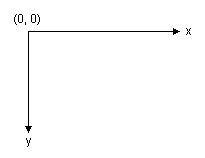
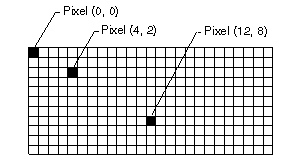
# Урок 1. Графические методы

## Общие сведения о векторной графике

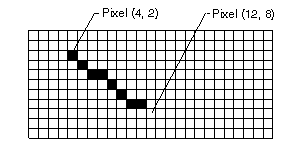
Рисование линий, прямоугольников и других фигур с использованием интерфейса GDI+ происходит в некоторой системе координат. Пользователь может выбрать одну из многих реализованных систем координат, но по умолчанию используется плоская декартова система координат, начало координат которой расположено в верхнем левом углу экрана, ось X направлена вправо, а ось Y — вниз. Единицей измерения в заданной по умолчанию системе координат является пиксель (минимальный элемент изображения).



Изображение на мониторе компьютера формируется как прямоугольный массив точек (пикселей), являющихся минимальными элементами изображения. Количество пикселей, отображаемых на экране, зависит от типа монитора. Количество пикселей, отображаемых на экране конкретного монитора, может быть изменено пользователем в определенных пределах.



При использовании интерфейса GDI+ для рисования линии, прямоугольника или формы необходимо указывать определенные ключевые данные, задающие параметры рисуемого элемент. Например, отрезок задается координатами двух точек, а прямоугольник — точкой, шириной и высотой. Интерфейс GDI+ взаимодействует с программным обеспечением драйвера экрана, чтобы определить, какие пиксели экрана должны быть высвечены, чтобы на экране возникло изображение линии, прямоугольника или кривой. На приведенном ниже рисунке показаны пиксели, высвечиваемые, чтобы отобразить отрезок прямой линии от точки с координатами (4, 2) до точки с координатами (12, 8).



## Графические методы

*Графические методы* – это функции, которые содержатся в языке *C#* и вызываются во время выполнения программы.

С помощью графических методов можно рисовать на форме, на кнопке, на текстовом поле, на графическом поле *PictureBox* или на других элементах управления. Для обеспечения рисования в *C#* имеется специальный класс *Graphics*, расположенный в пространстве имен *System.Drawing*. Его код и содержит программы для рисования фигур и для другой работы с графической информацией. Среди них много методов рисования фигур (например, все те, что начинаются на *Draw* и *Fill*).

<http://msdn.microsoft.com/ru-ru/library/system.drawing.graphics.aspx>

Перед тем как рисовать линии и фигуры и управлять ими необходимо создать объект *Graphics*. Объект – это «копия», экземпляр класса, обладающий всеми нужными его возможностями. Объект *Graphics* представляет поверхность рисования и используется для создания графических изображений.

Класс *Graphics* предоставляет методы для вывода объектов в устройстве отображения. Объект *Graphics* связан с конкретным элементом управления.

Ниже представлены два этапа работы с графикой.

1. Создание объекта *Graphics*.
2. Использование объекта *Graphics* для рисования.

## Создание объекта Graphics

<http://msdn.microsoft.com/ru-ru/library/5y289054.aspx>

Существуют различные способы создания графических объектов. Один из них:

Вызвать метод *CreateGraphics* элемента управления или формы, чтобы получить ссылку на объект *Graphics*, соответствующий поверхности рисования этой формы или элемента управления. Этот подход используется, если необходимо рисовать на поверхности уже существующей формы или элемента управления.

Существует еще такое требование: форме и каждому элементу управления – свой объект класса *Graphics*. Например, для рисования на графическом поле *PictureBox*, на форме и метке необходимо создать 3 объекта класса *Graphics*:

Graphics g = pictureBox1.CreateGraphics();

Graphics g2 = this.CreateGraphics();

Graphics g3 = label1.CreateGraphics();

## Некоторые методы класса Graphics

* *Clear* – Очищает всю поверхность рисования и выполняет заливку поверхности указанным цветом фона.

Используя объект *Graphics*, можно нарисовать много разных ***фигур и линий***. Мы рассмотрим только некоторые из них:

* *DrawLine* – Проводит линию, соединяющую две точки,
* *DrawRectangle* – Рисует прямоугольник,
* *DrawEllipse* – Рисует эллипс, определяемый ограничивающим прямоугольником,
* *FillRectangle* – Заполняет внутреннюю часть прямоугольника
* *FillEllipse* – Заполняет внутреннюю часть эллипса,

Для любого объекта, потребляющего системные ресурсы (например, для объектов *Pen* и *Graphics*), всегда нужно вызывать метод *Dispose*.

## Рисование фигур и изображений и управление ними

### Очистка изображения

Прежде чем создавать новое изображение, часто бывает необходимо вытереть старое. Для этого используется метод *Clear*. Очищает всю поверхность рисования и выполняет заливку поверхности указанным цветом фона.

*Синтаксис метода:*

*object*.Clear(Color.*Цвет*);

*Пример:*

g.Clear(Color.White);

### Отрезки прямых

<http://msdn.microsoft.com/ru-ru/library/xckcwxsa.aspx>

Рисуются методом *DrawLine*. Отрезок прямой можно построить, если известно положение двух его крайних точек. Они задаются в обращении к методу. Первая пара параметров вслед за указанием пера – координаты одной точки (любой из двух), вторая пара – другой. Первое число в каждой паре - горизонтальная координата, второе число – вертикальная.

*Синтаксис метода:*

*object*.DrawLine (Pens.*Цвет*, x1, y1, x2, y2);

*Пример прорисовки зеленой линии из левого верхнего угла объекта PictureBox в правый нижний:*

Graphics g = pictureBox1.CreateGraphics();

int x1 = 0;

int y1 = 0;

int x2 = pictureBox1.Width;

int y2 = pictureBox1.Height;

g.DrawLine(Pens.Green, 0, 0, x2, y2);

*Пример прорисовки зеленой линии из левого верхнего угла формы в правый нижний:*

Graphics g = this.CreateGraphics();

int x1 = 0;

int y1 = 0;

int x2 = this.ClientSize.Width;

int y2 = this.ClientSize.Height;

g.DrawLine(Pens.Green, x1, y1, x2, y2);

### Прямоугольники и квадраты

<http://msdn.microsoft.com/ru-ru/library/x6hb4eba.aspx>

<http://msdn.microsoft.com/ru-ru/library/c6ksfcek.aspx>

Рисуются методами *DrawRectangle* (контур) и *FillRectangle* (заливка).

Прямоугольник можно построить, если известно положение его верхнего левого угла, ширина и высота. Четыре параметра в скобках вслед за указанием пера (или кисти) и определяют эти величины.

*Синтаксис метода:*

*object*.FillRectangle(Brushes.*Цвет*, x, y, width, height);

*object*.DrawRectangle(Pens.*Цвет*, x, y, width, height);

*Пример прорисовки голубого квадрата с синим контуром в центре объекта PictureBox:*

Graphics g = pictureBox1.CreateGraphics();

int x = pictureBox1.Width / 2;

int y = pictureBox1.Height / 2;

int w = 100;

int h = 200;

g.FillRectangle(Brushes.SkyBlue, x - w / 2, y - h / 2, w, h);

g.DrawRectangle(Pens.Blue, x - w / 2, y - h / 2, w, h);

### Эллипсы и круги

Рисуются методами *DrawEllipse* (контур) и *FillEllipse* (заливка).

Они рисует эллипс, определенный ограничивающим прямоугольником, заданным с помощью пары координат, ширины и высоты.



*Синтаксис метода:*

*object*.FillEllipse (Brushes.*Цвет*, x, y, width, height);

*object*.DrawEllipse (Pens.*Цвет*, x, y, width, height);

*Пример прорисовки голубого круга с синим контуром, касающегося либо верхнего и нижнего краев формы, либо правого и левого:*

int w, h, d, x, y;

Graphics g = pictureBox1.CreateGraphics();

w = pictureBox1.Width;

h = pictureBox1.Height;

d = Math.Min(w, h);

x = w / 2;

y = h / 2;

g.FillEllipse(Brushes.Aqua, x - d / 2, y - d / 2, d, d);

### Координаты

Координаты в методах могут быть и отрицательными. В результате вся фигура или ее часть оказывается левей или выше формы и поэтому не видна. Координаты в методах могут быть и слишком большими положительными. В результате вся фигура или ее часть оказывается правей или ниже формы и поэтому не видна.

### Освобождение ресурсов системы, используемых объектом

После завершения использования объектов, которые потребляют ресурсы системы, например объектов *Graphics* или *Pen*, необходимо вызвать для этих объектов метод *Dispose*.

Каждый созданный объект, даже если он невидим, расходует ресурсы компьютера. Хорошим тоном у программистов является заботиться о компьютере и не перенапрягать его. Поэтому, когда ясно, что объект отработал свое и больше не понадобится, метод *Dispose* освобождает все ресурсы, используемые данным объектом *Graphics*.

g.Dispose();

drawPen.Dispose();