GDI+

- Następca GDI *(Graphics Device Interface)*
  - całkowicie obsługuje GDI dla kompatybilności z istniejącymi aplikacjami
  - optymalizacja wielu cech GDI
  - rozszerzenie o nowe możliwości
- Biblioteka obiektowa
  - dwie niezależne implementacje: .NET i kod niezarządzany
- Obsługiwane wersje Windows:
  - w instalacji XP i wszystkich następnych
  - jako dodatek możliwy do zainstalowania na NT 4.0 SP6, 2000, 98, Me
- Gdiplus.dll

Nowe cechy GDI+

- Gradientowe pędzle (liniowe i zadawane ścieżkami)
- Krzywe typu Cardinal *(cardinal splines)*
- Niezależne obiekty ścieżek (GraphicsPath)
- Przekształcenia, obiekty Matrix
- Przekształcenia regionów
- Przezroczystość *(alpha blending)*
- Formaty plików graficznych:
  - BMP, GIF, JPEG, Exif, PNG, TIFF, ICON, WMF, EMF

Obiekt klasy `Graphics`

- Podstawowa, najważniejsza klasa GDI+
- Zwykle związany z oknem
- Zawiera ustawienia decydujące o rysowaniu elementów
- Udoskonalenia w stosunku do GDI:
  - pióra, pędzle, ścieżki, obrazki i fonty jako parametry metod
  - nie ma aktualnej pozycji wykorzystywanej do rysowania linii
  - oddzielne metody do rysowania konturu i wypełnionej figury

```
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);
}
```

Użycie obiektu klasy `Graphics`

- Pobieranie **`Graphics`**:
  - obsługa zdarzenia **`Paint: PaintEventArgs.Graphics`**
  - **`Control.CreateGraphics()`**
  - **`Graphics: FromHdc(), FromHwnd(), FromImage()`**
- Wymuszanie odświeżenia:
  - **`Form.Invalidate()`**
  - **`Form.Update(), Form.Refresh()`**
- Zapobieganie mruganiu:

```
// zwykle w konstruktorze lub OnLoad
DoubleBuffered = true;
```

Proste figury

- Metody klasy **`Graphics`**:
  - proste figury:
    - **`DrawLine()`, `DrawRectangle()`, `DrawEllipse()`, `DrawArc()`, `DrawPolygon()`**
    - **`FillEllipse()`, `FillPie()`, `FillPolygon()`, `FillRectangle()`**
  - krzywe typu Cardinal:
    - **`DrawCurve()`, `DrawClosedCurve()`**
    - **`FillClosedCurve()`**
  - krzywe Beziera:
    - **`DrawBezier()`, `DrawBeziers()`**

Ścieżki

- Tworzone z linii, prostych figur i krzywych
- Klasa **`GraphicsPath`**
  - dodawanie prostych figur do ścieżki: **`AddLine()`**, **`AddRectangle()`**, **`AddEllipse()`**, **`AddArc()`**, **`AddPie()`**, **`AddBezier()`**, **`AddCurve()`**, **`AddClosedCurve()`**, **`AddPolygon()`**, **`AddString()`**
  - łączenie dwóch ścieżek: **`AddPath()`**
- **`Graphics.DrawPath()`**
- **`Graphics.FillPath()`**

Pióra

- Klasa **`Pen`**
- Właściwości:
  - szerokość – **`SetWidth()`**
  - wyrównanie – **`SetAlignment()`**, wyliczenie **`PenAlignment`**
  - końce linii – **`SetStartCap()`**, **`SetEndCap()`**, wyliczenie **`LineCap`**
  - łączenie linii – **`SetLineJoin()`**, wyliczenie **`LineJoin`**
  - kreskowanie linii – **`SetDashPattern()`**
  - tekstura

```
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);
```

Pędzle

- Abstrakcyjna klasa **Brush**
  - **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);
```

Obrazki

- Klasa **`Image`** - abstrakcyjna
- Klasa **`Bitmap`** dziedzicząca z **`Image`**
  - wyspecjalizowane metody do odczytu, wyświetlania i modyfikacji obrazków rastrowych
- **`PixelFormat.Format32bppPArgb`**
  (klasa **`CachedBitmap`** w wersji niezarządzanej)
  - format zgodny z aktualnymi ustawieniami ekranu
  - trzymanie obrazków w tym formacie może znacznie przyspieszyć ich rysowanie

`Graphics.DrawImage()`

- Obcinanie
- Skalowanie, wyliczenie **`InterpolationMode`**
- Obracanie, odwracanie, przekrzywianie
- Automatyczne skalowanie, gdy nie został podany rozmiar
  - GDI+ skaluje obrazki w taki sposób, by ich fizyczny rozmiar na urządzeniu używanym do rysowania był jak najbliższy fizycznemu rozmiarowi na urządzeniu, na którym zostały stworzone

Image Encoders i *Decoders*

- Wyliczenie zainstalowanych *encoders* i *decoders*:
  - `ImageCodecInfo: GetImageEncoders(), GetImageDecoders()`
- Konwersja obrazków

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

Przezroczystość

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

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

- Tryby składania *(Compositing modes)*

- Wartości *Alpha* w obrazkach

Teksty

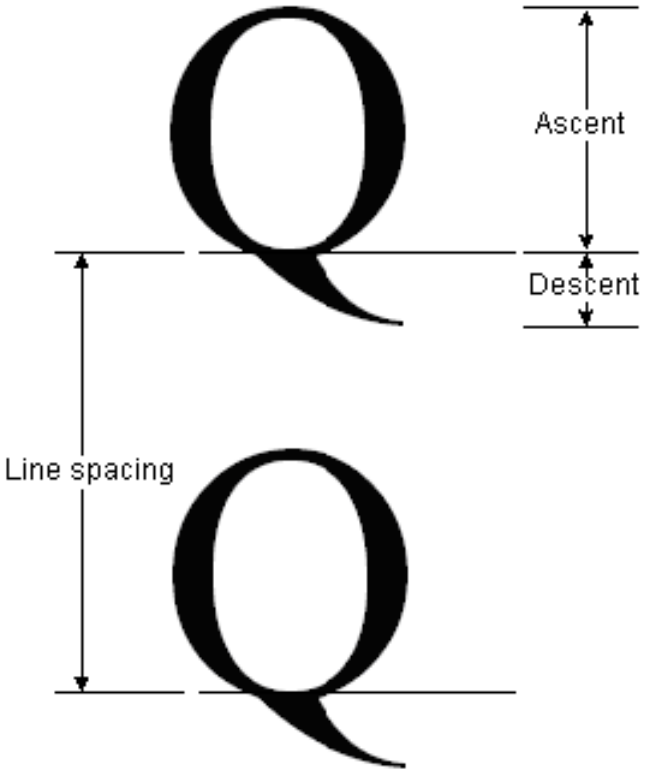
- `Graphics.DrawString()`
  - w określonym miejscu
  - w określonym prostokącie
- Formatowanie
  - wyrównanie – `StringFormat.SetAlignment()`, `StringFormat.SetLineAlignment()`
  - tabulatory – `StringFormat.SetTabStops()`
  - pionowy tekst – `StringFormat.SetFormatFlags()`
- Antyaliasing
  - `Graphics.SetTextRenderingHint()`

Fonty

- Tworzenie

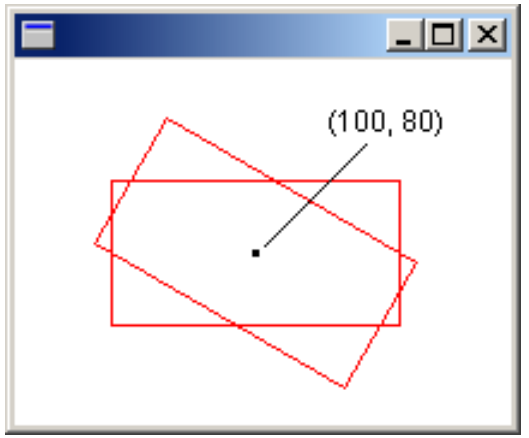
```
Font font = new Font("Arial", 16,
                     FontStyle.Regular);
```

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



Kontenery graficzne

- Stan obiektów klasy **`Graphics`**
  - powiązanie z kontekstem urządzenia
  - ustawienia dot. jakości
  - przekształcenia
  - region obcinania



- Użycie kontenerów graficznych

```
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);
```

Przekształcenia

- Klasa **`Matrix`**
- Proste metody klasy **`Graphics:`**
  - **`ScaleTransform()`**
  - **`RotateTransform()`**
  - **`TranslateTransform()`**
- Kolejność przekształceń jest znacząca

Regiony

- Klasa **`Region`**
- Elementy:
  - linie
  - wielokąty
  - krzywe
- Sprawdzenie zawierania punktu
  - **`Region.IsVisible(point, graphics)`**
- Obcinanie
  - **`Graphics.SetClip(region)`**

Zmiana kolorów

- Struktura **ColorMatrix**
- Klasa **ImageAttributes**

```
Image im = new Bitmap(@"d:\img.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);
```

Użycie GDI+ z kodu niezarządzanego

- Dołączyć nagłówek **gdiplus.h**

```
// !!! NIE DEFINIOWAC WIN32_LEAN_AND_MEAN
//#define WIN32_LEAN_AND_MEAN
#include <windows.h>
#include <gdiplus.h>
```

- Dołączyć przestrzeń nazw **Gdiplus**

```
using namespace Gdiplus;
```

- Dołączyć do projektu bibliotekę **gdiplus.lib**
  - w ustawieniach projektu albo użyć dyrektywy **#pragma**

```
// albo dodac w ustawieniach projektu
#pragma comment(lib, "gdiplus.lib")
```

Użycie GDI+ z kodu niezarządzanego c.d.

- Zainicjować GDI+

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

- Używać obiektów i metod GDI+

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

- Zwolnić zasoby używane przez GDI+

```
GdiplusShutdown(gdiplusToken);
```