آشنایی با گرافیک در .NET (GDI+)

آشنایی با GDI+

GDI+ در اصل به کتابخانه کلاسهایی گفته می شود که در ویندوز XP و بعد از آن برای گرافیک عرضه شد. این کتابخانه بصورت Unmanaged نوشته شده است. در .NET برای ترسیمات گرافیکی، یک لایه بالاتر (Wrapper) برای آن ساخته اند (با نام System.Drawing شناخته می شود).

ترسیمات گرافیکی در .NET

در .NET برای همه ترسیمات گرافیکی باید از کلاس Graphics استفاده کنید. این کلاس مثل بوم نقاشی شماست.

مراحل قدم به قدم یک پروژه گرافیکی ساده:

۱ -یک پروژه از نوع Windows Application بسازید.

۲ -گرداننده رویداد Form_Paint را ایجاد نموده و کد زیر را در آن بنویسید:

```
private void frmMain_Paint(object sender, PaintEventArgs e)
{
      Graphics gr = e.Graphics;
      gr.DrawRectangle(Pens.Azure, new Rectangle(5, 5, 100, 100));
}
```

این کد یک چهارضلعی روی فرم رسم می کند. ابتدا شی ای از کلاس Graphics تعریف کرده ایم. هر رویداد Paint متعلق به هر کلاسی که باشد یک پارامتر از نوع PaintEventArgs برمی گرداند که یکی از خواص آن یک کپی از Graphics همان شی ای است که می خواهیم ترسیمات را روی آن انجام دهیم. بنابراین شی Graphics جدید را برابر با شی Graphics پارامتر e قرار میدهیم.

شی Graphics متدهای زیادی برای ترسیمات مختلف دارد که در اینجا برای رسم یک چهارضلعی از متد DrawRectangle استفاده کردیم. معمولا متدهای شی Graphics هر کدام چندین حالت دارند که با توجه به نیاز، حالت مورد نظرتان را انتخاب می کنید.

نکته
در صورتی که می خواهید در آینده برنامه هایتان را روی پلتفرم Windows Mobile منتقل کنید، در صورت امکان حالتهایی از متدها را انتخاب کنید که در هر دو پلتفرم پشتیبانی می شوند.

در اینجا ما از یک قلم با تنظیمات پیش فرض و رنگ Azure استفاده کرده ایم. برای اینکه بتوانیم شکل مورد نظر را با قلم دلخواه رسم کنیم می توانیم مانند مثال زیر عمل کنیم (این کدها را در ادامه کد قبلی بنویسید) :

```
Pen myPen = new Pen(Color.Red, 5);
gr.DrawPie(myPen, new Rectangle(150, 100, 70, 50), 30, 135);
```

در این مثال یک قلم با رنگ قرمز و ضخامت ۵ پیکسل تعریف کرده ایم، سپس با استفاده از این قلم و فراخوانی متد DrawPie بخشی از یک دایره را رسم کردیم. متد DrawPie برای ترسیم تمام و یا بخشی از یک دایره استفاده می شود.

در ادامه این مقاله با متدهای دیگری از شی Graphics آشنا خواهیم شد.

رنگها

رنگها در .NET از ساختمان داده Color استفاده می کنند. در سیستم RGB هر رنگ با ۴ مولفه شناخته می شود که عبارتند از: A و R و G و B.

مولفه	شرح
A	**میزان شفافیت رنگ را مشخصص می کند**
R	**مولفه رنگ قرمز**
G	**مولفه رنگ سبز**
B	**مولفه رنگ آبی**

ساختمان داده Color حاوی مجموعه ای از رنگهایی است که در دنیای هنر آنها را با نام خاص می شناسند. این رنگها بصورت خواص Static ساختمان داده Color قابل استفاده هستند. برای استفاده از این رنگها باید به شکل کد زیر عمل کنید:

```
btnColorTest.BackColor = Color.Aqua;
```

لیست این رنگها و نامهای آنها را در شکل زیر مشاهده می کنید:

Transparent
AliceBlue
AntiqueWhite
Aqua
Aquamarine
Azure
Beige
Bisque
Black
BlanchedAlmond
Blue
BlueViolet
Brown
BurlyWood
CadetBlue
Chartreuse
Chocolate
Coral
CornflowerBlue
Cornsilk
Crimson
Cyan
DarkBlue
DarkCyan
DarkGoldenrod
DarkGray
DarkGreen
DarkKhaki
DarkMagenta
DarkOliveGreen
DarkOrange
DarkOrchid
DarkRed
DarkSalmon
DarkSeaGreen
DarkSlateBlue
DarkSlateGray
DarkTurquoise
DarkViolet
DeepPink
DeepSkyBlue
DimGray
DodgerBlue
Firebrick
FloralWhite
ForestGreen
Fuchsia
Gainsboro
GhostWhite
Gold
Goldenrod
Gray
Green
GreenYellow
Honeydew
HotPink
IndianRed
Indigo
Ivory
Khaki
Lavender
LavenderBlush
LawnGreen
LemonChiffon
LightBlue
LightCoral
LightCyan
LightGoldenrodYellow
LightGray
LightGreen
LightPink
LightSalmon
LightSeaGreen
LightSkyBlue
LightSlateGray
LightSteelBlue
LightYellow
Lime
LimeGreen
Linen
Magenta
Maroon
MediumAquamarine
MediumBlue
MediumOrchid
MediumPurple
MediumSeaGreen
MediumSlateBlue
MediumSpringGreen
MediumTurquoise
MediumVioletRed
MidnightBlue
MintCream
MistyRose
Moccasin
NavajoWhite
Navy
OldLace
Olive
OliveDrab
Orange
OrangeRed
Orchid
PaleGoldenrod
PaleGreen
PaleTurquoise
PaleVioletRed
PapayaWhip
PeachPuff
Peru
Pink
Plum
PowderBlue
Purple
Red
RosyBrown
RoyalBlue
SaddleBrown
Salmon
SandyBrown
SeaGreen
SeaShell
Sienna
Silver
SkyBlue
SlateBlue
SlateGray
Snow
SpringGreen
SteelBlue
Tan
Teal
Thistle
Tomato
Turquoise
Violet
Wheat
White
WhiteSmoke
Yellow
YellowGreen

مجموعه ای دیگر از رنگها به نام رنگهای سیستمی شناخته می شوند. اینها رنگهایی هستند که در سیستم عامل استفاده شده اند مثل رنگ کلیدها، رنگ Border ها و ....

برای استفاده از این رنگها باید از کلاس SystemColors استفاده کنید. این رنگها هم با استفاده از نام آنها قابل دسترسی می باشند. به مثال زیر توجه کنید:

```
btnColorTest.ForeColor = SystemColors.InactiveBorder;
```

شکل زیر رنگهای سیستمی و نام آنها را نمایش داده است:

ActiveBorder	ControlText	Info
ActiveCaption	Desktop	InfoText
ActiveCaptionText	GrayText	Menu
AppWorkspace	Highlight	MenuText
Control	HighlightText	ScrollBar
ControlDark	HotTrack	Window
ControlDarkDark	InactiveBorder	WindowFrame
ControlLight	InactiveCaption	WindowText
ControlLightLight	InactiveCaptionText	

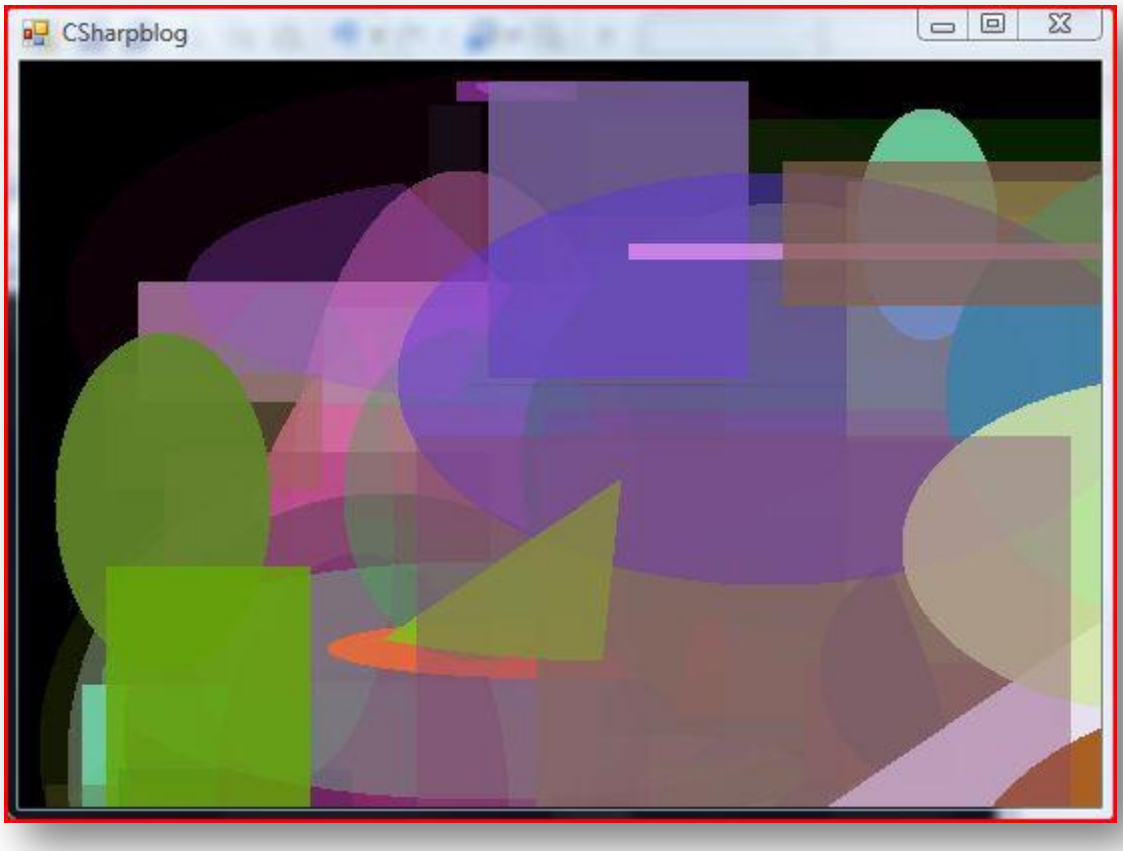
ساختمان داده Color دارای یک متد بسیار ارزشمند به نام FromArgb میی باشد که با استفاده از آن می توانید رنگ دلخواهتان را بسازید. این متد حالتهای مختلفی دارد. پارامترهای مهم آن عبارتند از : میزان Alpha و مولفه های RGB.

مثال: در مثال زیر ۱۰۰ شکل که مشخصات هر کدام بصورت تصادفی انتخاب می شوند رسم خواهد شد:

```
private void frmMain_Paint(object sender, PaintEventArgs e)
{
Random rnd = new Random();
Graphics gr=e.Graphics;
Point[] points=new Point[5];

points[0].X = rnd.Next(this.ClientRectangle.Width);
points[0].Y = rnd.Next(this.ClientRectangle.Height);
points[1].X = rnd.Next(this.ClientRectangle.Width);
points[1].Y = rnd.Next(this.ClientRectangle.Height);
points[2].X = rnd.Next(this.ClientRectangle.Width);
points[2].Y = rnd.Next(this.ClientRectangle.Height);
points[3].X = rnd.Next(this.ClientRectangle.Width);
points[3].Y = rnd.Next(this.ClientRectangle.Height);
points[4].X = rnd.Next(this.ClientRectangle.Width);
points[4].Y = rnd.Next(this.ClientRectangle.Height);

for (int i = 0; i < 100; i++)
```

```
{
        SolidBrush solidBrush=new
        SolidBrush(Color.FromArgb(rnd.Next(255),rnd.Next(255),
                    rnd.Next(255),rnd.Next(255)));

        int shapeCode=rnd.Next(3);
        switch(shapeCode)
        {
              case 0: //Draw Rectangle
                    gr.FillRectangle(solidBrush,
                          rnd.Next(this.ClientRectangle.Width),
                          rnd.Next(this.ClientRectangle.Height),
                          rnd.Next(this.ClientRectangle.Width),
                          rnd.Next(this.ClientRectangle.Height));
                    break;
              case 1:
                    gr.FillEllipse(solidBrush,
                          rnd.Next(this.ClientRectangle.Width),
                          rnd.Next(this.ClientRectangle.Height),
                          rnd.Next(this.ClientRectangle.Width),
                          rnd.Next(this.ClientRectangle.Height));
                    break;
              case 2:
                    gr.FillPie(solidBrush,
                          rnd.Next(this.ClientRectangle.Width),
                          rnd.Next(this.ClientRectangle.Height),
                          rnd.Next(this.ClientRectangle.Width),
                          rnd.Next(this.ClientRectangle.Height),
                          rnd.Next(180),rnd.Next(180));
                    break;
```

```
                case 3:
                        gr.FillPolygon(solidBrush,points);
                        break;
        }

        solidBrush.Dispose();
}
}
```



برس ها و قلم ها– Brushes and Pens

آشنایی با برس ها و قلم ها

از قلم ها (Pens) برای رسم خطوط بیرونی اشکال و از برس ها برای رنگ آمیزی بخش داخلی اشکال در GDI+ استفاده می شود.

با استفاده از دو کلاس Brushes و Pens می توانید از برس ها و قلم های ساخته شده ای که بصورت خواص Static در اختیار شما قرار داده شده است استفاده کنید.

برسها (Brushes)

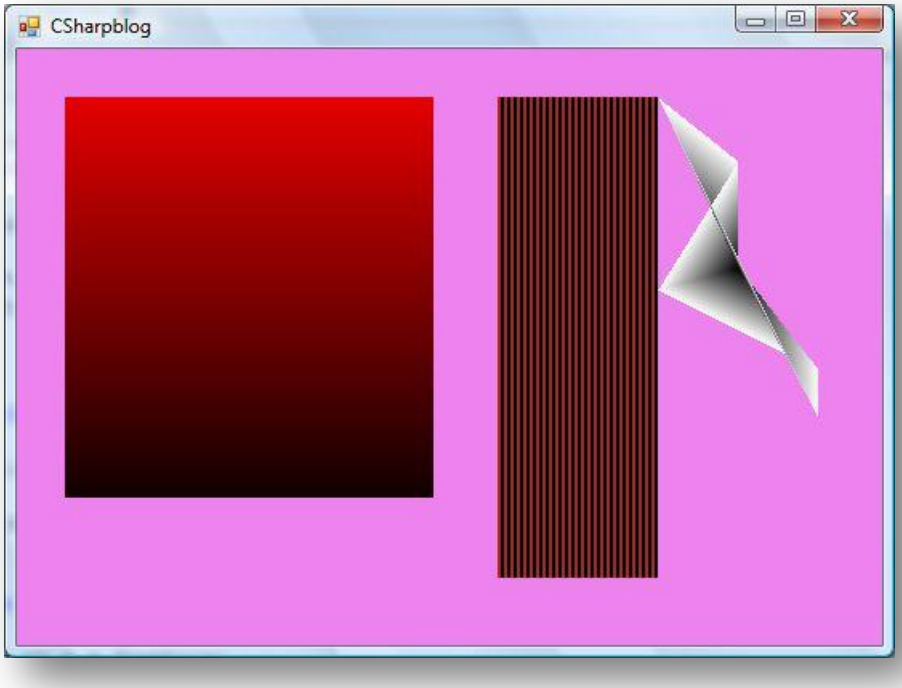
برای تعریف یک برس باید با توجه به نوع برسی که می خواهید استفاده کنید از یکی از کلاسهای مشتق شده از کلاس Brush استفاده کنید. مهمترین این کلاسها عبارتند از:

```
System.Drawing.SolidBrush
System.Drawing.Drawing2D.PathGradientBrush
System.Drawing.Drawing2D.LinearGradientBrush
System.Drawing.Drawing2D.HatchBrush
```

مثالهایی از نحوه استفاده از هر کدام از قلم ها را در کد زیر مشاهده می کنید: (این کدها را باید در گرداننده رویداد Paint قرار دهید):

```
Graphics gr=e.Graphics;
Point[] points = new Point[5];
points[0] = new Point(400, 30);
points[1] = new Point(450, 70);
points[2] = new Point(400, 150);
points[3] = new Point(500, 200);
points[4] = new Point(500, 230);
//Declare brushes
SolidBrush solidBrush=new SolidBrush(Color.Violet);
LinearGradientBrush linearGradientBrush=new LinearGradientBrush(
      new Rectangle(0,0,200,300),
```

```
        Color.Red,
        Color.Black,
        LinearGradientMode.Vertical);
HatchBrush hatchBrush=new HatchBrush(HatchStyle.DarkVertical,Color.Brown);
PathGradientBrush pathGradientBrush=new PathGradientBrush(points);

//Draw with brushes
gr.FillRectangle(solidBrush,
        new Rectangle(0,0,this.ClientSize.Width,this.ClientSize.Height));
gr.FillRectangle(linearGradientBrush,
        new Rectangle(30,30,230,250));
gr.FillRectangle(hatchBrush,
        new Rectangle(300,30,100,300));
gr.FillRectangle(pathGradientBrush,new Rectangle(400,30,100,300));
```



البته به این نکته هم توجه کنید که هر کدام از این برسها Constructor های مختلفی دارند. می توانید از برسهایی که بصورت سیستمی تعریف شده اند استفاده کنید. برای استفاده از این برسها باید از کلاس SystemBrushes استفاده کنید.

قلم ها(Pens)

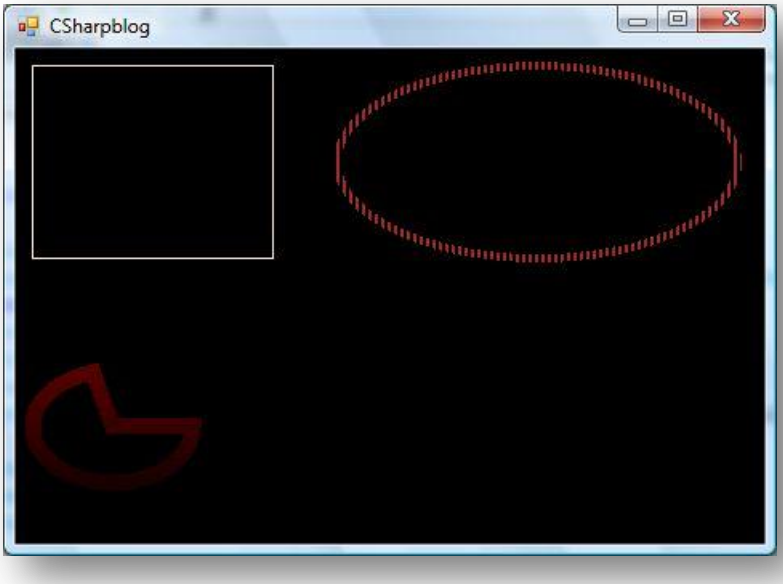
برای تعریف یک قلم از کلاس Pen استفاده می کنیم. این کلاس سازنده های مختلفی دارد که می توانید از آنها استفاده کنید. به مثال زیر توجه کنید:

```
Graphics gr = e.Graphics;
LinearGradientBrush linearGradientBrush = new LinearGradientBrush(
      new Rectangle(0, 0, 200, 300),
      Color.Red,
      Color.Black,
      LinearGradientMode.Vertical);
HatchBrush hatchBrush = new HatchBrush(HatchStyle.DarkVertical,
Color.Brown);

Pen silmplePen = new Pen(Color.BlanchedAlmond);
Pen hatchedPen = new Pen(hatchBrush, 5);
Pen gradientPen = new Pen(linearGradientBrush, 10);
gr.DrawRectangle(silmplePen, new Rectangle(10, 10, 150, 120));
gr.DrawEllipse(hatchedPen, new Rectangle(200, 10, 250, 120));
gr.DrawPie(gradientPen, new Rectangle(10, 200, 100, 70), 0, 250);
```

عددی که در بعضی از حالتهای سازنده Pen استفاده شده است بیانگر ضخامت قلم می باشد. به این نکته توجه داشته باشید که بعد از استفاده از Brush ها و Pen ها حتما باید آنها را از حافظه خارج کنید. این کار را با استفاده از متد Dispose انجام میدهیم:

```
hatchBrush.Dispose();
linearGradientBrush.Dispose();
silmplePen.Dispose();
```

```
hatchedPen.Dispose();
gradientPen.Dispose();
```

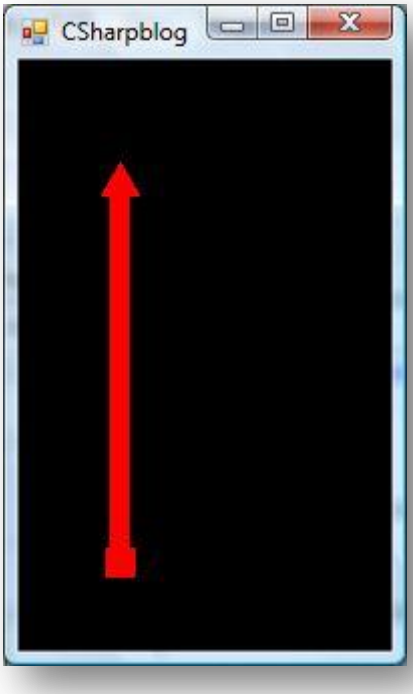


و در آخر این نکته را باید ذکر کرد که در اینجا هم برای استفاده از قلم های سیستمی باید از کلاس SystemPens استفاده کنید.

سرپوش خطوط (Line Caps)

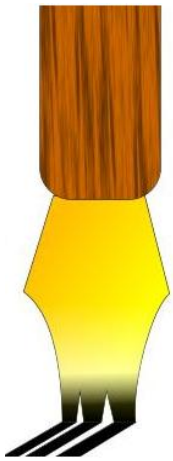
می توانید با استفاده از امکانات GDI+ برای ابتدا و انتهای خطوط سرپوشهایی را تعیین کنید. به مثال زیر توجه کنید:

```
Graphics gr = e.Graphics;
Pen pen=new Pen(Color.Red,10);
pen.StartCap = LineCap.ArrowAnchor;
pen.EndCap = LineCap.SquareAnchor;
gr.DrawLine(pen, 50, 50, 50, 250);
pen.Dispose();
```



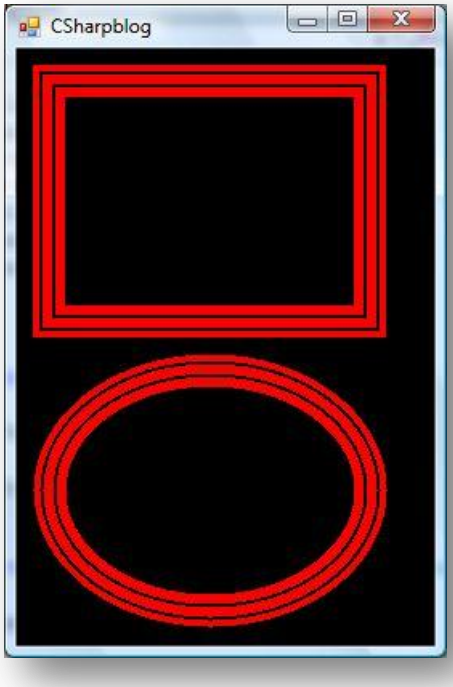
قلم مرکب (Compound Pen)

قلم مرکب نوعی قلم است که توسط شما طراحی شده و به شما اجازه رسم اشکالی با خطوط فانتزی را می دهد. قلمی را تصور کنید که نوک پهنی دارد و شما می توانید به دلخواه بخشهایی از نوک آن را بریده و آن را به شکل دندانه دار درآورید که دندانه ها دارای پهنای متفاوتی می باشند.



مشخصات این قلم را باید ابتدا در آرایه ای از جنس float تعریف کنید و سپس خاصیت CompoundArray قلم مورد نظر را با این آرایه مقداردهی نمایید. کد زیر نمونه ای از ترسیم با استفاده از یک قلم پیچیده می باشد:

```
Pen p = new Pen(Color.Black, 20);
p.CompoundArray = new float[]{0.0f, 0.2f, 0.3f, 0.6f, 0.7f, 1.0f};
e.Graphics.DrawRectangle(p, 20, 20, 200, 150);
e.Graphics.DrawEllipse(p, 20, 200, 200, 150);
p.Dispose();
```



متن و فونت (Text and Fonts)

فونت (Font)

فونت به نوع قلمی گفته می شود که برای نوشتن (در گرافیک ترسیم) متن مورد نظر بکار می رود. برای کار با فونت ها از کلاسهای Font و FontFamily استفاده می کنیم. مثال زیر نحوه استفاده از این کلاسها را نشان می دهد:

```
FontFamily fontFamily=new FontFamily("Tahoma");
Font font=new Font(fontFamily,18, FontStyle.Bold | FontStyle.Italic);
```

برای رسم متون روی یک شی Graphics در GDI+ از متد DrawString استفاده می شود. متد DrawString حالتهای مختلفی دارد که در مثال زیر یکی از این حالتها استفاده شده است:

```
Graphics gr=e.Graphics;
FontFamily fontFamily=new FontFamily("Tahoma");
Font font=new Font(fontFamily,18, FontStyle.Bold | FontStyle.Italic);
gr.DrawString("CsharpBlog",font,Brushes.DarkGoldenrod,50,50);
```

نکته
همه مثالهایی که در این مقاله نوشته شده است باید در گرداننده رویداد Form_Paint قرار گیرند.

اندازه گیری متون

یکی از بحثهای بسیار مهم در ترسیم متون، دانستن اندازه آنهاست. برای اندازه گیری یک متن از متد MeasureString کلاس Graphics استفاده می شود. کد زیر نحوه انجام این کار را نمایش می دهد:

```
string strText="CSharpBlog is C#'s wizard";
Graphics gr=e.Graphics;
FontFamily fontFamily=new FontFamily("Tahoma");
Font font=new Font(fontFamily,18, FontStyle.Bold | FontStyle.Italic);
gr.DrawString(strText,font,Brushes.DarkGoldenrod,50,50);
```

```
SizeF textSize= gr.MeasureString(strText,font);
lblSize.Text = "Width:" + textSize.Width.ToString() +
               "     Height:" + textSize.Height.ToString();
font.Dispose();
```

نکته

پس از استفاده از اشیایی از نوع Font آنها را باید از حافظه خارج نمود. برای این کار متد Dispose آنها را فراخوانی می کنیم.

نوشتن متن بصورت عمودی

برای تعیین فرمت متن مورد نظر از می توانید از کلاس StringFormat استفاده کنید. کد زیر نحوه استفاده از این کلاس را برای رسم متن بصورت عمودی نمایش داده است. در این کد اندازه متن را بدست می آوریم تا با استفاده از آن متن را در یک مستطیل نمایش دهیم:

```
Graphics gr = e.Graphics;
string strText="CSharpBlog is C#'s wizard";
FontFamily fontFamily=new FontFamily("Tahoma");
Font font=new Font(fontFamily,12, FontStyle.Bold | FontStyle.Italic);
StringFormat strFormat=new
StringFormat(StringFormatFlags.DirectionVertical);
SizeF textSize= gr.MeasureString(strText,font);
RectangleF rect = new RectangleF(50, 50, textSize.Height, textSize.Width);
gr.DrawRectangle(Pens.AliceBlue,rect.Left,rect.Top,rect.Width,rect.Height);
gr.DrawString(strText,font,Brushes.Crimson,rect,strFormat);
font.Dispose();
```

نمایش متن با استفاده از Brush

شما می توانید به جای استفاده از قلم ها و رنگهای معمول از برس برای ترسیم متون استفاده کنید. مثال زیر با استفاده از یک برس از نوع LinearGradientBrush متن را ترسیم می کند:

```
Graphics gr = e.Graphics;
```

```
string strText="CSharpBlog is C#'s wizard";
FontFamily fontFamily=new FontFamily("Tahoma");
Font font=new Font(fontFamily,18, FontStyle.Bold | FontStyle.Italic);
LinearGradientBrush gradientBrush=new LinearGradientBrush(
      new Point(0,0),
      new Point(200,200),
      Color.Red,Color.Black);
gr.DrawString(strText,font,gradientBrush,0,50);
font.Dispose();
gradientBrush.Dispose();
```

حالا با استفاده از تکنیک جدید و تکنیک نوشتن متن بصورت عمودی، در این مثال یک متن عمودی را با استفاده از برس LinearGradientBrush رسم می کنیم:

```
Graphics gr = e.Graphics;
string strText="CSharpBlog is C#'s wizard";
FontFamily fontFamily=new FontFamily("Tahoma");
Font font=new Font(fontFamily,18, FontStyle.Bold | FontStyle.Italic);
LinearGradientBrush gradientBrush = new LinearGradientBrush(
      new Point(0, 0),
      new Point(200, 300),
```

```
      Color.Red, Color.Black);

StringFormat strFormat=new
StringFormat(StringFormatFlags.DirectionVertical);

SizeF textSize= gr.MeasureString(strText,font);

RectangleF rect = new RectangleF(50, 50, textSize.Height, textSize.Width);

gr.DrawRectangle(Pens.AliceBlue,rect.Left,rect.Top,rect.Width,rect.Height);

gr.DrawString(strText,font,gradientBrush,rect,strFormat);

font.Dispose();

gradientBrush.Dispose();
```

تنظیم موقعیت متن

در این مثال می خواهیم متن مورد نظر را در یک مستطیل رنگی نمایش دهیم به نحوی که هم از نظر عمودی و هم از نظر افقی در وسط مستطیل قرار گرفته باشد. به مثال زیر توجه کنید:

```
Graphics gr = e.Graphics;

string strText="CSharpBlog is C#'s wizard";
```

```
FontFamily fontFamily=new FontFamily("Tahoma");

Font font=new Font(fontFamily,18, FontStyle.Bold | FontStyle.Italic);

StringFormat strFormat=new StringFormat();
strFormat.Alignment=StringAlignment.Center;

strFormat.LineAlignment=StringAlignment.Center;

SizeF textSize= gr.MeasureString(strText,font);

RectangleF rect = new RectangleF(50, 50, textSize.Width+50,
textSize.Height+50);

gr.FillRectangle(Brushes.White, rect.Left, rect.Top, rect.Width,
rect.Height);

gr.DrawString(strText,font,Brushes.Red,rect,strFormat);

font.Dispose();
```

تصاویر (Images)

برای کار با تصاویر اغلب از کلاسهای Bitmap، Image و Metafile استفاده می شود.

برای کار با تصاویر باید ۴ عملیات انجام دهیم که به ترتیب عبارتند از:

۱ -بارگذاری تصویر از فایل و یا ساخت یک تصویر جدید
۲ -نمایش تصویر بارگذاری شده در شی مورد نظر
۳ -انجام عملیات لازم برای تغییرات مورد نظر در تصویر
۴ -ذخیره تصویر موجود در حافظه در فایل

کلاس Image

کلاس Image یک کلاس انتزاعی (Abstract) می باشد بنابراین مستقیما نمی توانید از آن اشیایی را تعریف نمایید.

کلاس Bitmap

کلاس Bitmap امکاناتی جهت کار با تصاویر از نوع بیت مپی را فراهم نموده است. می توان آن را به عنوان نمونه ای از تصویر درون حافظه تصور نمود. تابع سازنده کلاس Bitmap حالتهای مختلفی دارد که با توجه به نیاز می توانید یکی از این حالتها را انتخاب نمایید. به کد زیر توجه نمایید:

```
string strPath = @"D:\Pictures\Tiger\TigerAndPigs.jpg";
Graphics gr=e.Graphics;
Bitmap bitmap=new Bitmap(strPath);
gr.DrawImage(bitmap,0,0);
bitmap.Dispose();
```

کلاس Metafile

از کلاس Metafile برای کار با تصاویر برداری استفاده می شود. کلاس Metafile با امکانات گسترده ای که دارد بیش از یک کلاس ساده برای نگهداری تصویر به حساب می آید.

انواع فرمتهای تصویری پشتیبانی شده در GDI+

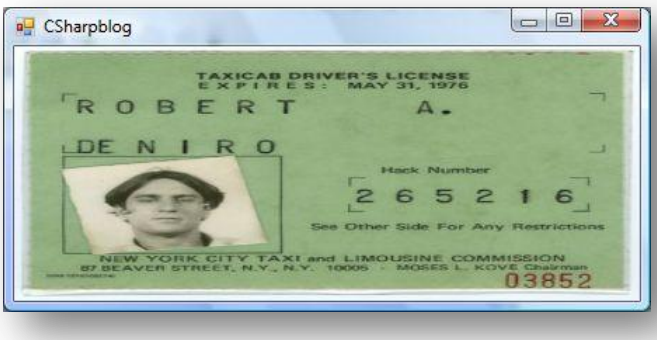
فرتهای تصویری پشتیبانی شده توسط GDI+ در جدول زیر نمایش داده شده است.

نام فرمت	عمق رنگ
BMP	۱، ۴، ۸، ۱۵، ۲۴، ۳۲، ۶۴
EXIF	۲۴
GIF	۱، ۲، ۴، ۸
GIF Animation	۱، ۲، ۴، ۸
JPEG	۲۴
PNG	۱، ۲، ۴، ۸ برای حالتهای سیاه و سفید ۸، ۲۴، ۴۸ برای حالتهای رنگی
TIFF	-

نمایش تصاویر

قبل از هر کاری بر روی یک تصویر باید ابتدا آن را بارگذاری نمود. بارگذاری تصاویر با استفاده از متد سازنده کلاس Bitmap انجام می شود. همانطور که قبلا نیز مشاهده کردید برای نمایش تصویر بارگذاری شده باید از متد DrawImage شی Graphics استفاده نمایید. در مثال زیر علاوه بر بارگذاری و نمایش تصویر، ابعاد تصویر را نیز به اندازه فرم تغییر می دهد. به این کار تغییر مقیاس (Scaling) گفته می شود.

```
string strPath = @" D:\Pictures\Robert De Niro\ Robert De Niro 03";
Graphics gr=e.Graphics;
Bitmap bitmap=new Bitmap(strPath);
gr.DrawImage(bitmap,this.ClientRectangle);
bitmap.Dispose();
```



تغییر در تصاویر

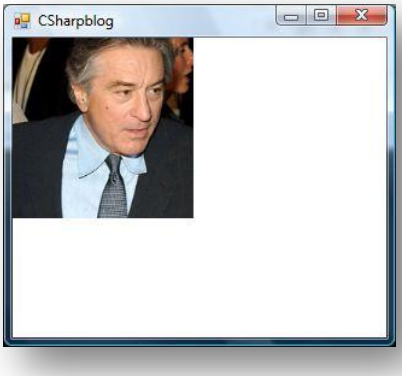
در اغلب موارد هدف از بارگذاری یک تصویر اعمال تغییرات مورد نیاز در تصویر بارگذاری شده می باشد. اغلب این تغییرات دارای الگوریتمهای مختلفی می باشند که بعضی از آنها پیچیده و بعضی دیگر ساده می باشند. در این مقاله تعدادی از این تغییرات که ساده تر و رایج تر می باشند را بررسی خواهیم کرد.

- **برش زدن (Cropping)**

برش زدن به معنای نمایش قسمتی از تصویر اصلی می باشد. کد زیر این کار را نمایش می دهد:

```
string strPath = @" D:\Pictures\Robert De Niro\ Robert De Niro 01";
Graphics gr=e.Graphics;
Bitmap bitmap=new Bitmap(strPath);
gr.FillRectangle(Brushes.White,this.ClientRectangle);
Rectangle sourceRect=new Rectangle(50,50,200,200);
Rectangle destRect=new Rectangle(0,0,150,150);
gr.DrawImage(bitmap,destRect,sourceRect,GraphicsUnit.Pixel);
bitmap.Dispose();
```

در این کد حالتی از متد DrawImage انتخاب شده است که دو ساختمان داده Rectangle را به عنوان پارامترهای ورودی دریافت می کند. با استفاده از این حالت از متد DrawImage می توان بخش مشخصی از یک تصویر (مختصات موجود در sourceRect) را درون محل مورد نظر با ابعاد تعیین شده (destRect) ترسیم نمود.



نکته

در مثال قبل خط زیر اضافه شده است که وظیفه آن رنگ آمیزی زمینه فرم با رنگ سیاه می باشد.

```
gr.FillRectangle(Brushes.White,this.ClientRectangle);
```

اضافه کردن این کد بدین دلیل است که با تغییر اندازه فرم، تصویر مجددا روی آن رسم خواهد شد بدون اینکه تصویر قبلی پاک شود و همین امر باعث می شود که لایه های مختلف رسم شده تصویری نامشخص را به شما نمایش دهند.

- رسم اریب تصویر (Skewing)

کد زیر تصویر را بصورت اریب ترسیم می کند:

```
string strPath = @" D:\Pictures\Robert De Niro\ Robert De Niro 02";
Graphics gr=e.Graphics;
Bitmap bitmap=new Bitmap(strPath);
gr.FillRectangle(Brushes.Black,this.ClientRectangle);

Point[] destPoints={
      new Point(0,0), //Upper Left
      new Point(250,0), //Upper Right
      new Point(50,250)}; //Lower Left
gr.DrawImage(bitmap,destPoints);
bitmap.Dispose();
```

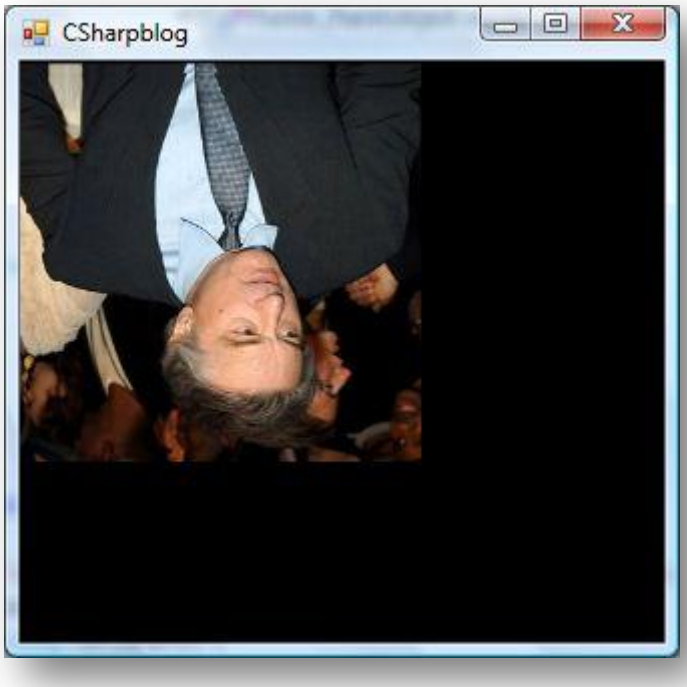
در این کد فقط با مشخص کردن سه نقطه بالا چپ، بالا راست و پایین چپ اندازه و موقعیت و مقدار اریب را مشخص می کنیم. بسیاری از محاسبات در این حالت بر عهده متد DrawImage می باشد.

- بازتاب تصویر (Image Reflection)

کد زیر تصویر را بصورت بازتاب نمایش می دهد:

```
string strPath = @"D:\Pictures\Tiger\TigerAndPigs.jpg";
Graphics gr=e.Graphics;
Bitmap bitmap=new Bitmap(strPath);
gr.FillRectangle(Brushes.Black,this.ClientRectangle);

Point[] destPoints={
      new Point(0,200), //upper left
      new Point(200,200), //upper right
```

```
        new Point(0,0) //lower left
        };
gr.DrawImage(bitmap,destPoints);
bitmap.Dispose();
```



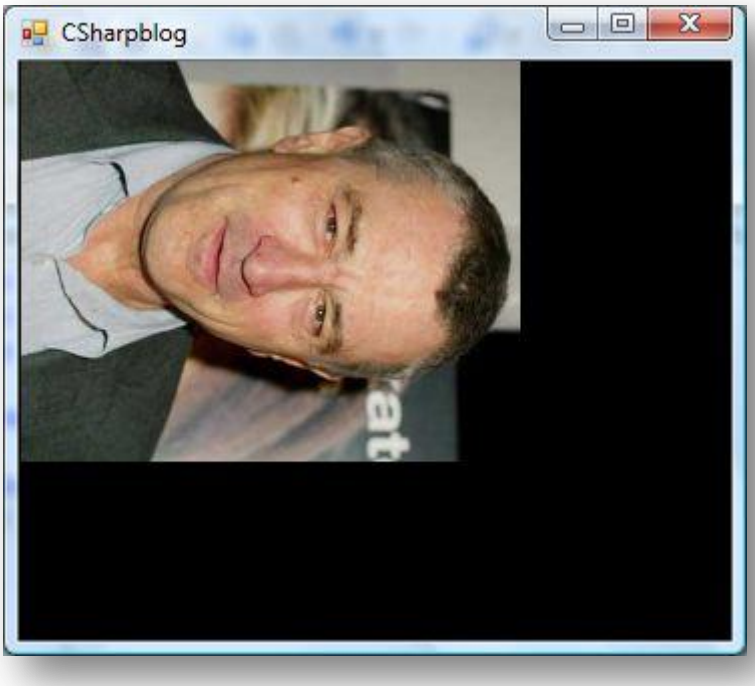
در این مثال برای نمایش بازتاب یک تصویر، مختصات نمایش تصویر جابجا شده است.

- چرخش تصویر (Rotating)

از همان تکنیکی که برای بازتاب استفاده شد، برای چرخش تصویر نیز می توانید استفاده کنید. کد زیر تصویر مورد نظر را ۹۰ درجه چرخش داده است:

```
string strPath = @"D:Pictures\Tiger\TigerAndPigs.jpg";
Graphics gr = e.Graphics;
Bitmap bitmap = new Bitmap(strPath);
gr.FillRectangle(Brushes.Black, this.ClientRectangle);

Point[] destPoints={
```

```
        new Point(200,0), //upper left
        new Point(200,200), //upper right
        new Point(0,0) //lower left
        };
gr.DrawImage(bitmap,destPoints);
bitmap.Dispose();
```



– ساخت Thumbnail

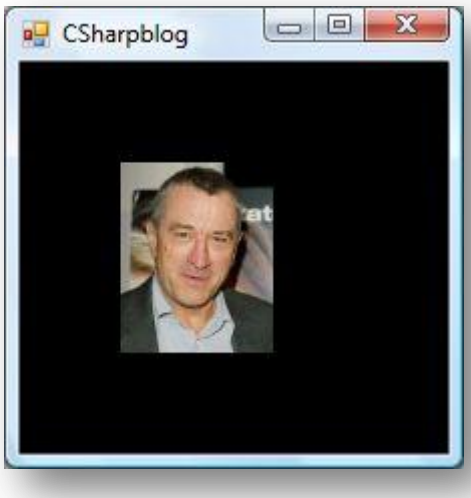
برای ساخت تصاویر انگشتی از یک تصویر باید از متد GetThumbnailImage کلاس Bitmap استفاده نمایید. این متد چهار پارامتر ورودی دارد که دو پارامتر اول آن طول و عرض Thumbnail می باشد. پارامتر سوم و چهارم هیچ وقت استفاده نمی شوند! با توجه به اینکه این پارامترها اجباری می باشند بنابراین باید آنها را مقداردهی نمود.

پارامتر سوم یک Delegate می باشد بنابراین باید یک متد برای آن بنویسید:

```
public bool ThumbnailCallback()
{
      return false;
}
```

سپس کد زیر را در رویداد Paint بنویسید:

```
string strPath = @"D:\Mohammad\Pictures\Tiger\TigerAndPigs.jpg";
Graphics gr = e.Graphics;
Bitmap bitmap = new Bitmap(strPath);
gr.FillRectangle(Brushes.Black, this.ClientRectangle);
Image.GetThumbnailImageAbort myCallback=new
            Image.GetThumbnailImageAbort(ThumbnailCallback);
Image thumbnail=bitmap.GetThumbnailImage(150,150,myCallback,IntPtr.Zero);
gr.DrawImage(thumbnail,50,50);
```



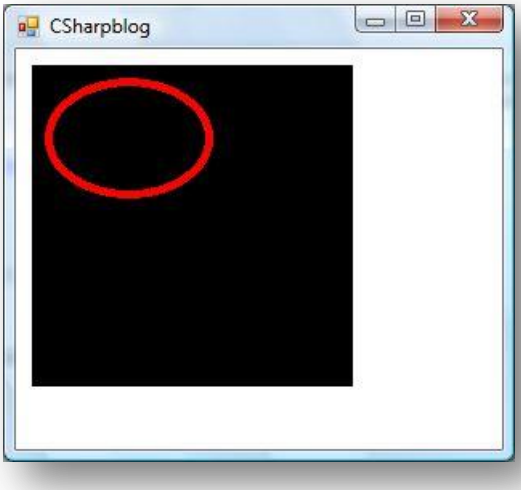
هرجا که از این کد استفاده کنید نیازی به تغییر پارامترهای سوم و چهارم نمی باشد. بنابراین نیازی به توضیحات اضافی درباره آنها نمی باشد.

ترسیم درون تصویر

برای ترسیم درون یک تصویر ابتدا باید تصویر را درون یک Bitmap بارگذاری نمایید. در ابتدای این مقاله در مورد متدهای ترسیم و اینکه این متدها در کلاس Graphics قابل دسترسی می باشند توضیح داده شده است. بنابراین ابتدا باید شی Graphics شی Bitmap مورد نظر را بدست آوریم. بدین منظور از متد FromImage کلاس Graphics استفاده می کنیم.

کد زیر مراحل انجام این کار را نمایش داده است:

```
private Bitmap bitmap=new Bitmap(320,200);


private void Form1_Load(object sender, EventArgs e)
{
      Graphics gr=Graphics.FromImage(bitmap);
      Pen pen=new Pen(Color.Red,5);
      gr.FillRectangle(Brushes.Black,0,0,bitmap.Height,bitmap.Width);
      gr.DrawEllipse(pen,new Rectangle(10,10,100,70));
      pen.Dispose();
}


private void Form1_Paint(object sender, PaintEventArgs e)
{
      Graphics gr=e.Graphics;
      gr.FillRectangle(Brushes.White,this.ClientRectangle);
      gr.DrawImage(bitmap,10,10,bitmap.Width,bitmap.Height);
}
```



در این روش ابتدا هر چه را که مورد نیاز است در Graphics شی Bitmap مورد نظر رسم می کنیم و سپس این شی Bitmap را به عنوان یک تصویر در Graphics فرم (یا هر شی دیگری که مد نظر است) ترسیم می کنیم.

نمایش انیمیشن

قطعا با فایلهای Gif Animation آشنا هستید. در ادامه مقاله نحوه نمایش این نوع فایلها را بررسی خواهیم کرد.

برای نمایش فایلهای انیمیشن باید از کلاس ImageAnimator استفاده کنید. استفاده از این کلاس روش خاصی دارد. روش آن بدین صورت است که ابتدا باید فایل حاوی انیمیشن را درون یک شی Bitmap بارگذاری کنید. با وجود اینکه این شی بارگذاری شده است ولی شی Bitmap به تنهایی قادر به نمایش فریم های موجود در فایل انیمیشن نمی باشد به عبارت دیگر باید هر فریم را قبل از نمایش رندر کرده و سپس محتوای جدید را ترسیم نمود. نحوه انجام این کار در کد زیر نمایش داده شده است:

```
private Bitmap bitmap;


private void OnFrameChanged(object sender, EventArgs e)
{
      this.Invalidate();
```

```
}

private void Form1_Load(object sender, EventArgs e)
{
        string strPath = @"D:\Pictures\device\iranflag.gif";
        bitmap=new Bitmap(strPath);
        ImageAnimator.Animate(bitmap,new EventHandler(this.OnFrameChanged));
}

private void Form1_Paint(object sender, PaintEventArgs e)
{
        Graphics gr=e.Graphics;
        ImageAnimator.UpdateFrames();
        gr.DrawImage(bitmap, 0,0,bitmap.Width,bitmap.Height);
}
```