GDI+

- Successor to Windows Graphics Device Interface (GDI)
  - supports GDI for compatibility with existing applications
  - optimizes many of the capabilities of GDI
  - provides additional features
- Class-based application programming interface (API)
  - two independent implementations: .NET (managed code) and unmanaged code
- Applicability
  - included in Windows XP and the Windows Server 2003
  - redistributable for NT 4.0 SP6, 2000, 98, Me
- Gdiplus.dll

New Features of GDI+

- Gradient brushes (linear and path) for filling shapes, paths and regions
- Cardinal splines
- Independent path objects (objects of GraphicsPath)
- Transformations, Matrix object
- Transformations of regions
- Alpha blending
- Image formats:
  - BMP, GIF, JPEG, Exif, PNG, TIFF, ICON, WMF, EMF

Graphics Object

- The core of GDI+
- It is associated with a particular window on the screen
- It contains attributes that specify how items are to be drawn
- Improvements (comparing with DC):
  - pens, brushes, paths, images, and fonts as parameters
  - no current position for drawing lines
  - separate methods for draw and fill

```
private void AboutForm_Paint(object sender,
                             PaintEventArgs e)
{
    e.Graphics.FillRectangle(Brushes.White,
        0, 0, 200, 100);
    Pen mP = new Pen(Color.Red, 3);
    e.Graphics.DrawLine(mP, 20, 10, 90, 90);
}
```

Using `Graphics` objects

- Obtaining `Graphics` object:
  - `Paint` event handler: `PaintEventArgs.Graphics`
  - `Control.CreateGraphics()`
  - `Graphics: FromHdc(), FromHwnd(), FromImage()`
- Triggering the `Paint` Event:
  - `Form.Invalidate()`
  - `Form.Update(), Form.Refresh()`
- Flicker-free drawing:

```
// in form's constructor or OnLoad
DoubleBuffered = true;
```

Simple Figures

- Methods of the `Graphics` class to draw:
  - Simple figures:
    - `DrawLine()`, `DrawRectangle()`, `DrawEllipse()`, `DrawArc()`, `DrawPolygon()`
    - `FillEllipse()`, `FillPie()`, `FillPolygon()`, `FillRectangle()`
  - Cardinal splines:
    - `DrawCurve()`, `DrawClosedCurve()`
    - `FillClosedCurve()`
  - Bezier splines:
    - `DrawBezier()`, `DrawBeziers()`

Paths

- Formed by combining lines, rectangles, and simple curves
- The `GraphicsPath` class
  - methods for adding simple figures: `AddLine()`, `AddRectangle()`, `AddEllipse()`, `AddArc()`, `AddPie()`, `AddBezier()`, `AddCurve()`, `AddClosedCurve()`, `AddPolygon()`, `AddString()`
  - joining two paths: `AddPath()`
- `Graphics.DrawPath()`
- `Graphics.FillPath()`

Pens

- The **`Pen`** class
- Attributes
  - width – **`SetWidth()`**
  - alignment – **`SetAlignment()`**, **`PenAlignment`** enumeration
  - line caps – **`SetStartCap()`**, **`SetEndCap()`**, **`LineCap`** enumeration
  - joining lines – **`SetLineJoin()`**, **`LineJoin`** enumeration
  - dashed line – **`SetDashPattern()`**
  - texture

```
Image image = new Bitmap("texture.jpg");
TextureBrush tBrush = new TextureBrush(image);
Pen texturedPen = new Pen(tBrush, 30);
e.Graphics.DrawLine(texturedPen, 0, 0, 50, 50);
```

Brushes

- The **Brush** class is abstract
  - **SolidBrush, HatchBrush, TextureBrush, LinearGradientBrush, PathGradientBrush**

```
SolidBrush solidBrush = new SolidBrush(
    Color.FromArgb(255, 255, 0, 0));
e.Graphics.FillEllipse(solidBrush,0,0,100,60);

HatchBrush hBrush = new HatchBrush(
    HatchStyle.Horizontal,
    Color.Green, Color.Blue);
e.Graphics.FillEllipse(hBrush,100,0,100,60);

Image image = new Bitmap("texture.jpg");
TextureBrush tBrush = new TextureBrush(image);
tBrush.Transform = new Matrix(75.0f/640.0f,
    0.0f, 0.0f, 75.0f/480.0f, 0.0f, 0.0f);
e.Graphics.FillEllipse(tBrush,200,0,100,60);
```

Images

- The **`Image`** class
- The **`Bitmap`** class is inherited from **`Image`**
  - it contains specialized methods for loading, displaying, and manipulating raster images
- **`PixelFormat.Format32bppPArgb (CachedBitmap`** for unmanaged code)
  - the format compatible with current screen's settings
  - using this format can increase the efficiency of drawing bitmaps

`Graphics.DrawImage()`

- Cropping
- Scaling, `InterpolationMode` enumeration
- Rotating, reflecting, skewing
- Automatic scaling – when size is not specified
  - GDI+ scales the image so that its physical size on the current display device is as close as possible to its physical size on the device where it was created

Image Encoders and Decoders

- Listing installed encoders and decoders:
  - `ImageCodecInfo: GetImageEncoders(), GetImageDecoders()`
- Converting images

```
Image image = new Bitmap("bird.jpg");
try {
    image.Save("bird.png", ImageFormat.Png);
}
catch (Exception exc) {
    MessageBox.Show("conversion failed: exc=" +
        exc.ToString());
}
```

Alpha Blending

- displayColor = sourceColor * alpha / 255 +
  backgroundColor * (255 – alpha) / 255
- `Color` – 4 values: alpha, red, green, blue

```
SolidBrush opaque(Color.FromArgb(255,0,0,255));
SolidBrush semi(Color.FromArgb(128,0,0,255));
```

- Compositing modes

- Alpha values in images

Texts

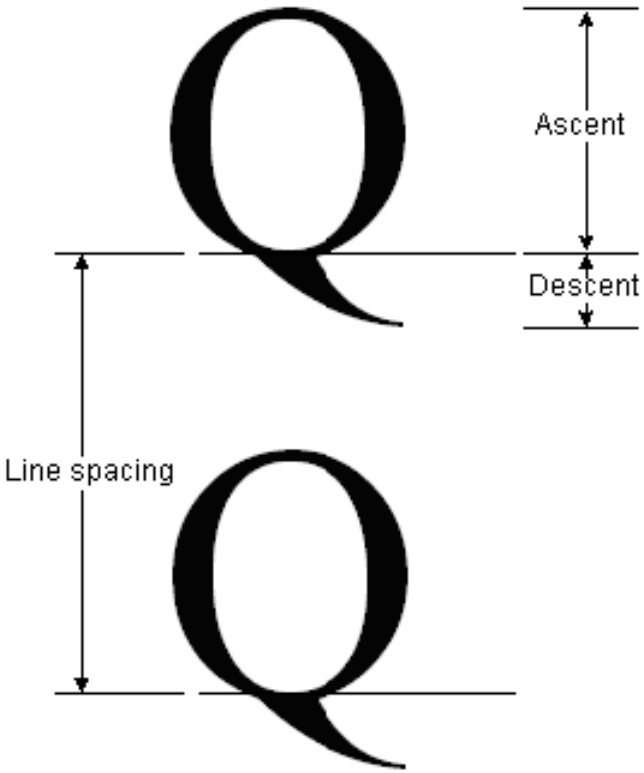
- `Graphics.DrawString()`
  - at specified location
  - in a rectangle
- Formatting
  - aligning – `StringFormat.SetAlignment()`, `StringFormat.SetLineAlignment()`
  - tab stops – `StringFormat.SetTabStops()`
  - vertical text – `StringFormat.SetFormatFlags()`
- Antialiasing
  - `Graphics.SetTextRenderingHint()`

Fonts

- Creating

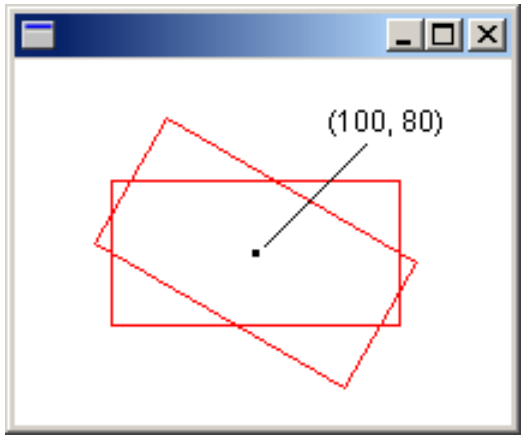
```
Font font = new Font("Arial", 16,
                     FontStyle.Regular);
```

- Metrics
  - `Font.GetSize()`
  - `FontFamily.GetEmHeight()`
  - `FontFamily.GetCellAscent()`
  - `FontFamily.GetCellDescent()`
  - `FontFamily.GetLineSpacing()`



Graphics Containers

- The state of a **`Graphics`** objects
  - link to device context
  - quality settings
  - transformations
  - clipping region
- Using graphics containers




```
Pen pen = new Pen(Color.Red);
GraphicsContainer graphicsContainer;
gr.TranslateTransform(100.0f, 80.0f);

graphicsContainer = gr.BeginContainer();
  gr.RotateTransform(30.0f);
  gr.DrawRectangle(pen,-60,-30,120,60);
gr.EndContainer(graphicsContainer);

gr.DrawRectangle(pen,-60,-30,120,60);
```

Transformations

- `Matrix` class
- Simple methods of `Graphics` class
  - `ScaleTransform()`
  - `RotateTransform()`
  - `TranslateTransform()`
- Transformation order is significant

Regions

- `Region` class
- Elements:
  - lines
  - polygons
  - curves
- Hit testing
  - `Region.IsVisible(point, graphics);`
- Clipping
  - `Graphics.SetClip(region);`

Recoloring

- `ColorMatrix` structure
- `ImageAttributes` class

```
Image im = new Bitmap(@"d:\_\img\kkk.jpg");
ImageAttributes imageAttributes = new
    ImageAttributes();
float[][] colorMatrixElements =  {
  new float[] {1.0f, 0.0f, 0.0f, 0.0f, 0.0f},
  new float[] {0.0f, 1.0f, 0.0f, 0.0f, 0.0f},
  new float[] {0.5f, 0.0f, 1.0f, 0.0f, 0.0f},
  new float[] {0.0f, 0.0f, 0.0f, 1.0f, 0.0f},
  new float[] {0.0f, 0.0f, 0.0f, 0.0f, 1.0f}};
ColorMatrix colorMatrix = new
ColorMatrix(colorMatrixElements);
imageAttributes.SetColorMatrix(colorMatrix,
    ColorMatrixFlag.Default,
    ColorAdjustType.Bitmap);
e.Graphics.DrawImage(im,
    new Rectangle(15, 10, im.Width, im.Height),
    0, 0, im.Width, im.Height,
    GraphicsUnit.Pixel, imageAttributes);
```

Using GDI+ from unmanaged code

- Include the **gdiplus.h** header

```
// !!! DO NOT DEFINE WIN32_LEAN_AND_MEAN
//#define WIN32_LEAN_AND_MEAN
#include <windows.h>
#include <gdiplus.h>
```

- Use the **Gdiplus** namespace

```
using namespace Gdiplus;
```

- Add the **gdiplus.lib** library to the project
  - in the project's settings or use the **#pragma** directive

```
// or set it in project's options
#pragma comment(lib, "gdiplus.lib")
```

Using GDI+ from unmanaged code cont.

- Initialize GDI+

```
GdiplusStartupInput gdiplusStartupInput;
ULONG_PTR gdiplusToken;
GdiplusStartup(&gdiplusToken,
               gdiplusStartupInput,
               NULL);
```

- Use GDI+ objects and functions

```
PAINTSTRUCT ps;
HDC hdc = BeginPaint(hWnd, &ps);
Graphics myGraphics(hdc);
// ...
EndPaint(hWnd, &ps);
```

- Clean up resources used by GDI+

```
GdiplusShutdown(gdiplusToken);
```