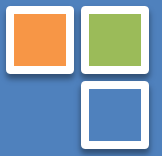


C# 입문 : 이론과 실습

제 12장 그리기

목차

그리기 개요

- Graphics(System.Drawing.Graphics) 클래스
  - System.Drawing 네임스페이스에 포함
  - 선, 사각형, 타원 등과 같은 도형을 그리는데 필요한 기본적인 메소드들이 존재

- 그래픽 객체
  - 도형을 그리기 위해 필요한 Graphics 클래스의 객체
  - 그리기판이 되는 대상
  - 객체를 얻거나 생성하는 방법
    ① Paint 이벤트 처리기의 매개변수
    ② Control 클래스의 CreateGraphics() 메소드
    ③ Graphics.FromImage() 메소드

Paint 이벤트의 매개변수

- Paint 이벤트의 매개변수
  - Paint 이벤트: 폼을 다시 그려야 할 때 발생하는 이벤트.
  - 처리기의 두 번째 매개변수에 그래픽 객체가 들어 있음.
  - 여기에 그리기 작업을 하면 폼에 그려짐.

- Paint 이벤트 처리기의 메소드 형식

```
private void Form1_Paint(object sender, System.Windows.Forms.PaintEventArgs e)
{
        Graphics g = e.Graphics;
        // 그래픽 객체를 이용한 그리기 작업
        // ...
}
```

- PaintEventArgs 클래스
  - Graphics 프로퍼티: 그리기에 필요한 Graphics 클래스의 객체
  - ClipRectangle 프로퍼티는 새로 그려야 하는 영역

CreateGraphics() 메소드

- Control 클래스의 CreateGraphics() 메소드
  - Paint 이벤트 처리기가 아닌 다른 곳에서 그리기를 하고자 할 때 사용.
  - 메소드를 사용하여 그래픽 객체를 생성
  - Control 클래스의 메소드이기 때문에 파생된 모든 클래스에서 그래픽 객체를 만들 수 있음.

- 사용 방법

```
private void DrawPrivateObject() {
    // 단계 1: 그래픽 객체를 생성한다.
    Graphics g = CreateGraphics();
    // ...
    // 단계 2: 그래픽 객체를 이용하여 그리기를 한다.
    // ...
}
```

Graphics.FromImage() 메소드

- Graphics.FromImage() 메소드
  - 이미지에서 만들어진 그래픽 객체이기 때문에 그리기 작업의 결과가 화면에 곧바로 나타나지 않음.
  - 결과를 화면에 표시하려면 이미지 객체를 화면에 출력해야 함.
  - 그리기 도중에 발생할 수 있는 깜박거림을 제거하기 위해 사용.
- 사용 방법

```
private void DrawOffScreenImage() {
    // 단계 1: 이미지 객체로부터 그래픽 객체를 생성한다.
    Image img = new Bitmap(w, h);
    Graphics bg = Graphics.FromImage(img);
    // ...
    // 단계 2: bg에 그리기 작업을 한다.
    // ...
    // 단계 3: 결과 이미지를 표시하기 위한 그래픽 객체를 생성한다.
    Graphics fg = CreateGraphics();
    // ...
    // 단계 4: 이미지를 화면에 출력한다.
    fg.DrawImage(img, 0, 0);
}
```

그리기 관련 자료형

- 좌표와 관련있는 Point 구조체와 Size 구조체, Rectangle 구조체
- 색상을 나타내는 Color 구조체
- 도형을 그리거나 채우는데 사용되는 Pen 클래스와 Brush 클래스
- 글꼴을 나타내는 Font 클래스
- 이미지를 나타내는 Image 클래스

Point 구조체

- 설명
  - 평면상의 한 점을 표시하기 위한 자료형
  - x 좌표와 y 좌표의 형식으로 위치를 표시
  - 그리기뿐만 아니라 폼이나 컨트롤에서 위치를 지정하는데도 사용
- 생성자

```
Point pt = new Point();       // (0, 0)을 나타냄.
Point pt = new Point(x, y);   // (x, y) 좌표를 나타냄.
```

- 주요 프로퍼티
  - X: X 좌표 또는 수평 위치.
  - Y: Y 좌표 또는 수직 위치.
  - IsEmpty: 빈 구조체 여부.
- 관련 구조체
  - PointF 구조체: 값을 실수로 표현.
- PointF -> Point 변환 메소드

```
public static Point Ceiling(PointF value);      // 올림
public static Point Round(PointF value);        // 반올림
public static Point Truncate(PointF value);     // 내림
```

Size 구조체

- 설명
  - 사각형 모양을 갖는 영역의 크기를 나타내기 위해서 사용되는 구조체
  - 영역의 크기는 폭(width)과 높이(height)로 나타냄.
  - 그리기뿐만 아니라 폼이나 컨트롤에서 사각형 모양으로 표시되는 영역의 크기를 나타낼 때도 사용
- 생성자

```
Size area = new Size();
Size area = new Size(Width, Height);
```

- 주요 프로퍼티
  - Width: 가로 폭.
  - Height: 세로 높이.
  - IsEmpty: 빈 구조체 여부.
- 관련 구조체
  - SizeF 구조체: 값을 실수로 표현.
- SizeF -> Size 변환 메소드

```
public static Size Ceiling(SizeF value);        // 올림
public static Size Round(SizeF value);          // 반올림
public static Size Truncate(SizeF value);       // 내림
```

Rectangle 구조체 [1/2]

- 설명
  - 사각형 모양을 갖는 영역의 위치와 크기를 나타내기 위해서 사용
  - Point 구조체와 Size 구조체의 개념을 모두 가지고 있는 구조체
  - 사각형이나 또는 폼과 컨트롤의 위치와 크기를 동시에 나타내기 위해서 사용
- 생성자

```
Rectangle r = new Rectangle();
Rectangle r = new Rectangle(Point, Size);
Rectangle r = new Rectangle(X, Y, Width, Height);
```

- 주요 프로퍼티
  - X: 사각 영역의 왼쪽 상단의 X 좌표.
  - Y: 사각 영역의 왼쪽 상단의 Y 좌표.
  - Width: 사각 영역의 가로 폭.
  - Height: 사각 영역의 세로 높이.
  - Left/Top: 사각 영역의 왼쪽 X 좌표/위쪽의 Y 좌표
  - Right/Bottom: 사각 영역의 오른쪽 X 좌표/아래쪽의 Y 좌표
  - Location: 사각 영역의 시작 위치 (Point(X, Y))
  - Size: 사각 영역의 크기 (Size(Width, Height))
  - IsEmpty: 빈 구조체 여부.

Rectangle 구조체 [2/2]

- 관련 구조체
  - RectangleF 구조체: 값을 실수로 표현.
- RectangleF -> Rectangle 변환 메소드

```
public static Rectangle Ceiling(RectangleF value);      // 올림
public static Rectangle Round(RectangleF value);        // 반올림
public static Rectangle Truncate(RectangleF value);     // 내림
```

- 주요 메소드
  - Contains(): 점이나 영역의 포함 여부를 알려준다.
  - Inflate(): 지정된 크기만큼 영역을 확장한다.
  - Intersect(): 두 영역의 교차 영역을 구한다.
  - IntersectsWith(): 두 영역의 교차 여부를 알려준다.
  - Offset(): 영역을 지정된 크기만큼 이동한다.
  - Union(): 두 영역을 포함한 최소 영역을 구한다.

Color 구조체 [1/3]

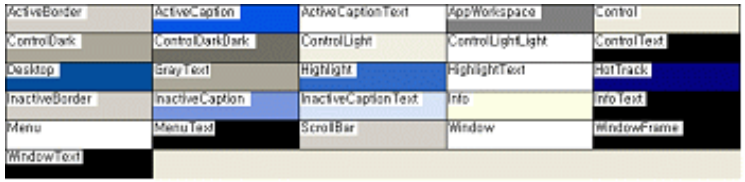
- 설명
  - 색을 RGB(Red, Green, Blue) 형식으로 나타낸 구조체
  - RGB마다 각각 0~255 값을 가짐.
  - 투명도 값인 A(Alpha)을 가질 수도 있음.

- 생성자

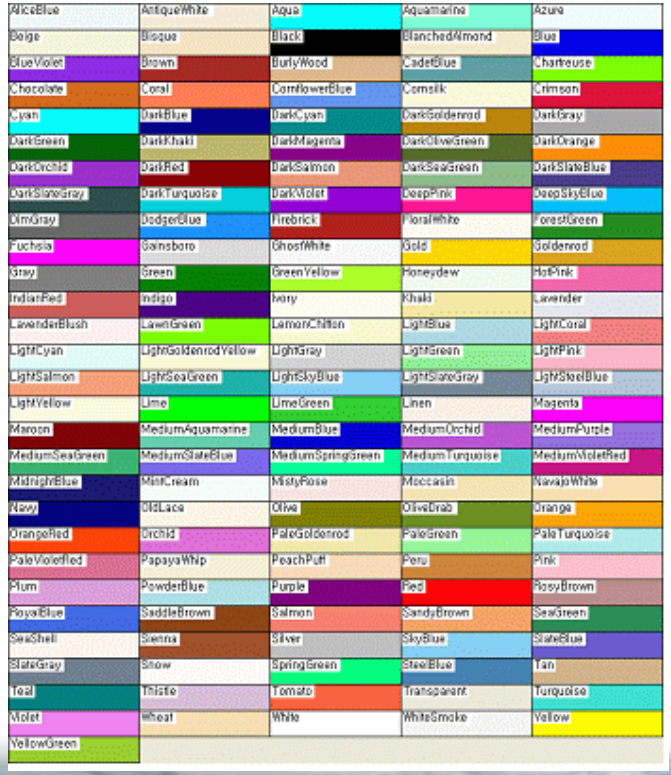
```
Color c = Color.FromArgb(R, G, B);
Color c = Color.FromArgb(A, R, G, B);
Color c = Color.FromKnownColor(KnownColor.Member);
Color c = Color.FromName("ColorName");
```

Color 구조체 [2/3]

- KnownColor 열거형
  - 시스템 색상: 윈도우의 구성 요소의 색을 나타내는 색상.



  - 그외 색상



Color 구조체 [3/3]

- 사용법

```
// ① 프로퍼티를 이용하여 객체 생성.
Color c = Color.White;
// ② 미리 정의된 열거형 상수를 이용하여 객체 생성.
Color c = Color.FromKnownColor(KnownColor.White);
// ③ 미리 정의된 색상 이름을 이용하여 객체 생성.
Color c = Color.FromName("White");
// ④ RGB 값을 이용하여 객체 생성.
Color c = Color.FromArgb(255, 255, 255);
```

- 프로퍼티
  - R: 빨간색(red)의 값 (0~255).
  - G: 녹색(green)의 값 (0~255).
  - B: 파란색(blue)의 값 (0~255).
  - A: 투명도(Alpha)의 값 (0~255).
  - IsEmpty: 빈 구조체 여부.

Pen 클래스

- 설명
  - 직선이나 도형을 그릴 때 색상이나 굵기 등 선에 관한 정보를 가지는 펜을 위한 클래스
- 생성자

```
Pen p = new Pen(Brush b);
Pen p = new Pen(Color c);
Pen p = new Pen(Brush b, float width);
Pen p = new Pen(Color c, float width);
```

- Pens 클래스
  - 미리 정의된 펜들을 가진 프로퍼티로 가진 클래스
  - 프로퍼티 이름은 KnownColor 열거형 상수(시스템 색상 이름을 제외)의 이름과 동일
  - 펜의 두께는 1.0이고 이름과 같은 색상을 가진 펜을 정의.

Pen 클래스 – 주요 프로퍼티

- Width 프로퍼티
  - 펜의 굵기를 나타내는 프로퍼티
    - public float Width { get; set; }
- DashStyle 프로퍼티
  - 점선이나 파선처럼 선의 모양을 나타내는 프로퍼티
    - public DashStyle DashStyle {get; set;}
  - DashStyle 열거형
    - System.Drawing.Drawing2D 네임스페이스에 포함.
- StartCap/EndCap 프로퍼티
  - 선 시작과 끝의 모양을 나타내는 프로퍼티
    - public LineCap StartCap {get; set;}
    - public LineCap EndCap {get; set;}
  - LineCap 열거형
    - System.Drawing.Drawing2D 네임스페이스에 포함.

Brush 클래스

- 설명
  - 도형과 같이 일정한 면적을 차지하는 부분을 색이나 모양 패턴, 혹은 그림으로 채울 때 사용하는 객체가 브러시
  - 브러시를 위한 클래스
- Brush 클래스의 파생 클래스
  - SolidBrush: 단색으로 된 브러시.
  - TextureBrush: 이미지(BMP, JPEG 등)로 된 브러시.
  - HatchBrush: 모양 패턴을 사용하는 브러시.
  - LinearGradientBrush: 선형 색전환(gradient)을 사용하는 브러시.
  - PathGradientBrush: 지정된 경로에 따른 색전환을 사용하는 브러시.
- Brushes 클래스
  - 미리 정의된 브러시들을 가진 프로퍼티로 가진 클래스
  - 프로퍼티 이름은 KnownColor 열거형 상수(시스템 색상 이름을 제외)의 이름과 동일
  - 이름과 같은 색상을 가진 브러시(SolidBrush)을 정의.

SolidBrush 클래스

- 단색으로 된 브러시 객체를 위한 클래스
- 생성자

```
SolidBrush b = new SolidBrush(Color c);
```

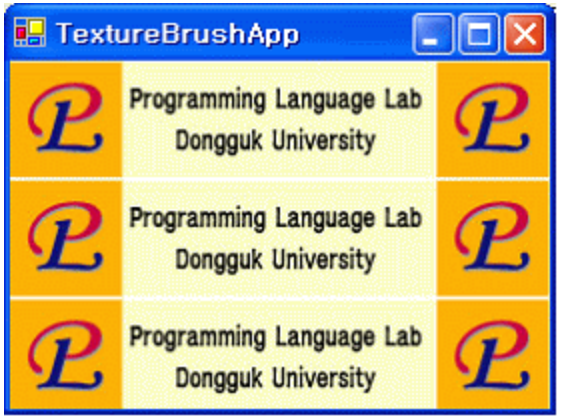
- 예제

```
SolidBrush b = new SolidBrush(Color.Lime);
g.FillRectangle(b, ClientRectangle);
b.Dispose();
```

TextureBrush 클래스

- 이미지로 된 브러시 객체를 위한 클래스
- 생성자

```
TextureBrush b = new TextureBrush(Image img);
```



- 예제

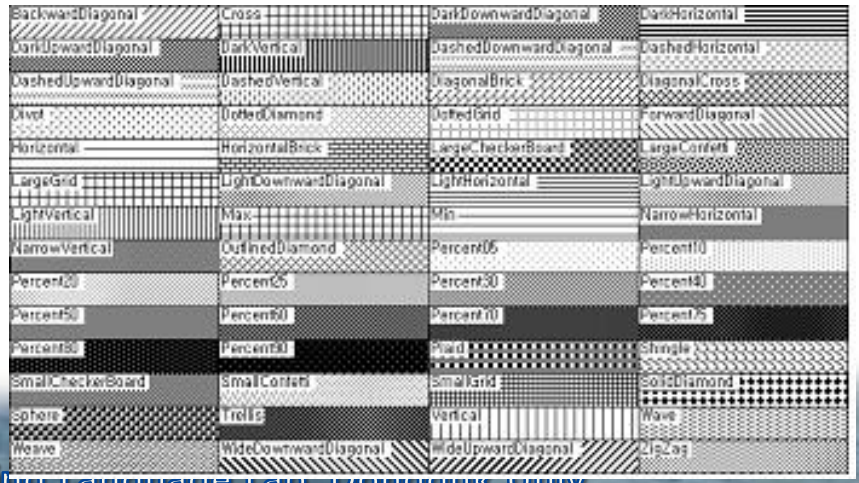
```
Image img = new Bitmap("plac.jpg");      // 이미지 객체 생성
TextureBrush b = new TextureBrush(img); // 이미지로 된 브러시 객체 생성
g.FillRectangle(b, ClientRectangle);    // 사용자 영역을 이미지로 채움
img.Dispose();                          // 이미지 객체를 해제
b.Dispose();                            // 브러시 객체를 해제
```

HatchBrush 클래스

- 모양 패턴으로 된 브러시 객체를 위한 클래스
- System.Drawing.Drawing2D 네임스페이스에 포함
- 생성자

```
HatchBrush b = new HatchBrush(HatchStyle hs, Color c);
```

- HatchStyle 열거형



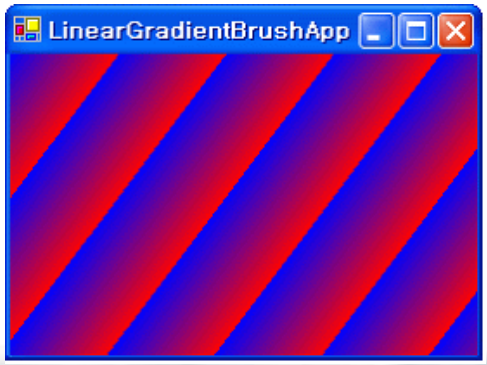
LinearGradientBrush 클래스

- 선형 색 전환으로 면을 칠할 때 사용하는 브러시
- System.Drawing.Drawing2D 네임스페이스에 포함
- 생성자

```
LinearGradientBrush b = new LinearGradientBrush(Point, Point, Color, Color);
```

- 예제

```
LinearGradientBrush b = new LinearGradientBrush(
                                new Point(0, 0), new Point(40, 30),
                                Color.Blue, Color.Red);
g.FillRectangle(b, ClientRectangle);
b.Dispose();
```



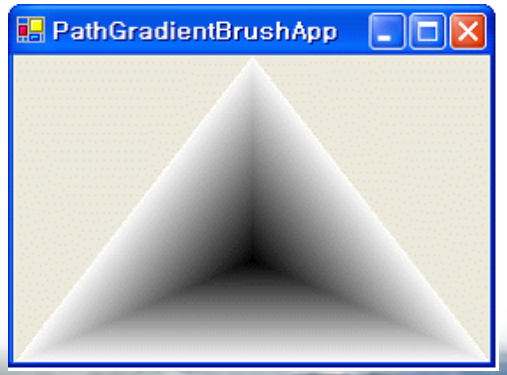
PathGradientBrush 클래스

- 색 전환에서 사용되는 기준점을 여러 개 사용하여 복잡한 형태의 색 전환 브러시를 나타내는 클래스
- System.Drawing.Drawing2D 네임스페이스에 포함

```
PathGradientBrush b = new PathGradientBrush(Point[] pts);
```

- 생성자
- 예제

```
Point[] pts = {
    new Point(ClientRectangle.Width/2, 0),
    new Point(0, ClientRectangle.Height),
    new Point(ClientRectangle.Width, ClientRectangle.Height)
};
PathGradientBrush b = new PathGradientBrush(pts);
g.FillRectangle(b, ClientRectangle);
b.Dispose();
```



Font 클래스

- 문자열을 그릴 때 사용하는 글꼴의 모양과 크기를 나타내는 클래스
- 생성자

```
public Font(string name, float size);
public Font(string name, float size, FontStyle fs);
```

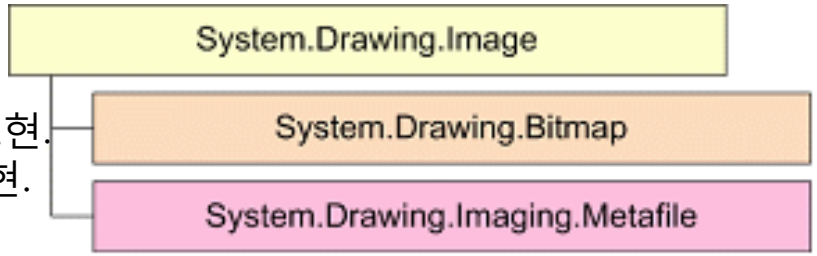
- FontStyle 열거형 상수
  - Bold: 굵은(또는 진한) 텍스트.
  - Italic: 기울임꼴 텍스트.
  - Regular: 일반 텍스트.
  - Strikeout: 중간에 줄이 그어진 텍스트.
  - Underline: 밑줄이 있는 텍스트.

Font 클래스의 주요 프로퍼티

- Name: 글꼴의 이름.
- Size: 글꼴의 크기.
- Height: 글꼴의 높이.
- Bold: 글꼴 모양이 굵은지 여부.
- Italic: 글꼴 모양이 기울임꼴인지 여부.
- Strikeout: 글꼴 모양이 중간에 줄이 있는지 여부.
- Underline: 글꼴 모양이 밑줄이 있는지 여부.

Image 클래스

- 이미지
  - 래스터 기반 이미지
    - 색상을 가진 점(pixel) 단위로 표현
    - 사진과 같이 복잡하고 일정한 형태를 가지지 않은 이미지를 표현하기에 적합
  - 벡터 기반 이미지
    - 점, 선, 도형과 같은 그래픽 단위로 표현
    - 일정한 형태로 구성된 이미지를 표현하기에 적합
    - 이미지의 변환 작업을 쉽게 할 수 있음.
- Image 클래스
  - 이미지를 나타내는 클래스
  - 파생 클래스
    - Bitmap 클래스: 래스터 기반 이미지를 표현.
    - Metafile 클래스: 벡터 기반 이미지를 표현.
- 주요 프로퍼티
  - Size: 이미지 크기.
  - Width: 이미지의 폭.
  - Height: 이미지의 높이.



Bitmap 클래스

- 래스터 기반 이미지를 나타내는 클래스.
  - 메모리에서만 존재하는 가상의 이미지를 나타낼 때도 사용
- 생성자

```
public Bitmap(string fileName);  // 이미지 파일 객체
public Bitmap(int w, int h);     // 가상 이미지 객체
```

- Bitmap 클래스에서 지원하는 파일 형식
  - BMP: Bit-mapped grahics format의 의미.
  - EXIF: EXchangeable Image File의 약자.
  - GIF: Graphics Interchange Format의 약자.
  - JPEG: Joint Photographic Experts Group의 약자.
  - PNG: Portable Network Graphics의 약자.
  - TIFF: Tag Image File Format의 약자.

Metafile 클래스

- 벡터 기반 이미지를 나타내는 클래스
  - 그리기 연산들을 수행하면 생성된 그래픽 객체에 그러한 연산이 기록.
  - 그려진 결과가 저장되는 것이 아니라 그려지는 과정을 기록.
- System.Drawing.Imaging 네임스페이스에 포함

도형 그리기

- DrawXxx() 메소드 계열
  - 선을 그리기 위한 펜을 첫 번째 매개변수로 받아 외곽선을 그리는 메소드.
- FillXxx() 메소드
  - 영역을 칠하기 위한 브러시를 첫 번째 매개변수로 받아 도형의 내부 영역만을 칠하는 메소드.

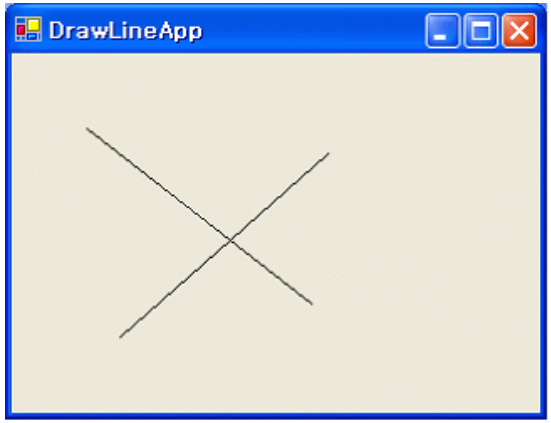
선 그리기 – DrawLine() 메소드

- 형식

```
public void DrawLine(Pen p, Point pt1, Point pt2);
```

- 예제

```
Pen p = new Pen(Color.Black);
Point startPoint = new Point(45, 45);
Point endPoint = new Point(180, 150);
g.DrawLine(p, startPoint, endPoint);
g.DrawLine(p, new Point(190, 60), new Point(65, 170));
p.Dispose();
```



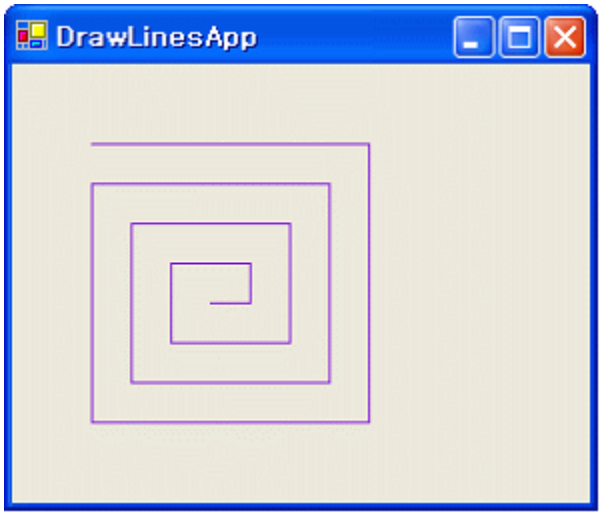
선 그리기 – DrawLines() 메소드

- 형식

```
public void DrawLines(Pen p, Point[] pts);
```

- 예제

```
Point[] pts = {
    new Point(40, 40),    new Point(180, 40),
    new Point(180, 180), new Point(40, 180),
    new Point(40, 60),    new Point(160, 60),
    new Point(160, 160), new Point(60, 160),
    new Point(60, 80), new Point(140, 80),
    new Point(140, 140), new Point(80, 140),
    new Point(80, 100), new Point(120, 100),
    new Point(120, 120), new Point(100, 120)
 };
g.DrawLines(new Pen(Color.BlueViolet), pts);
```



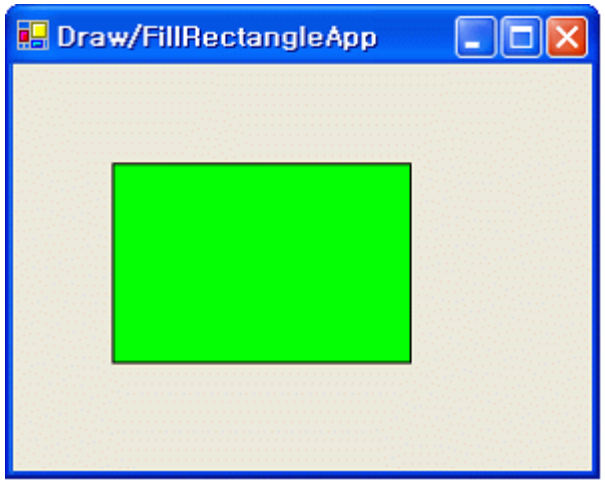
사각형 그리기 – DrawRectangle()

- 형식

```
public void DrawRectangle(Pen p, Rectangle r);
public void FillRectangle(Brush b, Rectangle r);
```

- 예제

```
Rectangle r = new Rectangle(50, 50, 150, 100);
g.FillRectangle(Brushes.Lime, r);
g.DrawRectangle(new Pen(Color.Black), r);
```



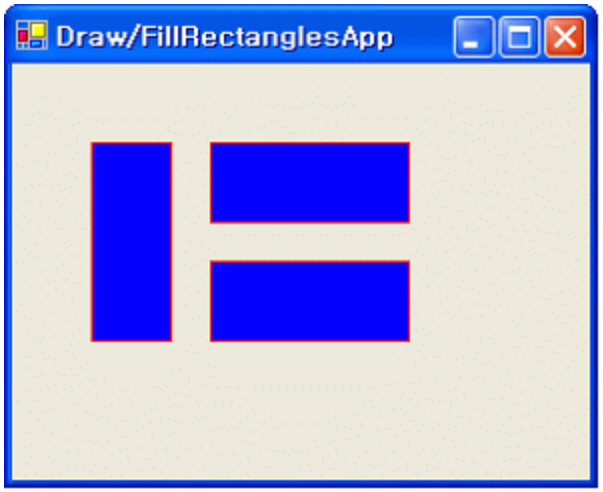
사각형 그리기 – DrawRectangles()

- 형식

```
public void DrawRectangles(Pen p, Rectangle[] rects);
public void FillRectangles(Brush b, Rectangle[] rects);
```

- 예제

```
Rectangle r = new Rec Rectangle[] rects = {
    new Rectangle(40, 40, 40, 100),
    new Rectangle(100, 40, 100, 40),
    new Rectangle(100, 100, 100, 40)
};
g.FillRectangles(Brushes.Blue, rects);
g.DrawRectangles(Pens.Red, rects);
```



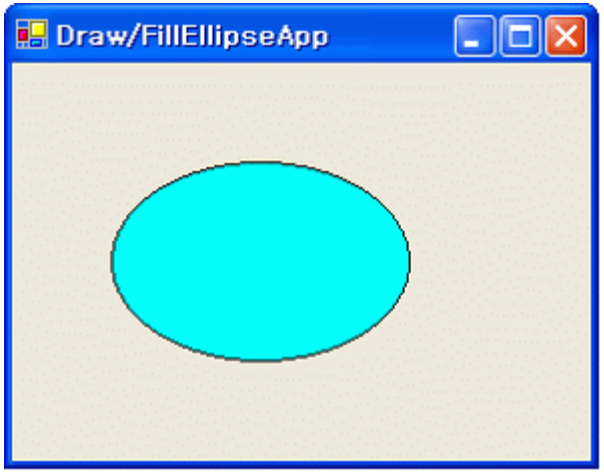
타원 그리기

- 형식

```
public void DrawEllipse(Pen p, Rectangle r);
public void FillEllipse(Pen p, Rectangle r);
```

- 예제

```
Rectangle r = new Rectangle(50, 50, 150, 100);
g.FillEllipse(Brushes.Cyan, r);
g.DrawEllipse(Pens.Black, r);
```



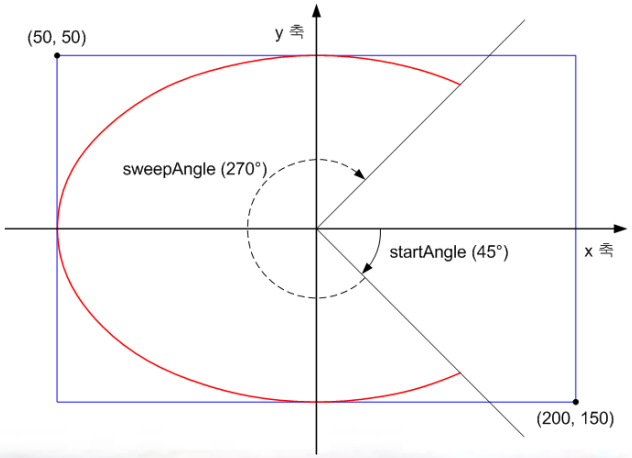
호 그리기

- 형식

```
public void DrawArc(Pen p, Rectangle r, int startAngle, int sweepAngle);
```

- 예제
  - 기준점이 (50, 50)이고 폭과 높이가 각각 150, 100
  - 시작 각도는 X축을 기준으로 45도, 호각이 270도

```
Rectangle r = new Rectangle(50, 50, 150, 100);
g.DrawArc(Pens.Red, r, 45, 270);
```



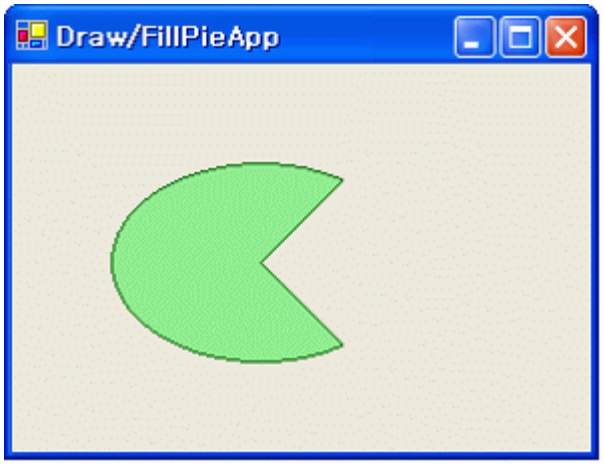
부채꼴 그리기

- 형식

```
public void DrawPie(Pen p, Rectangle r, int startAngle, int sweepAngle);
public void FillPie(Brush b, Rectangle r, int startAngle, int sweepAngle);
```

- 예제

```
Rectangle r = new Rectangle(50, 50, 150, 100);
g.FillPie(Brushes.LightGreen, r, 45, 270);
g.DrawPie(Pens.DarkGreen, r, 45, 270);
```



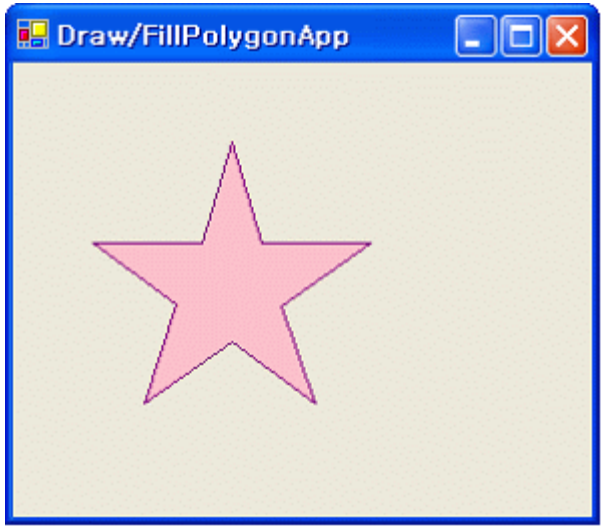
다각형 그리기

- 형식

```
public void DrawPolygon(Pen p, Point[] pts);
public void FillPolygon(Brush b, Point[] pts);
```

- 예제

```
Point[] pts = {
    new Point(110, 40),  new Point(125, 91),
    new Point(180, 91), new Point(135, 123),
    new Point(152, 172), new Point(110, 141),
    new Point(66, 172),  new Point(82, 122),
    new Point(40, 91), new Point(95, 91)
};
g.FillPolygon(Brushes.Pink, pts);
g.DrawPolygon(Pens.Purple, pts);
```



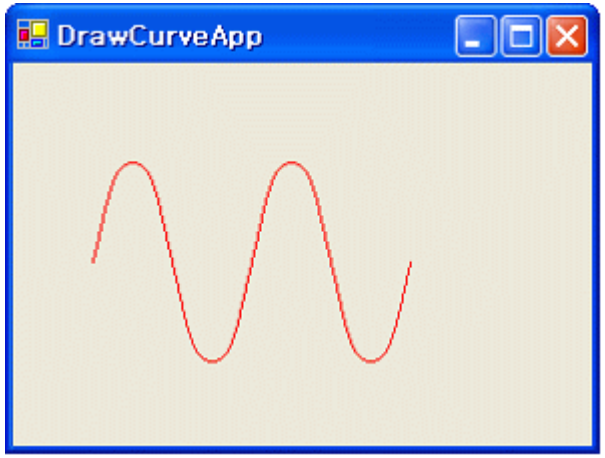
곡선 그리기

- 형식

```
public void DrawArc(Pen p, Rectangle r, int startAngle, int sweepAngle);
```

- 예제

```
Point[] pts = {
    new Point(40, 100),  new Point(50, 60),
    new Point(60, 50), new Point(70, 60),
    new Point(80, 100),  new Point(90, 140),
    new Point(100, 150), new Point(110, 140),
    new Point(120, 100), new Point(130, 60),
    new Point(140, 50),  new Point(150, 60),
    new Point(160, 100), new Point(170, 140),
    new Point(180, 150), new Point(190, 140),
    new Point(200, 100)
};
g.DrawCurve(Pens.Red, pts);
```



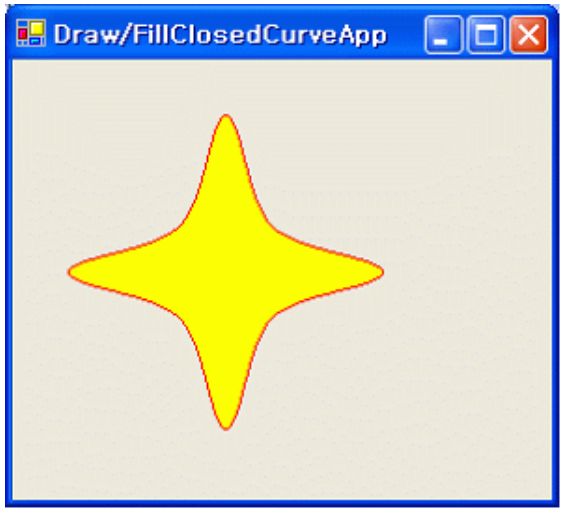
폐곡선 그리기

- 형식

```
public void DrawClosedCurve(Pen p, Point[] pts);
public void FillClosedCurve(Brush b, Point[] pts);
```

- 예제

```
Point[] pts = {
    new Point(115, 30), new Point(140, 90),
    new Point(200, 115), new Point(140, 140),
    new Point(115, 200), new Point(90, 140),
    new Point(30, 115),  new Point(90, 90)
};
g.FillClosedCurve(Brushes.Yellow, pts);
g.DrawClosedCurve(Pens.Red, pts);
```



베지어 곡선 그리기 [1/2]

- 형식

```
public void DrawBezier(Pen p, Point pt1, Point pt2, Point pt3, Point pt4);
public void DrawBeziers(Pen p, Point[] pts);
```

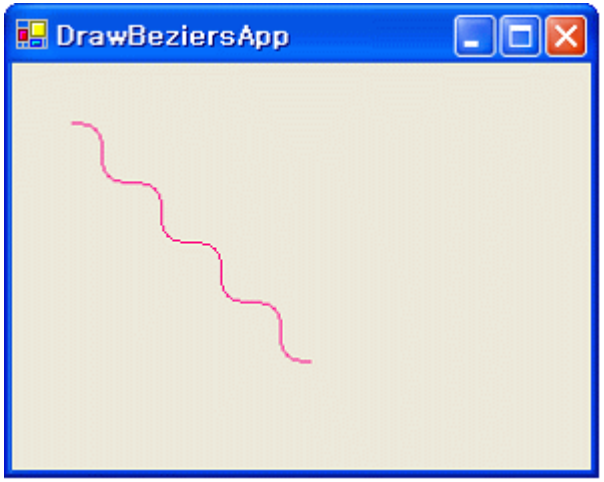
- 예제

```
g.DrawBezier(Pens.Magenta,
      new Point(100, 50),        // start point
      new Point(0, 100),         // control point one
      new Point(200, 100),       // control point two
      new Point(100, 150));      // end point
```

베지어 곡선 그리기 [2/2]

- 예제

```
Point[] pts = {
    new Point(30, 30),                          // 시작점
    new Point(60, 30), new Point(30, 60),       // 제어점
    new Point(60, 60),                          // 끝점 및 시작점
    new Point(90, 60), new Point(60, 90),       // 제어점
    new Point(90, 90),                          // 끝점 및 시작점
    new Point(120, 90), new Point(90, 120),     // 제어점
    new Point(120, 120),                        // 끝점 및 시작점
    new Point(150, 120), new Point(120, 150), // 제어점
    new Point(150, 150),                        // 끝점
};
g.DrawBeziers(Pens.DeepPink, pts);
```



문자열 그리기

- DrawString() 메소드
  - 문자열을 그리기 위해서 사용되는 메소드
- MeasureString() 메소드
  - 어떤 문자열을 그리는데 필요한 영역의 크기를 계산해 주는 메소드

DrawString() 메소드 [1/3]

- 폼에 문자열을 출력하기 위해서 사용하는 메소드.

- 형식

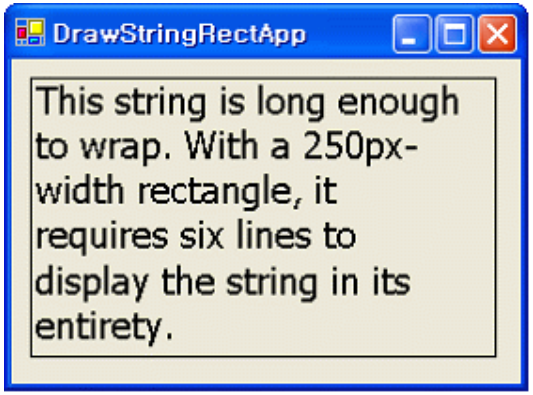
```
public void DrawString(string s, Font f, Brush b, int x, int y);
public void DrawString(string s, Font f, Brush b, Rectangle r);
public void DrawString(string s, Font f, Brush b, Rectangle r, StringFormat sf);
```

DrawString() 메소드 [2/3]

- 예제

```
Font f = new Font("Tahoma", 15);
g.DrawString("Hello World!", f, Brushes.Black, 10, 10);
f.Dispose();
```

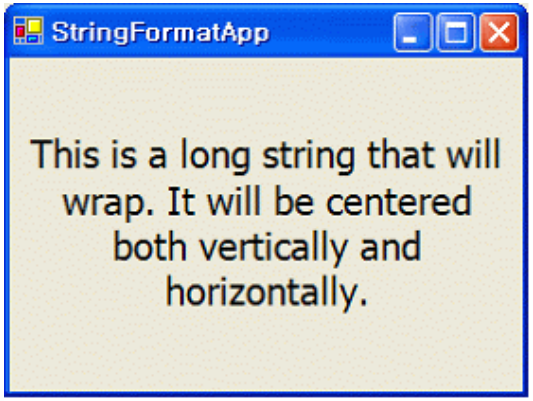

```
string s = "This string is long enough to wrap.";
s += " With a 250px-width rectangle, ";
s += "it requires six lines to display the string in its entirety.";
Font f = new Font("Tahoma", 15);
Rectangle r = new Rectangle(10, 10, 250, 150);
g.DrawRectangle(Pens.Black, r);
g.DrawString(s, f, Brushes.Black, r);
f.Dispose();
```



DrawString() 메소드 [3/3]

- StringFormat 클래스의 주요 프로퍼티
  - Alignment: 수평 정렬.
  - LineAlignment: 수직 정렬.
- StringAlignment 열거형 상수
  - Center: 가운데 정렬.
  - Near: 수평정렬인 경우에 왼쪽 정렬. 수직정렬인 경우에 위쪽 정렬.
  - Far: 수평정렬인 경우에 오른쪽 정렬. 수직정렬인 경우에 아래쪽 정렬.
- 예제

```
string s = "This is a long string that will wrap. ";
s += "It will be centered both vertically and horizontally.";
Font f = new Font("Tahoma", 15);
StringFormat sf = new StringFormat();
sf.Alignment = StringAlignment.Center;        // 수평 정렬
sf.LineAlignment = StringAlignment.Center;    // 수직 정렬
g.DrawString(s, f, Brushes.Black, ClientRectangle, sf);
f.Dispose();
```



MeasureString() 메소드 [1/2]

- 출력하는 문자열을 수용하는데 필요한 영역의 크기를 측정하기 위해서 사용하는 메소드.

- 형식

```
public SizeF MeasureString(string s, Font f);
public SizeF MeasureString(string s, Font f, int width);
```

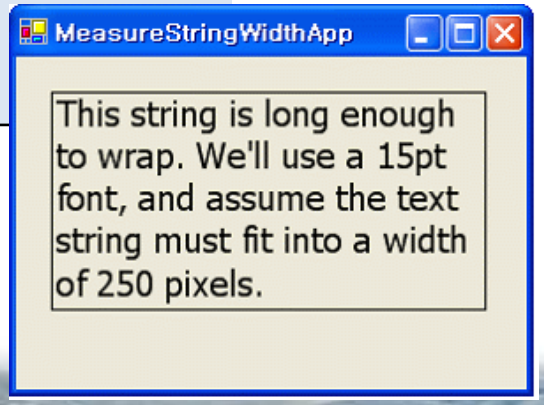
- 예제

```
string s = "Hello World!";
Font f = new Font("Tahoma", 15);
SizeF sf = g.MeasureString(s, f);
g.DrawString(s, f, Brushes.Black, 50, 50);
g.DrawRectangle(Pens.Black, 50, 50, sf.Width, sf.Height);
f.Dispose();
```

MeasureString() 메소드 [2/2]

- 예제

```
string s = "This string is long enough to wrap. ";  // 출력할 문자열
s += "We'll use a 15pt font, and assume ";
s += "the text string must fit into a width of 250 pixels. ";
Font f = new Font("Tahoma", 15);        // 서체는 타호마, 글자 크기는 15.
SizeF sf = g.MeasureString(s, f, 250); // 영역의 크기를 측정한다.
RectangleF rf = new RectangleF(20, 20, sf.Width, sf.Height);
Rectangle r = Rectangle.Ceiling(rf);   // 올림으로 변환
g.DrawString(s, f, Brushes.Black, r);  // 문자열을 출력한다.
g.DrawRectangle(Pens.Black, r);         // 외곽선을 그린다.
f.Dispose();
```



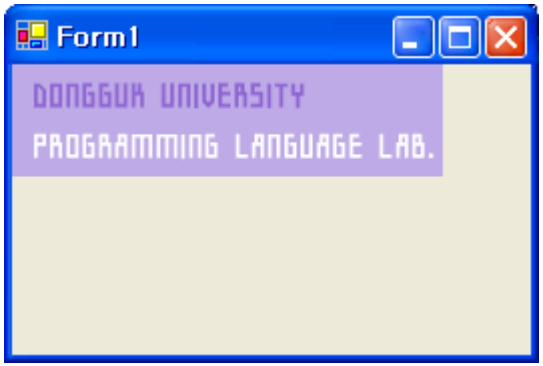
이미지 그리기 [1/4]

- 형식

```
public void DrawImage(Image, int, int);
public void DrawImage(Image, Point);
public void DrawImage(Image, int, int, int, int);
public void DrawImage(Image, Rectangle);
public void DrawImage(Image, Point[]);
public void DrawImage(Image, Rectangle, Rectangle, GraphicsUnit);
```

이미지 그리기 [2/4]

```
Image img = new Bitmap("plac.jpg");
g.DrawImage(img, 0, 0);  // g.DrawImage(img, new Point(0, 0));
```

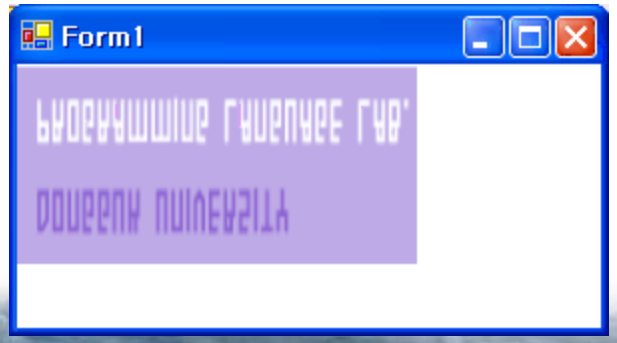



```
Image img = new Bitmap("plac.jpg");
g.DrawImage(img, ClientRectangle);
```

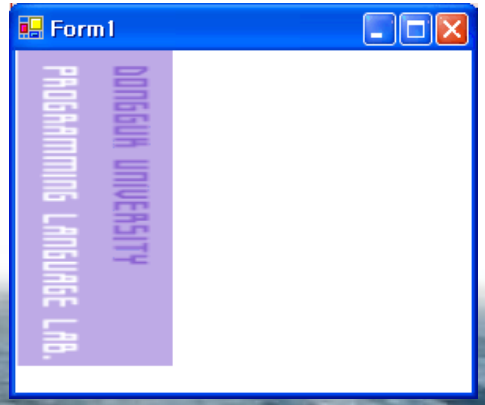
이미지 그리기 [3/4]

```
Image img = new Bitmap("plac.jpg");
Point[] pts = {
    new Point(0, 0),    // 원본의 왼쪽 상단 모서리의 대상 위치
    new Point(200, 0),  // 원본의 오른쪽 상단 모서리의 대상 위치
    new Point(50, 100)  // 원본의 왼쪽 하단 모서리의 대상 위치
};
g.DrawImage(img, pts);
```


```
Point[] pts = {
    new Point(0, 0), new Point(200, 0), new Point(50, 100)
};
```




```
Point[] pts = {
    new Point(100, 0), new Point(100, 200), new Point(0, 0)
};
```



이미지 그리기 [4/4]

```
Image img = new Bitmap("plac.jpg");
Rectangle sr = new Rectangle(0, 0, 80, 30);     // 원본의 부분적인 크기
Rectangle dr = new Rectangle(0, 0, 200, 100);  // 그려질 영역 크기
g.DrawImage(img, dr, sr, GraphicsUnit.Pixel);
```