Работа с графикой в .NET

- Обзор пространств имен GDI+
- Пространство имен System.Drawing
- Служебные типы System.Drawing: Point(F), Rectangle(F), Region
- Вывод графики
- Класс Graphics
- Системы координат в GDI+
- Работа с цветом. Класс ColorDialog
- Работа с перьями. Класс Pen
- Работа с кистью
- Вывод изображений

Обзор пространств имен

GDI+

GDI (Graphic Device Interface) – интерфейс графических устройств).

GDI+ — это набор программных интерфейсов, используемый в .NET

Обзор пространств имен GDI+

System.Drawing – содержит основные типы для вывода графики (для работы со шрифтами, перьями, кистью).

System.Drawing.Drawing2D – содержит типы для выполнения операций с двумерной графикой (градиентная заливка, геометрические преобразования).

System.Drawing.Imaging – определены типы, которые позволяют напрямую работать с графическими изображениями.

System.Drawing.Printing – определяет типы для вывода графики на принтер и взаимодействия с принтером в целом.

System.Drawing.Text – позволяет работать с

Пространство имен System.Drawing

Bitmap. Инкапсулирует файл изображения и определяет набор методов для выполнения различных операций.
Brush – Объекты Brush используются для заполнения пространства внутри геометрических фигур.
Pen – это класс, при помощи которого можно рисовать прямые и кривые.
Color – определяет набор статических полей, которые могут быть использованы для настройки цвета.
Graphics. Этот важнейший класс определяет набор методов для вывода текста, изображений и геометрических фигур.
Image – это абстрактный базовый класс, который обеспечивает возможности типов Bitmap, Icon и Cursor.
Point – обеспечивает работу с координатами точки.
Region – определяет область, занятую геометрической

Перечисления System.Drawing

ContentAlignment. Определяет расположение содержимого в области вывода (слева, справа, по центру и т.п.)
FontStyle. Определяет свойства шрифта.
GraphicsUnit. Определяет единицы измерения для графического элемента (аналогично константам режима отображения в Win32).
KnownColor. Определяет дружественные имена системных цветов.
StringAlignment. Определяет выравнивание текстовой строки
StringFormatFlags. Определяет форматирование текстовых строк (например, содержит значения NoWrap, LineLimit и т. п.)
StringTrimminig. Определяет, как будут обрезаться строки, которые не помещаются полностью в отведенной им

Служебные типы System.Drawing

Служебные типы System.Drawing – указывают положение или область для вывода графического объекта.

Point – используется для передачи координат (x, y).
Rectangle – определяет координаты двух точек (верхний левый и нижний правый углы прямоугольника).
Region – необходим для работы с непрямоугольными областями.
Size – определяет размер прямоугольной области.

Пример: Point pt = new Point(100, 72);

Служебные типы

System.Drawing

В классе **Rectangle** предусмотрен метод **Contains()**. Этот метод позволяет определить, попадает ли точка (или прямоугольник) с указанными координатами в область, занятую прямоугольником.

Пример:

```
Rectangle rg = new Rectangle(0, 0, 200, 300);
Rectangle rect = new Rectangle(new Point(0, 0), new
Size(20, 20));
if (rg.Contains(rect))
    label1.Text = "Вы попали в область
прямоугольника";
```

Служебные типы

System.Drawing

Класс **Region** представляет собой внутреннюю область, занятую геометрической фигурой.

Пример. Получить область прямоугольника размером 100 на 100 пикселов:

```
Rectangle r = new Rectangle(0, 0, 100, 100);
Region rgn = new Region(r);
```

Методы класса Region

Complement – дополняет объект Region другими графическими объектами, которые с ним не пересекаются.
Exclude – исключает область, занимаемую другим графическим объектом, из области объекта Region.
GetBounds – возвращает объект класса RectangleF, представляющий прямоугольник, в который точно вписана область, занимаемая объектом Region.
Intersect – возвращает область наложения друг на друга исходного и указанного объектов Region.
Translate – сдвигает координаты объекта Region.
Union – объединяет указанный объект Region с другим графическим объектом
Xor – объединяет указанный объект Region с другим графическим объектом, исключая при этом область пересечения этих объектов

Вывод графики

Для вывода графики на форму, необходимо заместить виртуальный метод OnPaint().

Пример. Вывести на форму желтый прямоугольник:

```
protected override void OnPaint(PaintEventArgs e)
{
    Rectangle rg = new Rectangle(0, 0, 200, 300);
    Graphics g = Graphics.FromHwnd(this.Handle);
    Brush brush1 = Brushes.Yellow;
    g.FillRectangle(brush1, rg);
}
```

Вывод графики

Метод Invalidate() – инициирует перерисовку формы.

```
//перерисовать прямоугольную область на форме
private void UpdateArea()
{
    Rectangle myRect = new Rectangle(0, 0, 75, 150);
    Invalidate(myRect);
}
```

Класс Graphics

Методы класса Graphics:

FromHdc, FromHwnd, FromImage – обеспечивают возможность получения объекта Graphics из элемента управления или изображения.
Clear – заполняет объект Graphics выбранным цветом, удаляя его предыдущее содержимое.
DrawArc, DrawEllipse, … – предназначены для вывода изображений и геометрических фигур.
FillElllpse, FillPolygon, … – предназначены для заполнения внутренних областей графических объектов.

Системы координат в GDI+

Система, принятая по умолчанию, использует в качестве единицы измерения пикселы, а в качестве исходной точки – верхний левый угол.
Координата X определяет смещение вправо, а координата Y – смещение вниз.

Пример. Вывод прямоугольника, размером 90x90, отстоящий от верхнего левого края формы на 10 пикселов вниз и вправо:

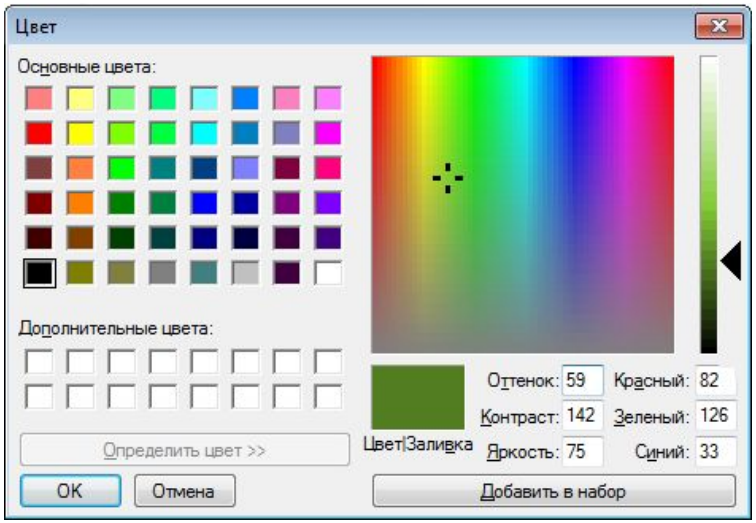
```
protected override void OnPaint(PaintEventArgs e)
{
    e.Graphics.DrawRectangle(new Pen(Color.Red), 10, 10, 100, 100);
}
```

Работа с цветом

Структура **Color** позволяет задать цвет в системе ARGB (aplha-red-green-blue. Альфа-канал – прозрачность.
Пример:

```
Color c = Color.SeaGreen;
```

Класс **ColorDialog** обеспечивает пользователей приложения диалоговым окном для выбора цвета.



Работа с цветом

Пример. Меняем цвет фона и выводим информацию о выбранном цвете:

```
Color currColor = new Color();

ColorDialog colorDlg = new ColorDialog();
if (colorDlg.ShowDialog() != DialogResult.Cancel)
{
   currColor = colorDlg.Color;
   this.BackColor = currColor;

   string strARGB = colorDlg.Color.ToString();
   MessageBox.Show("Color Is: " + strARGB);
}
```

Работа с перьями. Класс Pen

Методы DrawXXXX() класса Graphics, принимающие в качестве
параметра объект Pen:

DrawArc Этот метод предназначен для вывода дуги. Он принимает в качестве параметров объект Реп и данные, позволяющие построить эллипс и выделить на нем дугу.

DrawBezier Метод для вывода кубической кривой Безье (нескольких кривых) по четырем точкам.

DrawCurve Метод для вывода кривой на основе массива точек.

DrawEllipse Метод для вывода эллипса, вписанного в прямоугольник (передаются координаты прямоугольника).

DrawLine (DrawLines) Эти методы соединяют прямыми линиями точки (массив точек).

DrawPath Этот метод выводит коллекцию прямых и кривых линий при помощи типа GraphicsPath, определенном в

Работа с перьями. Класс Pen

Свойства класса Pen:

Brush Определяет кисть, используемую данным объектом.
Color Определяет цвет создаваемых объектом Реп линий.
DashOffset Устанавливает смещение начала пунктира относительно исходной точки пунктирной линии.
DashPattem Позволяет получить или установить массив штрихов и пробелов между ними для пунктирных линий.
DashStyle Позволяет получить или установить стиль для пунктирных линий, создаваемых при помощи данного объекта.
LineJoin Позволяет получить или установить стиль объединения при пересечении двух линий, выводимых данным объектом.
PenType Позволяет получить стиль линий, выводимых при

Работа с перьями. Класс Pen

```
Graphics g = e.Graphics;
// Создаем большое перо синего цвета
Pen bluePen = new Pen(Color.Blue, 20);
// Создаем еще одно перо при помощи заготовок из коллекции Pens
Pen pen2 = Pens.Firebrick;
// Выводим при помощи созданных нами перьев геометрические фигуры
g.DrawEllipse(bluePen, 10, 10, 100, 100);
g.DrawLine(pen2, 10, 130, 110, 130);
g.DrawPie(Pens.Black, 150, 10, 120, 150, 90, 80);
// Выводим многоугольник пурпурного цвета
Pen pen3 = new Pen(Color.Purple, 5);
pen3.DashStyle = DashStyle.DashDotDot;
g.DrawPolygon(pen3, new Point[] { new Point (30, 140),
    new Point(265, 200),
    new Point(100, 225),
    new Point(190, 190),
    new Point(50, 330),
    new Point(20, 180),});
// Добавляем прямоугольник со вписанным нами текстом
Rectangle r = new Rectangle(150, 10, 130, 60);
g.DrawRectangle(Pens.Blue, r);
g.DrawString("Привет, как дела?", new Font("Arial", 12), Brushes.Black, r);
```

Работа с перьями. Класс Pen

Значения DashStyle:

Custom – Пользовательский стиль пунктира
Dash – Штриховая линия
DashDot – Штрихпунктирная линия: штрих -- точка — штрих
DashDotDot – Штрихпунктирная линия: штрих — точка — точка — штрих
Dot – Пунктир из одних точек
Solid – Сплошная линия

Работа с кистью

Абстрактный класс **Brush**

Методы:

FillClosedCurve Закрашивает область внутри замкнутой кривой

FillEllipse Закрашивает область внутри эллипса

FillPath Закрашивает область внутри траектории

FillPie Закрашивает область внутри сегмента эллипса

FillPolygon Закрашивает область внутри многоугольника

FillRectangle Закрашивают область внутри прямоугольника

FillRegion Закрашивает внутреннюю область объекта Region (Region – это внутренняя область геометрической фигуры)

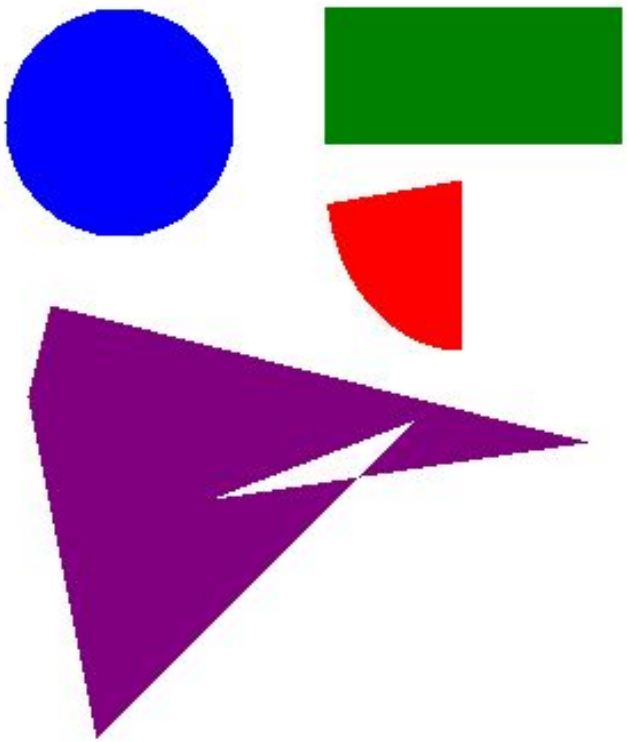
Работа с кистью

Пример:

```
Graphics g = e.Graphics;
// Создаем кисть синего цвета
SolidBrush blueBrush = new SolidBrush(Color.Blue);
// Закрашиваем геометрические фигуры
g.FillEllipse(blueBrush, 10, 10, 100, 100);
g.FillPie(Brushes.Red, 150, 10, 120, 150, 90, 80);
// Закрашиваем многоугольник пурпурным цветом
SolidBrush brush3 = new SolidBrush(Color.Purple);
g.FillPolygon(brush3, new Point[]{ new Point(30, 140),
    new Point (265, 200),
    new Point(100, 225),
    new Point (190, 190),
    new Point (50, 330),
    new Point (20, 180)});
// прямоугольник зеленого цвета:
Rectangle r = new Rectangle(150, 10, 130, 60);
g.FillRectangle(Brushes.Green, r);
```

Работа с кистью

Результат:



Работа с кистью

Класс **HatchBrush** – штрихованные кисти

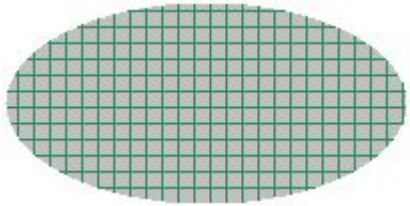
BackwardDiagonal Диагональная штриховка с наклоном вправо
Crass “Крестообразная” штриховка, состоящая из пересекающихся вертикальных и горизонтальных линий
DiagonalCross Еще одна разновидность “крестообразной” штриховки, состоящая из пересекающихся диагональных линий
Forward Diagonal Диагональная штриховка с наклоном влево
Horizontal Горизонтальная штриховка
Pattern Штриховка, которая создается на основе указанного пользователем растрового изображения
Solid Обычная “плотная” кисть без всякой штриховки

Работа с кистью

Пример. Закрашиваем эллипс штриховой кистью:

```
HatchBrush  theBrush = new HatchBrush(HatchStyle.Cross,
Color.SeaGreen, Color.Silver);
g.FillEllipse(theBrush, 250, 100, 200, 100);
```

Вывод:



Вывод изображений

Класс **Image** является абстрактным.

FromFile() – предназначен для создания объекта Image из файла.
FromHbitmap() Создает объект Bitmap на основе Window handle.
Palette Это свойство возвращает объект ColorPalette, представляющий цветовую палитру, использованную для данного графического изображения.
GetBounds() Возвращает прямоугольник, представляющий текущую область, занятую изображением.

Вывод изображений

Пример:

```
Image img = new Bitmap("c:\\image.jpg");
g.DrawImage(img, 10, 10, 200, 260);
```

Вывод изображений

Пример. Перетаскивание элемента PictureBox:

```
bool  isDragging;
private void pictureBox1_MouseDown(object sender, MouseEventArgs e)
{
   isDragging = true;
}
private void pictureBox1_MouseMove(object sender, MouseEventArgs e)
{
   if (isDragging)
   {
      pictureBox1.Top = pictureBox1.Top + (e.Y);
      pictureBox1.Left = pictureBox1.Left + (e.X);
   }
}
private void pictureBox1_MouseUp(object sender, MouseEventArgs e)
{
   isDragging = false;
}
```

Лабораторная работа 10

Форма переменного размера содержит единственный компонент PictureBox. Меню формы содержит два подменю: *«Image»* (команды *«New», «Open», «Save»*) и *«Draw» («Image»).*

Команда *«New»* позволяет создать новое изображение; команда *«Open»* позволяет загрузить существующее изображение. По команде *«Save»* - существующее изображение сохраняется.

Новый элемент изображения можно добавить в нужную позицию с помощью щелчка мышью на элементе Image.

Команда *«Image»* отображает подменю третьего уровня, позволяющее выбрать вид изображаемого элемента.

Виды элементов изображения:

1. **рисование круга случайного размера и цвета;**
2. **рисование набора концентрических кругов;**
3. **рисование последовательности кругов уменьшающегося радиуса, центры которых расположены на одном отрезке.** Для указания данного элемента мышью требуется щелкнуть на начальной и конечной позиции нужного отрезка.