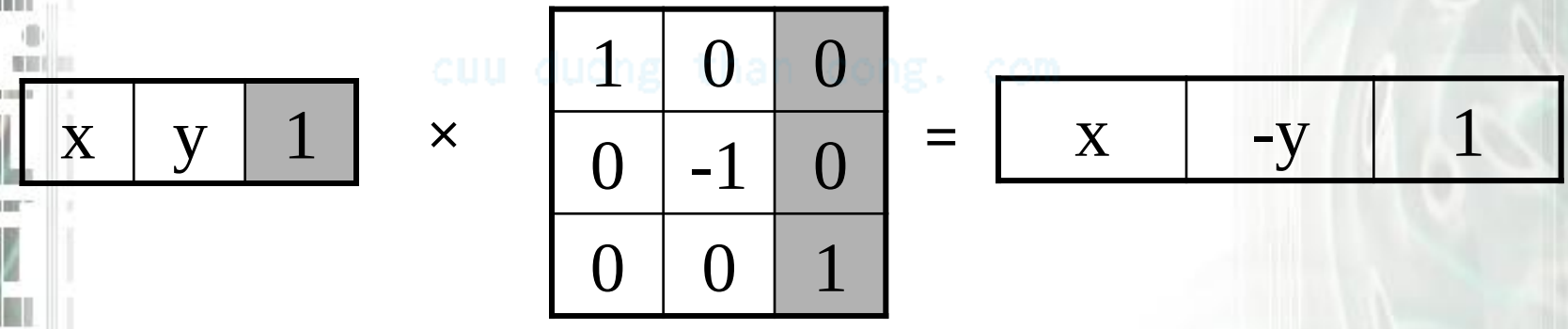
=

x+10	y+20	1

Biến đổi tọa độ điểm bằng ma trận

- Các phép biến đổi khác vẫn được bảo toàn





Biến đổi tọa độ điểm bằng ma trận

- Tại sao dùng nhân ma trận mà không tính toán trực tiếp?
- Ma trận có khả năng "ghép" nhiều phép biến đổi làm 1.
- Để làm 100 phép biến đổi cùng lúc, chỉ cần tính tích của 100 ma trận biến đổi, sau cùng nhân ma trận của điểm và ma trận tích
- Như vậy vẫn phải nhân nhiều lần????
- **Đừng quên:** một hình có nhiều **điểm**

Matrix class

- Lớp Matrix của GDI+ có sẵn tất cả các phương thức cần thiết để thao tác trên ma trận biến đổi.
  - Multiply: nhân một ma trận biến đổi với ma trận hiện tại
  - Scale: nhân một ma trận dãn với ma trận hiện tại
  - Shear: nhân một ma trận kéo với ma trận hiện tại
  - Translate: nhân một ma trận dịch chuyển với ma trận hiện tại
  - Rotate: nhân một ma trận xoay với ma trận hiện tại
  - RotateAt: nhân một ma trận xoay quanh một tâm định trước với ma trận hiện tại.
  - Reset: đặt ma trận về ma trận đơn vị.
- Sau khi tính toán ma trận biến đổi, "áp dụng" ma trận bằng:
  - Graphics.Transform = <matrix>

Biến đổi cục bộ

- Các biến đổi hệ trục có tính toàn cục (có tác dụng trên tất cả đối tượng). Để áp dụng cục bộ trên một đối tượng, dùng **GraphicsPath.Transform(matrix)**
- Biến đổi cục bộ thường được dùng để tạo chuyển động cục bộ của một thành phần của một đối tượng. Chẳng hạn, nòng súng của một chiếc xe tăng.