Використання графічних можливостей технології Windows Forms

Графічні інструменти Windows об'єднані під однією назвою – GDI (Graphic Device Interface – інтерфейс графічних пристроїв). GDI – це підсистема Windows, призначена для виведення графічних зображень на екран і на принтер.

GDI+ – це новий набір програмних інтерфейсів, що використовується в .NET. *GDI+* є керованою альтернативою *Win32 GDI.*

Простори імен *GDI+*

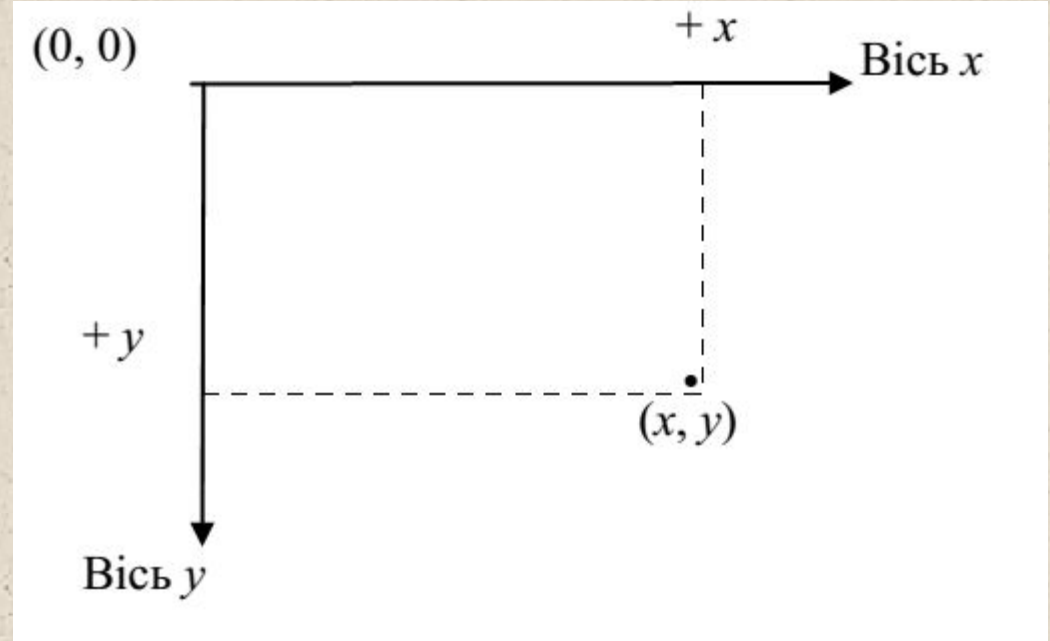
Простір імен	Опис
System.Drawing	Основний простір імен GDI+, що визначає безліч типів для основних операцій візуалізації (шрифти, пера, пензлі і т. д.), а також клас Graphics для підтримки Windows-графіки
System.Drawing.Drawing2D	Пропонує типи для складнішої двовимірної і векторної графіки – градієнтні пензлі, стилі кінців ліній для пер, геометричні перетворення і т. д.
System.Drawing.Imaging	Пропонує типи, що забезпечують обробку графічних зображень, – зміна палітри, витягання метаданих зображення, робота з метафайлами і т. д.
System.Drawing.Printing	Пропонує типи, що забезпечують відображення графіки на друкарській сторінці, безпосередню взаємодію з принтером і визначення повного формату завдання друку
System.Drawing.Text	Дозволяє працювати з колекціями системних шрифтів

Основні типи простору імен System.Drawing

Тип	Опис
1	2
Bitmap	Інкапсулює дані зображення і визначає набір методів для виконання різних операцій з цим зображенням
Brush, Brushes, SystemBrushes, HatchBrush, LinearGradientBrush, SolidBrush, TextureBrush	Об'єкти Brush (пензлі) використовуються для заповнення внутрішнього простору графічних форм (прямокутників, еліпсів або багатокутників). Тип Brush – це абстрактний базовий клас, решта типів є похідними від Brush і визначає різні набори можливостей. Додаткові типи Brush визначені в просторі імен System. Drawing. Drawing2D
Color	Структура Color визначає набір статичних полів, які можуть бути використані для настройки кольору
Font, FontFamily	Тип Font інкапсулює характеристики шрифту (ім'я, розмір, зображення і т.п.). FontFamily представляє набір шрифтів, які відносяться до одного сімейства, але мають деякі невеликі відмінності
Graphics	Цей найважливіший клас визначає набір методів для виведення тексту, зображень і геометричних фігур. Методи малювання класу вимагають об'єкт Pen (перо зображає контур фігури) або Brush (пензель створює залиті фігури) для візуалізації заданої фігури. Можна вважати цей тип еквівалентом типу Hdc в Win32

1	2
Icon, SystemIcons	Ці класи призначені для роботи з призначеними для користувача і системними піктограмами
Image, ImageAnimator	Image – це абстрактний базовий клас, який забезпечує можливості типів Bitmap, Icon і Cursor. ImageAnimator дозволяє проводити показ зображень (типів, похідних від Image) через вказані вами інтервали часу
Pen, Pens,SystemPens	Pen (перо) – це клас, за допомогою якого можна малювати прямі і криві лінії. У класі Pen визначений набір статичних властивостей, за допомогою яких можна отримати об'єкт Pen із заданими властивостями (наприклад, зі встановленим кольором)
Point, POINTF	Ці структури забезпечують роботу з координатами точки. Point працює із значеннями типу int, а POINTF — із значеннями типу float
Rectangle, RECTANGLEF	Ці структури призначені для роботи з прямокутними областями (int/float)
Size, SIZEF	Ці структури забезпечують роботу з розмірами (висотою і шириною). Size використовує значення типу int, а SIZEF – типу float

За умовчанням верхній лівий кут компоненту *GDI+ (*наприклад, *Form* або *Panel*) має координати (0, 0). Координата *x* – це відстань по горизонталі (вправо) від верхнього лівого кута. Координата *y* – це відстань по вертикалі (вниз) від верхнього лівого кута. Координати вимірюються в пікселях, що є найменшими одиницями дозволу монітора комп'ютера.



Можливості класу Graphics

У класі Graphics інкапсульовані поверхні малювання GDI+. Є три основних типи поверхонь малювання:

- Вікна і управляючі елементи на екрані.
- Сторінки, що посилаються на принтер.
- Растрові зображення в пам'яті.

У класі Graphics передбачені функції, які дозволяють малювати на будь-якій з цих поверхонь. Цей клас дозволяє також малювати дуги, криві, криві Безьє (Bezier), еліпси, малюнки, прямі, прямокутники і текст.

Клас System.Drawing.Graphics – це "вхід" у функціональні можливості GDI+. Всі можливості виведення зображень в GDI+ зосереджені саме в цьому класі. Можна вважати цей клас якимсь віртуальним пристроєм, на який проводиться виведення графіки.

За допомогою властивостей і методів класу Graphics можна малювати на поверхні видимих об'єктів, які включають цей клас і, відповідно, мають властивість Graphics. Наприклад, властивість Graphics мають такі об'єкти, як форма (Form), напис (Label), кнопка (Button).

Клас Graphics має велике число властивостей і методів, які дозволяють переміщатися по елементу управління, малювати графічні примітиви, копіювати зображення і їх окремі області, а також виводити текстову й іншу графічну інформацію. Клас Graphics забезпечує:

- завантаження і зберігання графічних зображень;
- створення нових і зміна зображень, що зберігаються, за допомогою пера, пензля і шрифту;
- малювання і зафарбовування різних фігур, ліній і текстів;
- комбінування різних зображень.

Деякі методи класу Graphics

Метод	Призначення
FromHdc(), FromHwnd(), FromImage()	Статичні методи, що забезпечують можливість отримання об'єкта Graphics з елементу управління або зображення
Clear()	Заповнює об'єкт Graphics вибраним користувачем кольором, видаляючи його попередній вміст
DrawArc(), DrawBezier(), DrawBeziers(), DrawCurve(), DrawEllipse(), DrawIcon(), DrawLines(), DrawLine(), DrawPie(), DrawPath(), DrawRectange(), DrawRectangles(), DrawString()	Ці методи (як і багато інших) призначені для виведення зображень і геометричних фігур
FillEllIpse(), FillPath(), FillPie, FillPolygon(), FillRectangle()	Ці методи призначені для заповнення внутрішніх областей графічних об'єктів

Як тільки потрібне оновлення вікна (унаслідок його перекриття або часткового псування), для нього генерується подія Paint (подія Paint успадкована від класу Control).

Подія Paint наступає, коли приходить повідомлення Windows про необхідність перемальовувати зіпсоване зображення. У обробнику цієї події і потрібно перемальовувати зображення. Проте, по обробникові події форми Paint перемальовувалося зображення всієї форми, а це може бути трудомістка операція.

Способи перемальовування:

1. Для малювання написати процедуру (метод форми), яку викликати в обробнику події Paint форми.

2. Перемальовування істотно прискориться, якщо перемальовувати тільки зіпсовану частину елементу управління.

До аргументів події Paint відноситься об'єкт PaintEventArgs, з якого можна отримати об'єкт Graphics для управління. Об'єкт Graphics необхідно отримувати для кожного звернення до методу Paint, тому що властивості графічного контексту, що представляються графічним об'єктом, можуть мінятися.

Об'єкт Graphics для деякого вікна можна одержати двома шляхами.

Перший полягає в перевизначенні події OnPaint() – віртуального методу, який клас Forms успадковує від класу control. В цьому випадку об'єкт Graphics одержуємо з PaintEventArgs, який передається разом з подією:

```
protected override void OnPaint(PaintEventArgs e)
{
Graphics g = e.Graphics;
// Малюємо тут
}
```

У деяких ситуаціях потрібно виконувати малювання у вікні безпосередньо, не чекаючи настання події OnPaint(). Це може виявитися актуальним у тому випадку, якщо створюємо код, призначений для вибору у вікні яких-небудь графічних об'єктів (аналогічний вибору ікон в Windows Explorer), або переносимо якийсь об'єкт за допомогою миші.

В цьому випадку доступ до об'єкта Graphics можна дістати, звертаючись до методу CreateGraphics() даної форми, який є ще одним методом, успадкованим класом Forms від класу Control:

```
protected void Forml_Click(object sender, System.EventArgs e)
{Graphics g = this.CreateGraphics();
// Малюємо тут
g.Dispose(); // Це важливий момент
}
```

Тепер для зображення фігур і рядків на формі доступна змінна g.

При розкритті, закритті вікна або зміні його розмірів автоматично генерується подія Paint для форми. Так само при відображенні будь-якого елементу управління (наприклад, TextBox або Button) програми генерується подія Paint для цього елементу управління. Подію Paint можна згенерувати примусово за допомогою виклику методу Invalidate, також успадкованого від класу Control. Даний метод оновлює клієнтську область елементу управління і неявно перемальовував всі графічні компоненти. У *.NET* містяться декілька перевантажених методів Invalidate, що дають можливість оновлення частин клієнтської області.

Виклик методу Invalidate з параметром Rectangle оновлює тільки область, позначену прямокутником, що підвищує продуктивність програми.

Управління кольором – структура Color.

Структура Color має поля і визначає методи і константи для маніпулювання з кольором.

Для завдання кольору використовується *ARGB-модель (A* – альфакомпонент прозорості, *R* – червоний компонент (кількість червоного кольору в підсумковому кольорі), *G* – зелений компонент, *B* – синій компонент). Всі чотири компоненти *ARGB-моделі* є байтами, що представляють цілі числа від 0 до 255.

Альфа-значення визначає непрозорість кольору. Якщо *A=0*, то колір прозорий, якщо *A=255* – колір насичений. Значення *A* між 0 і 255 дають зважений ефект поєднання *RGB-значення* кольору із значенням будь-якого фонового кольору, роблячи колір напівпрозорим. Існує можливість вибору з порядку 17 млн кольорів. Якщо монітор не здатний відобразити всі ці кольори, то буде вибраний колір, найближчий до заданого.

Для створення кольору, заснованого на червоному, зеленому, синьому і альфа-каналі, можна використовувати статичний метод *Color.FromArgb.*

Робота зі шрифтами

Основний клас, який використовується для роботи зі шрифтами в GDI+, – це клас System.Drawing.Font. Об'єкти цього класу представляють конкретні шрифти, встановлені на комп'ютері. У цьому класі передбачена безліч перевантажених конструкторів.

Зазвичай конструктор шрифту вимагає:

- назву гарнітури шрифту (font name);
- розмір гарнітури шрифту (font size);
- стиль гарнітури шрифту (font style з перерахування FontStyle).

Змінити властивості об'єкта Font не можна, для використання іншої гарнітури необхідно створити новий об'єкт Font.

Приклад.

Два конструктора, що використовуються найчастіше мають вигляд:

```
//Створюємо об'єкт Font, вказуючи ім'я шрифту і його розмір
Font f1 = new Font("Times New Roman", 12);
//Створюємо об'єкт Font, вказуючи ім'я, розмір і стиль
Font f2 = new Font("Courier New", 16, FontStyle.Bold FontStyle.Underline);
```

Виведення зображень (клас Image)

Для виведення зображень використовується клас Image простору імен System.Drawing. Тип Image визначає безліч властивостей і методів, які можна використовувати для настройки параметрів зображення, що виводиться.

Клас Image є абстрактним, і створювати об'єкти цього класу не можна. Зазвичай оголошені змінні Image присвоюються об'єктам класу Bitmap. Крім того, можна створювати об'єкти класу Bitmap безпосередньо і використовувати їх замість об'єктів класу Image.

Виведення отриманих зображень проводиться за допомогою спеціального методу класу Graphics, який називається, – DrawImage(). Цей метод багато разів перевантажений, тому в нашому розпорядженні безліч варіантів того, як помістити зображення в потрібне нам місце на формі. Клас Bitmap дозволяє виводити зображення, які зберігаються у файлах найрізноманітнійшого формату, – *bmp*, *jpg*, *gif*, *ico*.

Приклад. Розробити Windows-додаток, який будує графік функції y=sin(x).

```
private void Form1_Paint(object sender, PaintEventArgs e){
Graphics g = this.CreateGraphics();
Pen p = new Pen(Color.Green);
for (int i = 0; i < this.ClientSize.Width; i+=30)
g.DrawLine(p,i,0,i,this.ClientSize.Height);
for (int i = 0; i < this.ClientSize.Height; i+=30)
g.DrawLine(p,0,i,this.ClientSize.Width,i);
Pen p1 = new Pen(Color.Red,3);
double angle1 = 0,angle2;
double y1, y2;
for (int i = 1; i < this.Width; i++){
angle2 = i * 1.0 / 180 * Math.PI;
y1 = this.ClientRectangle.Height / 2 * (1 -Math.Sin(angle1));
y2 = this.ClientRectangle.Height / 2 * (1 -Math.Sin(angle2));
g.DrawLine(p1, i - 1, (int) y1, i, (int) y2);
angle1 = angle2;}}
private void Form1_Resize(object sender, EventArgs e){
this.Invalidate();
}
```