Graphics - Custom Controls - Sprite 2D

KHOA CÔNG NGHỆ THÔNG TIN
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

Nội dung

- Graphics (GDI+)
- Custom Controls
- Sprite 2D

Nội dung

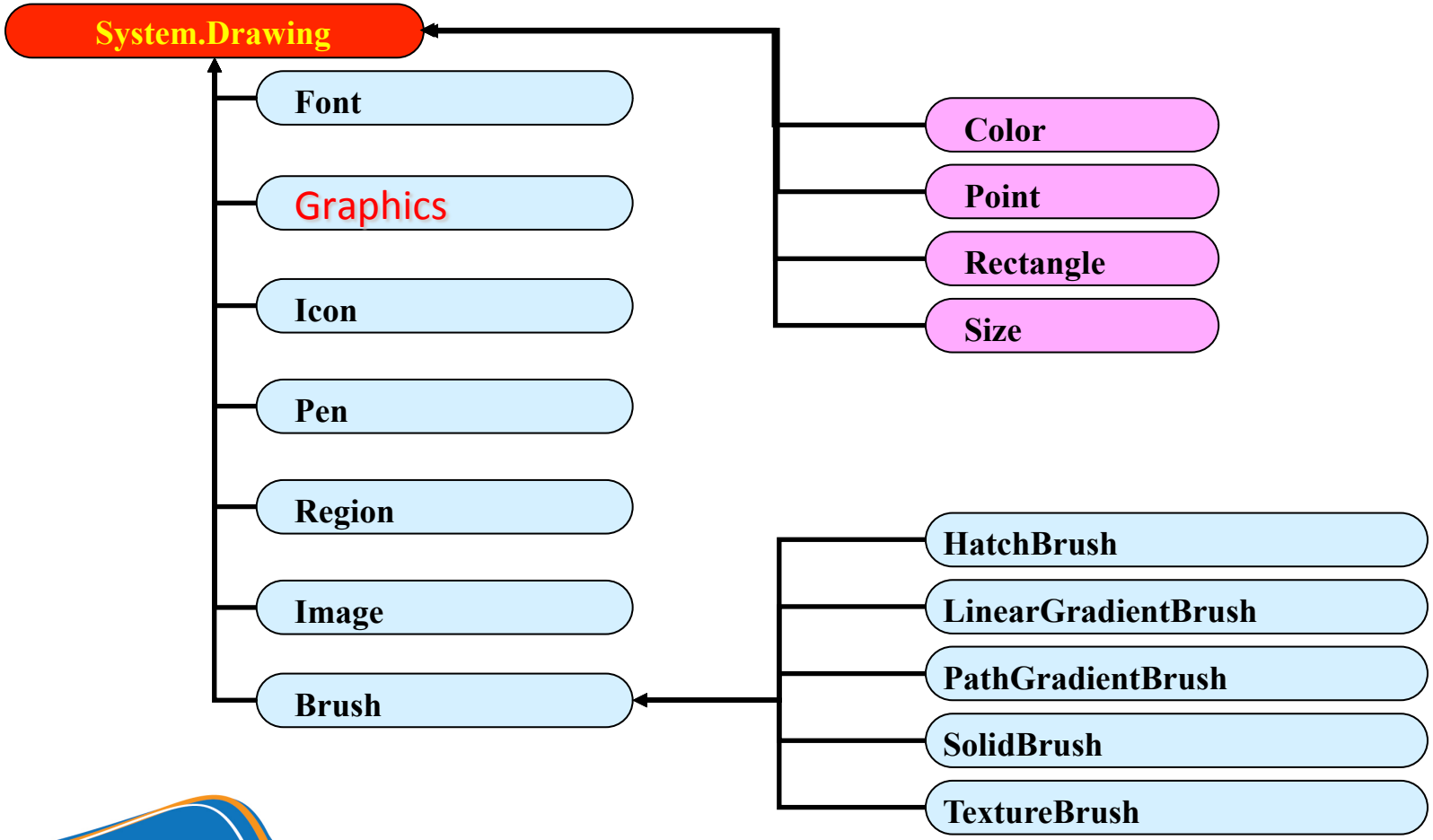
- **Graphics (GDI+)**
- Custom Controls
- Sprite 2D

Graphics (GDI+)

- Là thư viện đồ họa chính trong .NET framework.
- Cung cấp các phương thức cần thiết cho việc vẽ lên thiết bị (device) thể hiện hình ảnh.
- GDI+ tóm lược trong 3 chức năng chính
  - Đồ họa vector 2 chiều: cho phép vẽ đường thẳng, đường cong, hình khối,…
  - Hình ảnh: cho phép dựng hình (render) bitmap lên bề mặt hiển thị và đồng thời thực hiện một vài thao tác trên hình ảnh (như co giãn hình, xoay hình,…)
  - Trình bày bản in: cho phép dựng hình ký tự, văn bản mượt mà, khử răng cưa với các định dạng khác nhau về font, size, colors và orientations.

System.Drawing

- Là Namespace chứa các lớp thao tác vẽ.



Đối tượng Graphics

- Lớp Graphics thể hiện
  - "***Abstract***" drawing surface
  - Tập hợp những "***tool***" cho phép thao tác trên surface đó
- Để lấy đối tượng **Graphics**
  - Sử dụng thuộc tính Graphics được truyền cho OnPaint()
  - Sử dụng phương thức CreateGraphics() của control
  - Lấy từ đối tượng dẫn xuất từ Bitmap
- Để ép buộc sự kiện Paint xảy ra thì sử dụng phương thức Invalidate().

Lấy đối tượng Graphics

- Lấy từ sự kiện Paint

```
void gbPaint_Paint(object sender, PaintEventArgs e)
{
    Graphics g = e.Graphics;
    g.DrawLine(pen, pStart, pEnd);
}
```

- Lấy từ phương thức CreateGraphics()

```
using (Graphics g = gbPaint.CreateGraphics())
{
    g.DrawLine(pen, pStart, pEnd);
}
```

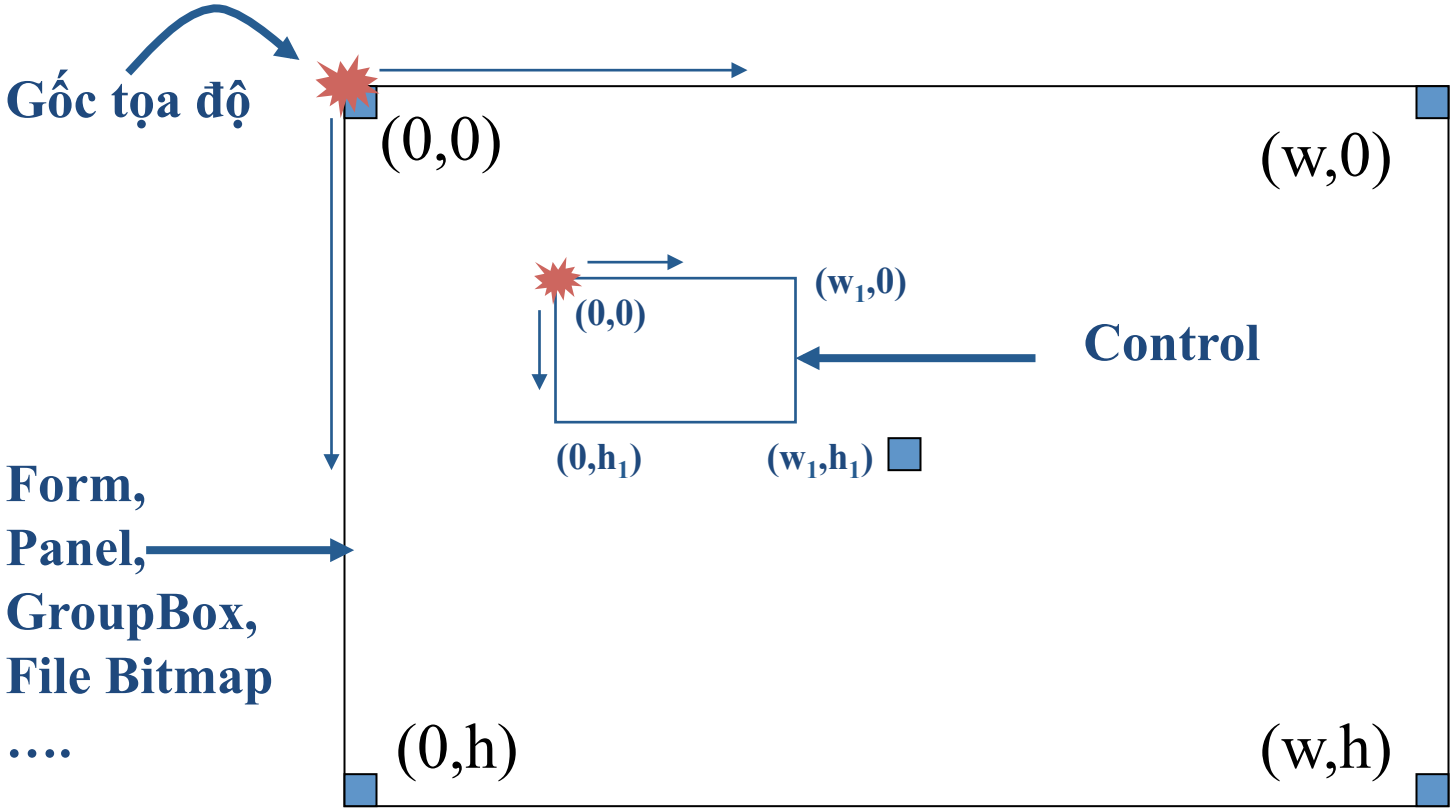
Lấy đối tượng Graphics

- Lấy từ Bitmap

```
Bitmap bmpTemp = new Bitmap(800, 600);
using (Graphics g = Graphics.FromImage(bmpTemp))
{
    g.DrawLine(pen, pStart, pEnd);
}
```

Hệ thống tọa độ

- Tọa độ cục bộ hay toàn cục



Font, Color, Pen, Brush

- Font: cần chú ý đến Size (phụ thuộc vào unit), Style (đậm, nghiêng,…), Family,…
- Color: có nhiều Color được tạo sẵn hoặc có thể tạo Color tùy ý từ phương thức tĩnh FromArgb().
- Pen: để vẽ nét, cần chú ý đến Width, Style, DashStyle, LineJoins,...
- Brush: để tô, lấp đầy,… cần chú ý đến Color, Type (Solid, Hatch, LinearGradient, Texture,…).

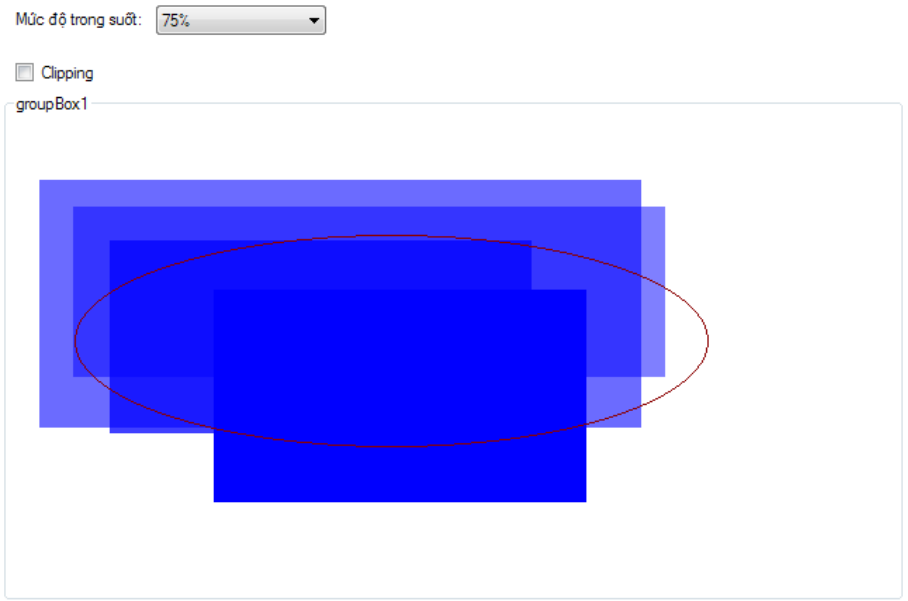
Vẽ chuỗi ký tự

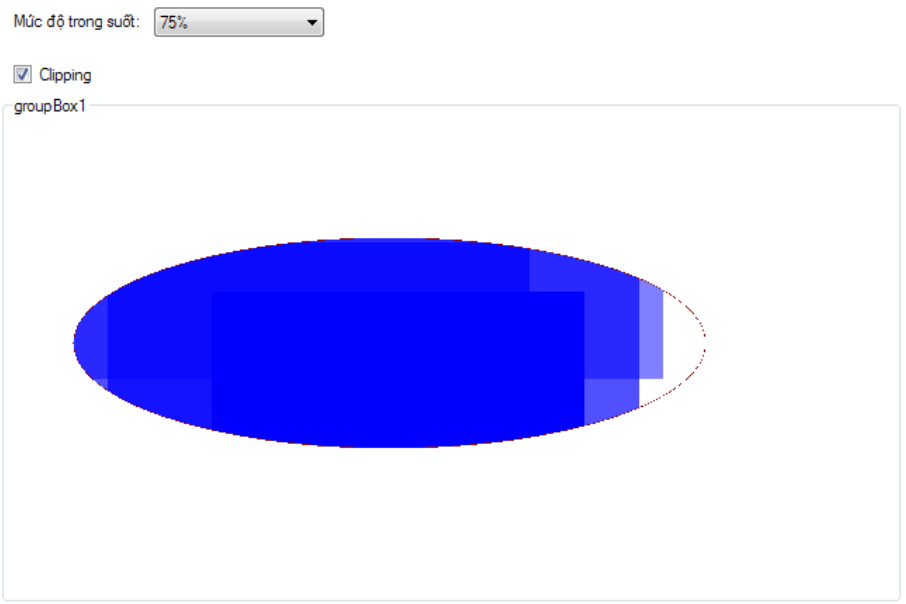
- Sử dụng phương thức DrawString() của lớp đối tượng Graphics.
- Cần các đối tượng chủ yếu là: Font, Pen, Brush, Location và StringFormat.
- Ví dụ:

```
StringFormat stringFormat = new StringFormat();
stringFormat.Alignment = StringAlignment.Center;
g.DrawString("Sample Text", Font, Brushes.Black,
picBPaint.ClientRectangle, stringFormat);
```

Alpha Blending, Clipping

- Cho phép vẽ với thuộc tính Alpha (độ trong suốt) của màu nền.
- Clipping cho phép giới hạn lại bề mặt vẽ trong một vùng cụ thể (mặc định là bao phủ tất cả bề mặt vẽ).





Transform, DoubleBuffer

- Cung cấp các phương thức biến đổi cơ bản cho hệ trục tọa độ (tịnh tiến, co giãn, xoay).
- Double Buffer là phương pháp sử dụng bộ nhớ làm bề mặt vẽ tạm (là đối tượng Bitmap trong bộ nhớ) để thực hiện mọi thao tác vẽ, sau đó mới thực hiện vẽ một lần ra bề mặt vẽ thật sự. Điều này sẽ giúp giảm thiểu vấn đề nháy hình khi phải thực hiện vẽ quá nhiều trong 1 lần refresh.
  - Lưu ý: chỉ có Form hay Panel mới truy suất được thuộc tính DoubleBuffered để sử dụng tính năng có sẵn của .NET framework.

Nội dung

- *Graphics (GDI+)*
- **Custom Controls**
- Sprite 2D

Controls

- Trong **Windows Forms**, mọi thứ đều được bắt đầu từ class **Control** (định nghĩa các chức năng tối thiểu mà mọi **control** đều cần phải có).
- Chức năng chính của **Control**
  - Cách thức hiển thị (vị trí, màu nền,…)
  - Cách thức tương tác, gắn kết (sự kiện, phím nóng,…)
  - Cách thức hành xử (thực hiện chức năng,…)
- Trong .NET Framework có rất nhiều các Control dựng sẵn theo từng nhóm nhu cầu cụ thể như: Common, Container, Menu,…

Custom Controls

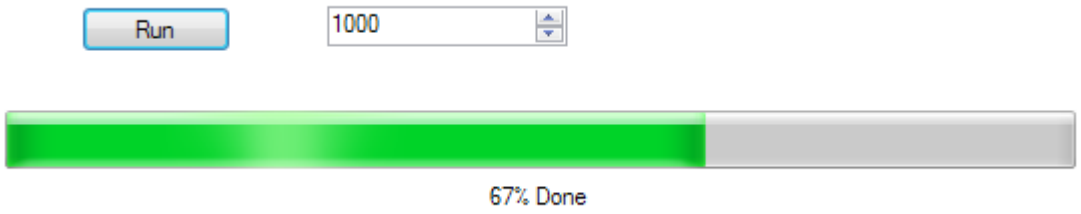
- Là một chủ đề quan trọng trong lập trình Windows Forms.
- Giúp cải thiện đóng gói, đơn giản hóa mô hình lập trình và giúp cho giao diện có tính mở hơn, bóng bẩy hơn, hiện đại hơn.
- Nhìn chung khi các Control có sẵn không đáp ứng đủ như cầu hay có một nhóm control cần sử dụng lại thì giải pháp Custom Control là hiệu quả.

Các loại Custom Controls

- **User Controls**: đây là kiểu đơn giản nhất, kế thừa từ class **System.Windows.Forms.UserControl** và làm theo một mô hình thành phần (composition). Thông thường User controls kết hợp nhiều thành phần các controls theo một trình tự định sẵn.
- **Derived Controls**: xây dựng control kế thừa từ 1 control có sẵn và thay đổi hoặc thêm vào một số thuộc tính hay phương thức theo nhu cầu.
- **Owner-drawn Controls**: kế thừa từ class Control, tự xây dựng cách thức vẽ cũng như các thuộc tính và phương thức thích hợp.

User Controls

- Đây là kiểu custom control đơn giản nhất, mục tiêu là kết hợp các controls có sẵn theo một cách thức hoạt động logic mong muốn.
- Ví dụ:



Derived Controls

- Xây dựng controls dựa vào việc kế thừa từ các controls sẵn có là cách lý tưởng để có các chức năng sẵn có và mở rộng thêm.
- Có thể có nhiều sự khác biệt hay chỉ thêm một số chức năng so với controls gốc.
- Ví dụ: TextBox chỉ cho phép nhập chữ số

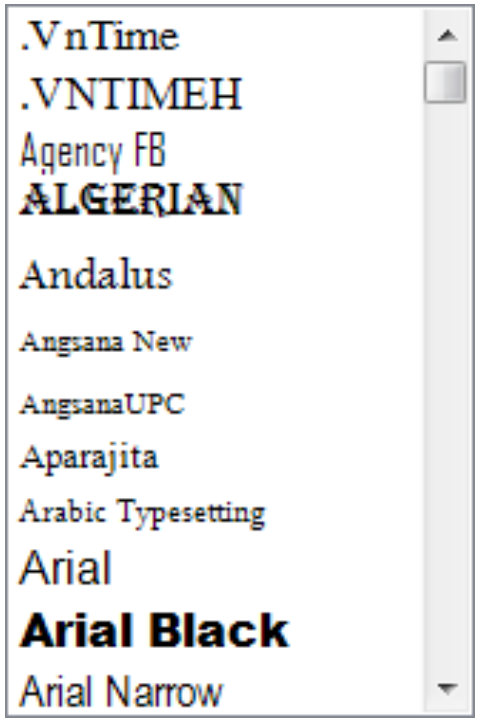
```
public class NumericTextBox : TextBox
{
    protected override void OnKeyPress(KeyPressEventArgs e)
    {
       if (!char.IsControl(e.KeyChar) & !char.IsDigit(e.KeyChar))
            e.Handled = true;
        base.OnKeyPress(e);
    }
}
```

Owner-drawn Controls

- Mục tiêu chính của việc xây dựng lại controls dạng này là cần sự thể hiện hình ảnh khác với mặc định (tập trung vào việc vẽ lại controls).
- Một số controls dựng sẵn trong .NET có hỗ trợ việc tự vẽ thể hiện (owner drawing) như: ListBox, ComboBox, ListView, TreeView, ToolTip, MenuItem.
- Ngoài ra có thể tự xây dựng hoàn toàn dạng controls này bằng cách kế thừa từ class Control.

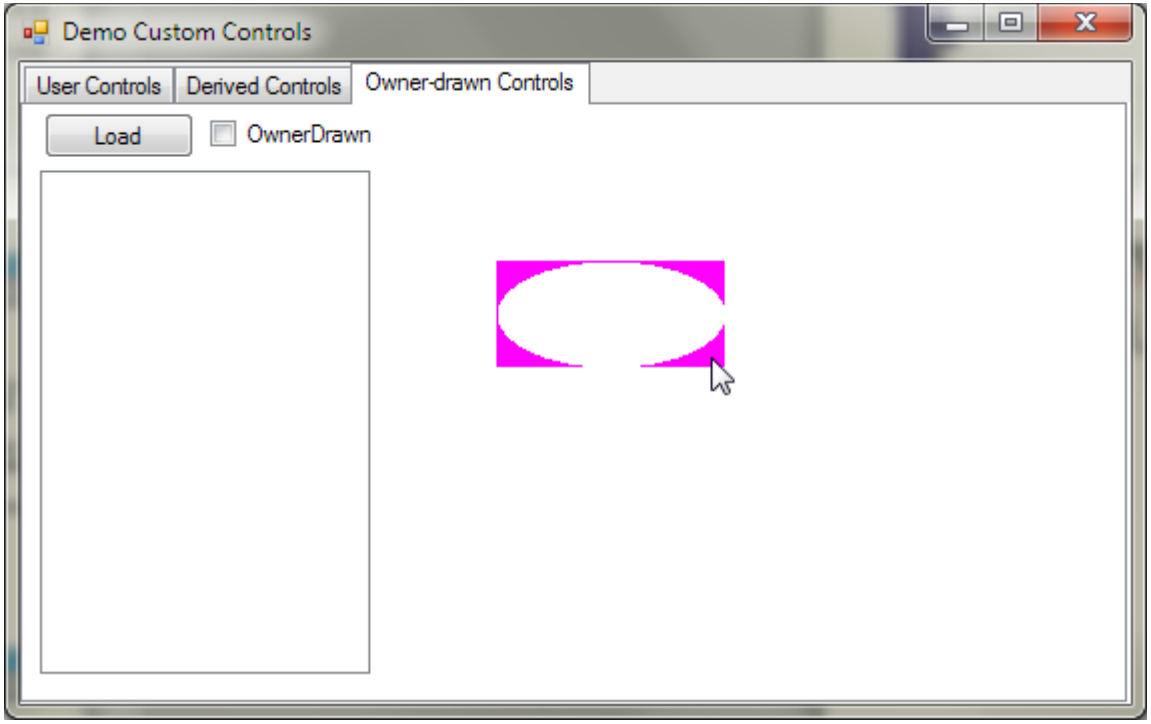
Owner-Drawn ListBox

- Lưu ý để sử dụng được chức năng tự vẽ thì phải set thuộc tính DrawMode sang `OwnerDrawVariable`.



DragDrop Control

- Xây dựng Control với hình dạng tự vẽ và đồng thời có chức năng thay đổi vị trí theo thao tác của mouse (kéo thả).



Nội dung

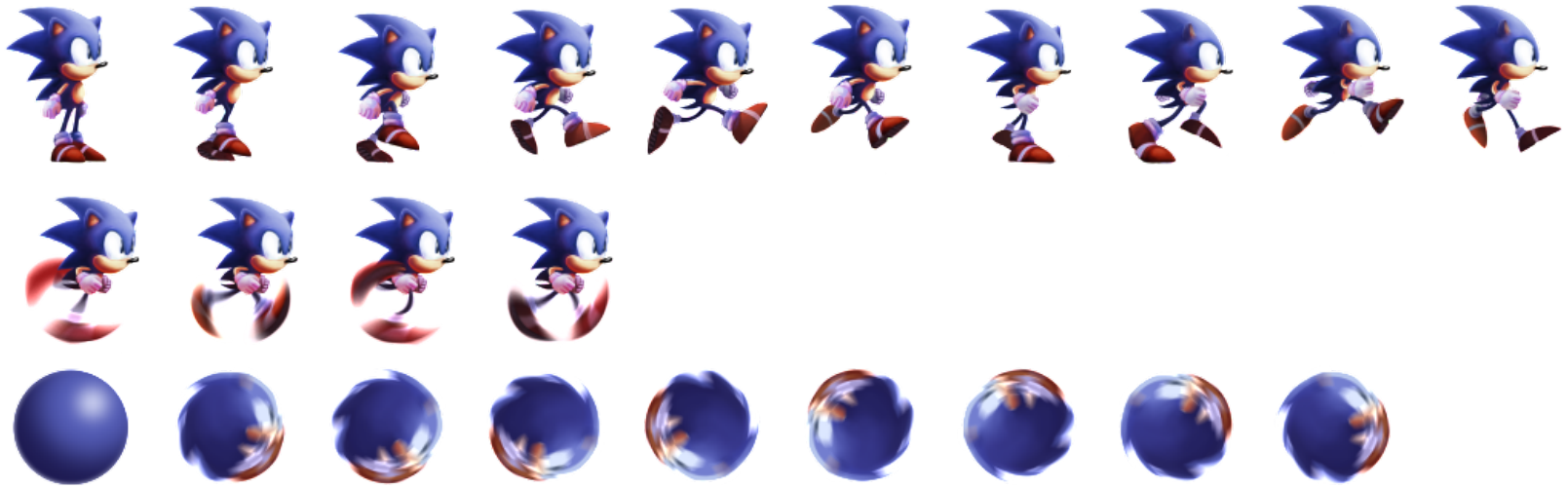
- *Graphics (GDI+)*
- *Custom Controls*
- **Sprite 2D**

Animation (2D)

- Thực hiện hình ảnh động (animation) để thể hiện được trực quan sinh động hơn trạng thái cảu đối tượng mà thông thường là áp dụng trong việc xây dựng trò chơi (game).
- Ý tưởng từ phim ảnh đó là kết hợp nhiều khung hình tĩnh (thể hiện hành động thay đổi theo thời gian) với nhau trong một khoảng thời gian hợp lý với cảm nhận của con người (thông thường là 24 hình trong 1 giây), tức là vẽ tuần tự từng ảnh một theo thời gian đủ nhanh.
- Như vậy để có được animation cho nhiều hành động thì phải quản lý khá nhiều ảnh tĩnh nên cần có phương pháp hiệu quả.

Sprite (2D)

- Sprite hiểu đơn giản là chuỗi các hình ảnh cùng thể hiện một hay nhiều hành động của đối tượng được ghép chung vào 1 ảnh lớn đại diện cho đối tượng đó.
- Ví dụ:



Xây dựng class Sprite (2D)

- Các thông tin cần có là: Bitmap giữ hình ảnh sprite, kích thước Frame, số lượng Frame, vị trí của sprite,…
- Các phương thức cần có là: cập nhật trạng thái, cập nhật vị trí, thực hiện chuyển bước kế tiếp, thực hiện vẽ, …
- Ví dụ: (demo)

Thắc mắc?