Шпаргалка по графике в Windows Forms

1. Основные классы и пространства имен

- System.Drawing: Основное пространство имен для работы с графикой.

Классы:

- Graphics: Для рисования на формах и элементах управления.
- Pen: Для рисования контуров.
- Brush: Для заливки фигур.
  - SolidBrush: Заливка сплошным цветом.
  - HatchBrush: Заливка с узором.
  - LinearGradientBrush: Линейный градиент.
- Image: Абстрактный класс для работы с изображениями.
- Bitmap: Для работы с растровыми изображениями.
- Color: Для работы с цветами.

2. Основные операции с графикой

Получение объекта Graphics:

Событие Paint:

```
private void MainForm_Paint(object sender, PaintEventArgs e) {
    Graphics g = e.Graphics;
    // Рисование здесь
}
```

3. Класс Graphics.

Основные свойства класса Graphics:

- Clip — определяет область отсечения для рисования.
- DpiX и DpiY — возвращают горизонтальное и вертикальное разрешение (в точках на дюйм) для объекта Graphics.
- SmoothingMode — определяет качество сглаживания при рисовании линий и кривых.
- TextRenderingHint — определяет качество рендеринга текста.

Основные методы класса Graphics:

- DrawLine(Pen pen, int x1, int y1, int x2, int y2) — рисует линию между двумя точками.
- DrawRectangle(Pen pen, int x, int y, int width, int height) — рисует прямоугольник.
- FillRectangle(Brush brush, int x, int y, int width, int height) — заливает прямоугольник цветом.
- DrawEllipse(Pen pen, int x, int y, int width, int height) — рисует эллипс.
- FillEllipse(Brush brush, int x, int y, int width, int height) — заливает эллипс цветом.
- DrawString(string s, Font font, Brush brush, float x, float y) — рисует текст.
- Clear(Color color) — очищает поверхность рисования и заполняет её указанным цветом.
- DrawImage(Image image, int x, int y) — рисует изображение.
- DrawPolygon(Pen pen, Point[] points) — рисует многоугольник.
- FillPolygon(Brush brush, Point[] points) — заливает многоугольник цветом.

```
private void Form1_Paint(object sender, PaintEventArgs e)
{
    Graphics g = e.Graphics;

    // Рисуем линию
    Pen pen = new Pen(Color.Blue, 3);
    g.DrawLine(pen, 10, 10, 200, 10);
```

```
// Рисуем прямоугольник
g.DrawRectangle(pen, 10, 30, 100, 50);

// Заливаем прямоугольник
Brush brush = new SolidBrush(Color.Red);
g.FillRectangle(brush, 120, 30, 100, 50);

// Рисуем эллипс
g.DrawEllipse(pen, 10, 100, 100, 50);

// Заливаем эллипс
g.FillEllipse(brush, 120, 100, 100, 50);

// Рисуем текст
Font font = new Font("Arial", 12);
g.DrawString("Hello, World!", font, Brushes.Black, 10, 170);

// Рисуем изображение
Image image = Image.FromFile("path_to_image.png");
g.DrawImage(image, 10, 200);

// Рисуем треугольник
Point[] points = { new Point(10, 300), new Point(50, 350), new Point(100, 300) };
g.DrawPolygon(pen, points);

// Заливаем треугольник
g.FillPolygon(brush, points);
```

4. Класс Pen (перо).

Класс Pen в C# используется для определения стиля, ширины и цвета линий, которые рисуются с помощью объекта Graphics. Он предоставляет свойства и методы для настройки внешнего вида линий, таких как цвет, толщина, тип штриха и другие параметры.

Основные свойства класса Pen:

- Color — определяет цвет линии.
- Width — задает толщину линии.
- DashStyle — определяет стиль штриховки линии (например, сплошная, пунктирная и т.д.).
- StartCap и EndCap — определяют стиль начала и конца линии (например, квадратный, круглый и т.д.).
- LineJoin — определяет стиль соединения линий (например, скругление, угол и т.д.).
- DashOffset — задает расстояние от начала линии до начала штриховки.
- Brush — определяет кисть, используемую для заливки линии (если линия имеет градиент или текстуру).

Основные методы класса Pen:

- Dispose() — освобождает ресурсы, используемые объектом Pen.
- SetLineCap(LineCap startCap, LineCap endCap, DashCap dashCap) — задает стиль начала, конца и штриховки линии.

```
Graphics g = e.Graphics;

// Пример 1: Простая линия
Pen bluePen = new Pen(Color.Blue, 5); // Синяя линия толщиной 5
g.DrawLine(bluePen, 10, 10, 200, 10);
```

```
// Пример 2: Пунктирная линия
Pen redDashPen = new Pen(Color.Red, 3);
redDashPen.DashStyle = DashStyle.Dash; // Пунктирная линия
g.DrawLine(redDashPen, 10, 30, 200, 30);

// Пример 3: Линия с закругленными концами
Pen greenRoundPen = new Pen(Color.Green, 8);
greenRoundPen.StartCap = LineCap.Round; // Закругленный конец
greenRoundPen.EndCap = LineCap.Round; // Закругленный конец
g.DrawLine(greenRoundPen, 10, 60, 200, 60);

// Пример 4: Линия с градиентом
LinearGradientBrush gradientBrush = new LinearGradientBrush(
    new Point(10, 90), new Point(200, 90),
    Color.Yellow, Color.Purple); // Градиент от желтого к фиолетовому
Pen gradientPen = new Pen(gradientBrush, 10);
g.DrawLine(gradientPen, 10, 90, 200, 90);

// Пример 5: Линия с соединением "скругление"
Pen orangePen = new Pen(Color.Orange, 10);
orangePen.LineJoin = LineJoin.Round; // Скругление соединений
Point[] points = { new Point(10, 120), new Point(100, 150), new Point(200, 120) };
g.DrawLines(orangePen, points);

// Пример 6: Линия с пользовательским стилем штриховки
Pen customDashPen = new Pen(Color.Brown, 5);
customDashPen.DashPattern = new float[] { 10, 5, 2, 5 }; // Пользовательский шаблон штриховки
g.DrawLine(customDashPen, 10, 170, 200, 170);
```

Объект Pen должен быть освобожден после использования с помощью метода Dispose(), чтобы избежать утечек ресурсов. В современных версиях C# это можно сделать с помощью блока using.

```
// Создание пера
using(Pen pen = new Pen(Color.Blue, 2))  // Цвет и толщина
{
    pen.DashStyle = DashStyle.Dash;  // Пунктирная линия
    g.DrawLine(pen, 0, 0, 100, 100);
}
```

В C# (Windows Forms) существуют предопределенные перья (Pens), которые предоставляют стандартные объекты Pen для рисования линий с различными цветами. Эти перья имеют фиксированную толщину (1 пиксель) и используются для упрощения работы с часто используемыми цветами.

```
Graphics g = e.Graphics;

// Использование предопределенных перьев
g.DrawLine(Pens.Red, 10, 10, 200, 10); // Красная линия
g.DrawLine(Pens.Green, 10, 30, 200, 30); // Зеленая линия
g.DrawLine(Pens.Blue, 10, 50, 200, 50); // Синяя линия
g.DrawLine(Pens.Black, 10, 70, 200, 70); // Черная линия
g.DrawLine(Pens.Yellow, 10, 90, 200, 90); // Желтая линия
```

5. Класс Brush (кисть).

Основные свойства и методы класса Brush

Свойства:

- Disposed — указывает, был ли объект Brush освобожден.
- CanRaiseEvents — определяет, может ли объект вызывать события (наследуется от Component).

Методы:

- Dispose() — освобождает ресурсы, используемые объектом Brush.
- Clone() — создает копию объекта Brush.

Основные производные классы Brush:

- SolidBrush — заливает фигуру сплошным цветом.
- HatchBrush — заливает фигуру узором (например, штриховкой).
- LinearGradientBrush — заливает фигуру линейным градиентом.
- TextureBrush — заливает фигуру текстурой (изображением).
- PathGradientBrush — заливает фигуру градиентом, который изменяется вдоль пути.

```
Graphics g = e.Graphics;

// Пример 1: SolidBrush (сплошная заливка)
SolidBrush solidBrush = new SolidBrush(Color.Red);
g.FillRectangle(solidBrush, 10, 10, 100, 50);

// Пример 2: HatchBrush (штриховка)
HatchBrush hatchBrush = new HatchBrush(HatchStyle.Cross, Color.Blue, Color.Yellow);
g.FillRectangle(hatchBrush, 120, 10, 100, 50);

// Пример 3: LinearGradientBrush (линейный градиент)
LinearGradientBrush gradientBrush = new LinearGradientBrush(
    new Point(10, 70), new Point(200, 70),
    Color.Green, Color.Yellow);
g.FillRectangle(gradientBrush, 10, 70, 200, 50);

// Пример 4: TextureBrush (текстура)
Image image = Image.FromFile("path_to_image.png");
TextureBrush textureBrush = new TextureBrush(image);
g.FillRectangle(textureBrush, 10, 130, 200, 50);

// Пример 5: PathGradientBrush (градиент вдоль пути)
GraphicsPath path = new GraphicsPath();
path.AddEllipse(10, 190, 100, 50);
PathGradientBrush pathBrush = new PathGradientBrush(path);
pathBrush.CenterColor = Color.Red;
pathBrush.SurroundColors = new Color[] { Color.Blue };
g.FillPath(pathBrush, path);
```

Пояснение к примерам

SolidBrush:

- Создается кисть сплошного красного цвета.
- Заливается прямоугольник с координатами (10, 10) и размерами 100x50.

HatchBrush:

- Создается кисть с узором HatchStyle.Cross (перекрестная штриховка), где основной цвет — синий, а фоновый — желтый.
- Заливается прямоугольник с координатами (120, 10) и размерами 100x50.

LinearGradientBrush:

- Создается кисть с линейным градиентом от зеленого к желтому.
- Градиент идет от точки (10, 70) до точки (200, 70).
- Заливается прямоугольник с координатами (10, 70) и размерами 200x50.

TextureBrush:

- Создается кисть с текстурой из изображения, загруженного из файла.
- Заливается прямоугольник с координатами (10, 130) и размерами 200x50.

PathGradientBrush:

- Создается путь в виде эллипса.
- Создается кисть с градиентом, где центр эллипса красный, а окружающие цвета — синие.
- Заливается эллипс с координатами (10, 190) и размерами 100x50.

Основные свойства и методы для конкретных кистей

SolidBrush

- Color — определяет цвет заливки.

HatchBrush

- BackgroundColor — цвет фона.

- ForegroundColor — цвет узора.
- HatchStyle — стиль штриховки.

LinearGradientBrush
- LinearColors — массив цветов градиента.
- Rectangle — область, к которой применяется градиент.
- WrapMode — определяет, как градиент повторяется за пределами области.

TextureBrush
- Image — изображение, используемое в качестве текстуры.
- WrapMode — определяет, как текстура повторяется за пределами области.

PathGradientBrush
- CenterColor — цвет в центре градиента.
- SurroundColors — массив цветов вокруг градиента.
- FocusScales — определяет масштабирование центра градиента.

Примечание.
Кисти (Brush) должны быть освобождены после использования с помощью метода Dispose(), чтобы избежать утечек ресурсов. В современных версиях C# это можно сделать с помощью блока using.

6. Класс Font.
Основные свойства класса Font:
- Name — возвращает имя шрифта (например, "Arial").
- Size — возвращает размер шрифта в единицах, указанных в свойстве Unit.
- Style — возвращает стиль шрифта (например, FontStyle.Bold, FontStyle.Italic).
- Bold — указывает, является ли шрифт полужирным (true или false).
- Italic — указывает, является ли шрифт курсивом (true или false).
- Underline — указывает, подчеркнут ли шрифт (true или false).
- Strikeout — указывает, зачеркнут ли шрифт (true или false).
- Unit — возвращает единицы измерения размера шрифта (например, GraphicsUnit.Point).
- Height — возвращает высоту шрифта в пикселях.
- FontFamily — возвращает объект FontFamily, связанный с шрифтом.

Основные методы класса Font:
- Dispose() — освобождает ресурсы, используемые объектом Font.
- Clone() — создает копию объекта Font.
- ToLogFont(object logFont) — преобразует шрифт в структуру LOGFONT (используется для взаимодействия с WinAPI).

Примеры использования класса Font

```
Graphics g = e.Graphics;

// Пример 1: Простой шрифт
Font font1 = new Font("Arial", 16); // Шрифт Arial, размер 16
g.DrawString("Hello, World!", font1, Brushes.Black, 10, 10);
```

```
// Пример 2: Полужирный и курсивный шрифт
Font font2 = new Font("Times New Roman", 20, FontStyle.Bold | FontStyle.Italic);
g.DrawString("Bold and Italic", font2, Brushes.Blue, 10, 50);

// Пример 3: Подчеркнутый и зачеркнутый шрифт
Font font3 = new Font("Verdana", 18, FontStyle.Underline | FontStyle.Strikeout);
g.DrawString("Underline and Strikeout", font3, Brushes.Red, 10, 100);

// Пример 4: Шрифт с использованием GraphicsUnit
Font font4 = new Font("Courier New", 24, FontStyle.Regular, GraphicsUnit.Pixel);
g.DrawString("Custom Unit (Pixel)", font4, Brushes.Green, 10, 150);

// Пример 5: Использование FontFamily
FontFamily fontFamily = new FontFamily("Georgia");
Font font5 = new Font(fontFamily, 22, FontStyle.Regular);
g.DrawString("Font Family Example", font5, Brushes.Purple, 10, 200);
```

Пояснение к примерам

Простой шрифт:

- Создается шрифт "Arial" размером 16.
- Метод DrawString рисует текст "Hello, World!" черным цветом на координатах (10, 10).

Полужирный и курсивный шрифт:

- Создается шрифт "Times New Roman" размером 20 с комбинированным стилем (FontStyle.Bold | FontStyle.Italic).
- Рисуется текст "Bold and Italic" синим цветом на координатах (10, 50).

Подчеркнутый и зачеркнутый шрифт:

- Создается шрифт "Verdana" размером 18 с комбинированным стилем (FontStyle.Underline | FontStyle.Strikeout).
- Рисуется текст "Underline and Strikeout" красным цветом на координатах (10, 100).

Шрифт с использованием GraphicsUnit:

- Создается шрифт "Courier New" размером 24 с единицей измерения GraphicsUnit.Pixel.
- Рисуется текст "Custom Unit (Pixel)" зеленым цветом на координатах (10, 150).

Использование FontFamily:

- Создается объект FontFamily для шрифта "Georgia".
- На его основе создается шрифт размером 22.
- Рисуется текст "Font Family Example" фиолетовым цветом на координатах (10, 200).

Работа со шрифтом (текст с тенью)

```
private void Form1_Paint(object sender, PaintEventArgs e)
{
    Graphics g = e.Graphics;

    // Создаем шрифт Times New Roman размером 24 пункта жирным начертанием
    Font font = new Font("Times New Roman", 24, FontStyle.Bold);

    // Текст для отображения
    string text = "Текст с тенью";

    // Позиция текста
    Point position = new Point(50, 50);

    // Рисуем тень
    using (SolidBrush shadowBrush = new SolidBrush(Color.FromArgb(128, 0, 0, 0)))
    {
        /*
        * Тень:
        * - Color.FromArgb(128, 0, 0, 0): Полупрозрачный черный цвет (альфа-канал 128).
        * - Смещение на 5 пикселей вправо и вниз для создания эффекта тени.
        */
        g.DrawString(text, font, shadowBrush,
                     new Point(position.X + 5, position.Y + 5));
    }

    // Рисуем основной текст
    using (SolidBrush textBrush = new SolidBrush(Color.Yellow))
    {
        /*
        * Основной текст:
        * - Желтый цвет текста.
        * - Рисуется поверх тени для создания эффекта объема.
        */
        g.DrawString(text, font, textBrush, position);
    }
}
```

Основные стили шрифта (FontStyle)

- FontStyle.Regular — обычный шрифт.
- FontStyle.Bold — полужирный шрифт.
- FontStyle.Italic — курсив.
- FontStyle.Underline — подчеркнутый шрифт.
- FontStyle.Strikeout — зачеркнутый шрифт.

Единицы измерения шрифта (GraphicsUnit)

- GraphicsUnit.Point — размер шрифта в пунктах (по умолчанию).
- GraphicsUnit.Pixel — размер шрифта в пикселях.
- GraphicsUnit.Inch — размер шрифта в дюймах.
- GraphicsUnit.Millimeter — размер шрифта в миллиметрах.

Примечания

Объект Font должен быть освобожден после использования с помощью метода Dispose(), чтобы избежать утечек ресурсов. В современных версиях C# это можно сделать с помощью блока using. Шрифты можно комбинировать с кистями (Brush) для заливки текста различными цветами, градиентами или текстурами.

Пример с использованием using:

```
using (Font font = new Font("Arial", 16))
{
    g.DrawString("Hello, World!", font, Brushes.Black, 10, 10);
}
```

7. Класс Color.

Класс Color в C# используется для представления цвета в формате ARGB (альфа, красный, зеленый, синий).

Основные свойства класса Color:

- A — возвращает значение альфа-канала (прозрачность) в диапазоне от 0 (полностью прозрачный) до 255 (полностью непрозрачный).
- R — возвращает значение красного канала в диапазоне от 0 до 255.
- G — возвращает значение зеленого канала в диапазоне от 0 до 255.
- B — возвращает значение синего канала в диапазоне от 0 до 255.
- Name — возвращает имя цвета (например, "Red" для Color.Red).
- IsKnownColor — указывает, является ли цвет предопределенным (из списка KnownColor).
- IsNamedColor — указывает, является ли цвет именованным (например, "Red", "Blue").
- IsSystemColor — указывает, является ли цвет системным (например, SystemColors.Control).

Основные методы класса Color:

- FromArgb(int alpha, int red, int green, int blue) — создает цвет с указанными значениями альфа, красного, зеленого и синего каналов.
- FromArgb(int alpha, Color baseColor) — создает цвет на основе существующего цвета с новым значением альфа-канала.
- FromArgb(int red, int green, int blue) — создает непрозрачный цвет с указанными значениями красного, зеленого и синего каналов.
- FromKnownColor(KnownColor color) — создает цвет из предопределенного списка KnownColor.
- FromName(string name) — создает цвет по его имени (например, "Red").
- ToArgb() — возвращает значение цвета в формате ARGB (32-битное целое число).
- ToString() — возвращает строковое представление цвета.

Предопределенные цвета.

Класс Color содержит множество статических свойств, которые возвращают предопределенные цвета. Например:

- Color.Red
- Color.Green
- Color.Blue
- Color.Yellow
- Color.Black
- Color.White
- Color.Transparent

Примеры использования класса Color

```
Graphics g = e.Graphics;

// Пример 1: Использование предопределенного цвета
SolidBrush redBrush = new SolidBrush(Color.Red);
g.FillRectangle(redBrush, 10, 10, 100, 50);

// Пример 2: Создание цвета с помощью FromArgb
Color customColor = Color.FromArgb(128, 255, 0, 0); // Полупрозрачный красный
SolidBrush customBrush = new SolidBrush(customColor);
g.FillRectangle(customBrush, 120, 10, 100, 50);
```

```
// Пример 3: Использование системного цвета
SolidBrush systemBrush = new SolidBrush(SystemColors.Control);
g.FillRectangle(systemBrush, 10, 70, 100, 50);

// Пример 4: Использование цвета по имени
Color namedColor = Color.FromName("Blue");
SolidBrush namedBrush = new SolidBrush(namedColor);
g.FillRectangle(namedBrush, 120, 70, 100, 50);

// Пример 5: Использование прозрачного цвета
SolidBrush transparentBrush = new SolidBrush(Color.Transparent);
g.FillRectangle(transparentBrush, 10, 130, 100, 50);
```

Пояснение к примерам:
Использование предопределенного цвета:

- Создается кисть (SolidBrush) с цветом Color.Red.
- Заливается прямоугольник с координатами (10, 10) и размерами 100x50.

Создание цвета с помощью FromArgb:

- Создается полупрозрачный красный цвет с альфа-каналом 128.
- Заливается прямоугольник с координатами (120, 10) и размерами 100x50.

Использование системного цвета:

- Создается кисть с системным цветом (SystemColors.Control).
- Заливается прямоугольник с координатами (10, 70) и размерами 100x50.

Использование цвета по имени:

- Создается цвет по имени "Blue".
- Заливается прямоугольник с координатами (120, 70) и размерами 100x50.

Использование прозрачного цвета:

- Создается кисть с прозрачным цветом (Color.Transparent).
- Заливается прямоугольник с координатами (10, 130) и размерами 100x50 (прямоугольник будет прозрачным).

Примечания

- Класс Color неизменяем, то есть после создания объекта Color его свойства нельзя изменить.
- Для работы с прозрачностью используется альфа-канал (значение от 0 до 255).
- Предопределенные цвета (Color.Red, Color.Blue и т.д.) удобны для быстрого доступа к часто используемым цветам.
- Системные цвета (SystemColors) используются для согласования с текущей темой операционной системы.

8. Работа с компонентом PictureBox
Основные свойства:

- Image: для задания изображения.
- SizeMode: определяет, как изображение будет отображаться.
    - Normal: натуральный размер.
    - StretchImage: растягивает изображение.
    - Zoom: масштабирует пропорционально.

Методы:

- Load(string path): загружает изображение.
- LoadAsync(string path): асинхронная загрузка изображения.

```
// Создание в коде
PictureBox pictureBox = new PictureBox();
pictureBox.Size = new Size(200, 200);
pictureBox.SizeMode = PictureBoxSizeMode.StretchImage;

// Загрузка изображения
pictureBox.Image = Image.FromFile("picture.jpg");
// или
pictureBox.Load("picture.jpg");

// Асинхронная загрузка
pictureBox.LoadAsync("picture.jpg");

// Режимы отображения
pictureBox.SizeMode = PictureBoxSizeMode.Normal;      // Обычный размер
pictureBox.SizeMode = PictureBoxSizeMode.StretchImage;// Растянуть
pictureBox.SizeMode = PictureBoxSizeMode.AutoSize;    // Автоматический размер
pictureBox.SizeMode = PictureBoxSizeMode.CenterImage; // По центру
pictureBox.SizeMode = PictureBoxSizeMode.Zoom;
```

9. Классы Point, PointF, Size, Rectangle

Описание : Классы для работы с геометрическими объектами:

Point: Представляет точку с целочисленными координатами.

PointF: Представляет точку с координатами типа float.

Size: Представляет размер (ширину и высоту).

Rectangle: Представляет прямоугольник с координатами и размерами.

Пример использования :

```
Point startPoint = new Point(10, 20);
Rectangle rect = new Rectangle(startPoint, new Size(100, 50));
```

10. Класс Bitmap.

Описание : Класс, представляющий растровое изображение. Используется для загрузки, сохранения и манипулирования изображениями.

Пример использования :

```
Bitmap myBitmap = new Bitmap("image.png");
Graphics g = Graphics.FromImage(myBitmap);
g.DrawRectangle(new Pen(Color.Red), new Rectangle(10, 10, 50, 50));
myBitmap.Save("modified_image.png");
```

11. Класс PaintEventArgs.

Описание : Класс, передаваемый в обработчик события Paint. Содержит объект Graphics, который можно использовать для рисования.

Пример использования :

```
private void Form1_Paint(object sender, PaintEventArgs e)
{
    Graphics g = e.Graphics;
    g.DrawLine(new Pen(Color.Black), new Point(0, 0), new Point(100, 100));
}
```