# Создание сложных графических объектов

## Путь (траектория, контур)

<http://msdn.microsoft.com/ru-ru/library/system.drawing.drawing2d.graphicspath%28v=vs.100%29.aspx>

<http://msdn.microsoft.com/ru-ru/library/b5hek5ky%28v=vs.100%29.aspx>

Класс *GraphicsPath* пространства имен *System.Drawing.Drawing2D* позволяет создавать контур, состоящий из последовательности кривых и геометрических фигур, которые обрабатываются и отображаются как один объект.

В этом классе имеются шесть конструкторов, которые создают либо пустой контур, либо контур, построенный по множеству вершин и по типам, ассоциированным с ними.

Контур состоит из последовательности прямолинейных и криволинейных отрезков. Путь может быть как замкнутым, так и незамкнутым. Его можно замкнуть автоматически, применив метод *CloseFigure*. В этом случае начальная и конечная точки фигуры соединяются прямой линией. Новый путь неявно начинается при его инициализации или когда предыдущий контур был замкнут. Явно новый путь можно начать методом *StartFigure*.

В классе *GraphicsPath* имеется ряд методов, позволяющих динамически модифицировать экземпляр этого класса. Очередной элемент добавляется в путь одним из методов:

|  |  |  |  |
| --- | --- | --- | --- |
| AddArc | AddCurve | AddPath | AddRectangles |
| AddBezier | AddEllipse | AddPie | AddString |
| AddBeziers | AddLine | AddPolygon |  |
| AddClosedCurve | AddLines | AddRectangle |  |

Некоторые другие методы класса *GraphicsPath*:

|  |  |
| --- | --- |
| *Метод* | *Описание* |
| *Clone* | Создает точную копию объекта |
| *Flatten* | Преобразует каждую кривую в данном контуре в последовательность соединенных отрезков прямых. |
| *GetBounds* | Возвращает прямоугольник, ограничивающий этот объект *GraphicsPath*. |
| *GetLastPoint* | Получает последнюю точку массива *PathPoints* для этого объекта *GraphicsPath*. |
| *IsVisible* | Указывает, содержится ли указанная точка внутри этого объекта *GraphicsPath*. |
| *IsOutlineVisible* | Указывает, содержится ли определенная точка внутри контура этого объекта *GraphicsPath* при его отображении с помощью указанного объекта *Pen*. |
| *Reverse* | Изменяет порядок точек в массиве *PathPoints* этого объекта *GraphicsPath* на противоположный. |
| *Transform* | Применяет матрицу преобразования к этому объекту *GraphicsPath*. |
| *Warp* | Применяет преобразование перекоса к этому объекту *GraphicsPath*. |
| *Widen* | Добавляет дополнительное очертание к контуру. |
| *StartFigure* | Открывает новую фигуру, не замыкая при этом текущую фигуру. Все последующие точки, добавляемые к контуру, добавляются к этой новой фигуре. |

Контур можно разделить на отдельные фигуры, которые являются либо незамкнутыми, либо замкнутыми. Фигура может содержать несколько примитивов.

Рисует созданный путь в контексте графического устройства метод *DrawPath*.

Заполненный путь создает метод *FillPath*.

*Пример 1. Создается контур, состоящий из одной фигуры. Фигура представляет собой одну дугу. Дуга имеет угол развертки, равный 180 градусам, в принятой по умолчанию системе координат он соответствует движению против часовой стрелки.*

GraphicsPath path =new GraphicsPath();

path.AddArc(175, 50, 50, 50, 0, -180);

gr.DrawPath(new Pen(Color.Blue,4), path);

*Пример 2. Создается контур, состоящий из двух фигур. Первая фигура представляет собой дугу, за которой следует линия. Вторая фигура является прямой, за которой следуют кривая и другая прямая. Первая фигура незамкнутая, а вторая — замкнутая*

// Создаем массив точек для кривой во второй фигуре

Point[] points ={new Point(40,60), new Point(50,70), new Point(30,90)};

GraphicsPath path =new GraphicsPath();

path.StartFigure() ;// Начинаем первую фигуру.

path.AddArc(175, 50, 50, 50, 0, -180);

path.AddLine(100, 0, 250, 20);

/// Первая фигура не замкнута.

pathStartFigure(); // Начинаем вторую фигуру.

pathAddLine(50, 20, 5, 90);

pathAddCurve(points,3);

pathAddLine(50, 150, 150, 180);

pathCloseFigure(); // Вторая фигура замкнута.

gr.DrawPath(new Pen(Color.Green, 2), path);

*Задача 1. Создать контур, состоящий из прямоугольника и двух эллипсов.*

Graphics g = pictureBox1.CreateGraphics();

GraphicsPath myPath =new GraphicsPath();

Point p =new Point(100, 50);

Size s =new Size(100, 200);

Rectangle Rect =new Rectangle(p, s);

Rectangle Rect2 =new Rectangle(250, 150, 200, 100);

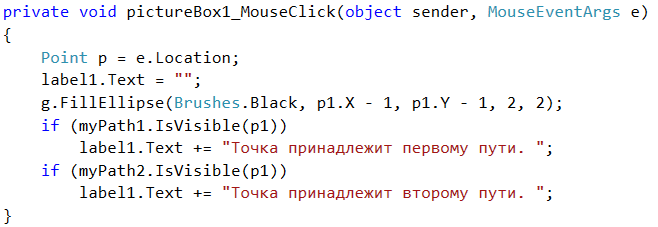
myPath.AddRectangle(Rect);

myPath.AddEllipse(Rect2);

myPath.AddEllipse(110, 120, 200, 100);

g.DrawPath(Pens.Blue, myPath);

*Задача 2. Даны две области. Определить, принадлежит ли точка, в которой кликнули мышкой, какой-то из этих областей.*



## Класс Region

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

<http://msdn.microsoft.com/ru-ru/library/4t53hf8d.aspx>

Класс *Region* позволяет создавать области, над которыми допустимо совершать операции объединения и пересечения, которые можно закрашивать, используя специфицированную кисть, и трансформировать, применяя аффинные преобразования координат.

Область – часть зоны, отображаемой устройством вывода. Области могут быть как простыми, так и сложными (набор многоугольников и замкнутых фигур). Они могут принимать любую форму.

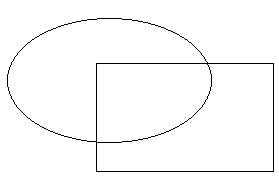
Области часто используются для задания отсечения или для проверки попадания в некоторую зону. Отсечение заключается в запрете на рисование вне границ области экрана, обычно это область, нуждающаяся в обновлении.

В классе *Region* имеются пять конструкторов, позволяющих инициализировать объект *Region*, создавать его на базе прямоугольника, пути либо уже существующего объекта *Region*. В последнем случае в качестве значения параметра конструктора используется объект *RegionData*, представляющий информацию о существующем объекте *Region*.

Можно также создавать сложные области путем совмещения существующих областей. У класса *Region* имеются следующие методы для объединения областей: *Intersect*, *Union*, *Xor*, *Exclude* и *Complement*.

|  |  |
| --- | --- |
| *Метод* | *Описание* |
| *Intersect* | *Пересечение* (набор всех точек, принадлежащих обеим областям).  Обновляет объект, оставляя в нем только общие точки с указанным объектом |
| *Union* | *Объединение* (набор всех точек, принадлежащих одной или обеим областям).  Обновляет объект, объединяя точки базового и указанного объектов |
| *Xor* | *Исключающее объединение* (набор всех точек, принадлежащих только одной из областей). Обновляет объект, оставляя в нем только точки из объединения первого и второго объектов, без из общих точек |
| *Exclude* | *Исключение* (набор всех точек первой области, не являющихся точками второй области). Обновляет объект, исключая из него общие точки с указанным объектом |
| *Complement* | *Дополнение* (набор всех точек объединения, не входящих в первую область).  Обновляет объект вторым объектом, исключая из него все точки первого объекта. |

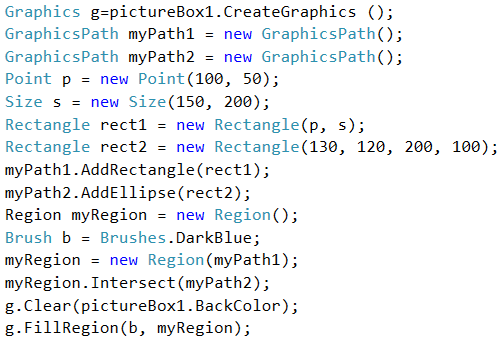
Например, даны две области: первая – эллипс, вторая – прямоугольник.



Тогда результаты объединения этих областей будут следующими:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *Intersect* | *Union* | *Xor* | *Exclude* | *Complement* |
|  |  |  |  |  |

Пример пересечения двух областей:



Некоторые другие методы класса *Region*:

|  |  |
| --- | --- |
| *Метод* | *Описание* |
| *Clone* | Создает точную копию объекта |
| *GetBounds* | Возвращает прямоугольник, ограничивающий данный объект |
| *IsEmpty* | Проверяет, является ли объект пустым |
| *Islnfinite* | Проверяет, является ли объект неограниченным в контексте данного графического устройства |
| *IsVisibie* | Проверяет, принадлежит ли указанная точка или прямоугольник данному объекту |
| *MakeEmpty* | Устанавливает содержание объекта пустым |
| *Transform* | Трансформирует объект, используя указанную матрицу преобразований |
| *Translate* | Переносит начало координат |

В классе *Region*, как и в классе *GraphicsPath,* имеется метод *IsVisible*, который позволяет проверить, принадлежит ли точка данной области.