ИНФОРМАТИКА И ПРОГРАММИРОВАНИЕ

УДК 004.92

А.В. Приступа, А.В. Петрухин

УНИВЕРСАЛЬНЫЙ СПОСОБ ОТРИСОВКИ СПЕЦИАЛЬНЫХ ЛИНИЙ СРЕДСТВАМИ GDI+

Рассмотрены варианты применения графической библиотеки GDI+ в контексте рисования специальных типов линий широкого класса. Приведены конкретные примеры специальных линий, с которыми сталкиваются разработчики программных инструментов для построения схем дислокаций дорожных знаков, разметки и обустройства автомобильных дорог.

Ключевые слова: *компьютерная графика*, *GDI+*, *специальные типы линий.*

При создании графических приложений перед разработчиками очень часто встает задача отрисовки различных типов линий вдоль заданной траектории. При этом примитивы, формирующие шаблон специальной линии, могут быть сколь угодно различны. Рассмотрим некоторые типы специальных линий, с которыми приходится работать при проектировании схем дислокации дорожных знаков, разметки и обустройства автомобильных дорог.

1. Примеры специальных линий

1.1. Дорожная разметка

Для описания данной спецлинии необходимо выделить повторяющуюся часть (шаблон) и задать ее в некотором заранее разработанном формате. Для отрисовки рассмотренной линии шаблон следует повторять вдоль заданной траектории, при этом на кривых участках изображение должно соответствующим образом изгибаться вдоль траектории (рис. 1).

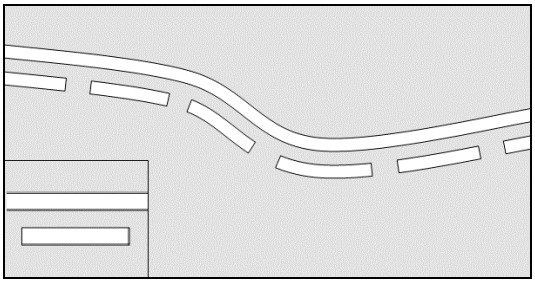

Рис. 1. Дорожная разметка

1.2. Дорожные ограждения

В данном примере помимо основного шаблона, задающего линию, можно выделить начальный и конечный шаблоны, которые имеют отличное от основного начертание (рис. 2). Дополнительное неудобство возникает и в случае, когда длина траектории, вдоль которой необходимо отрисовать спецлинию, не является кратной сумме длин начального, конечного и некоторого количества основных шаблонов (на практике так обычно и бывает).

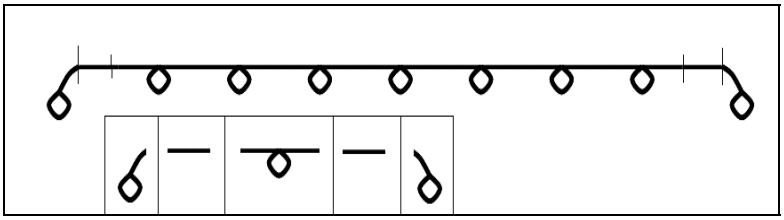

Рис. 2. Дорожное ограждение

Если предположить, что длина линии, вдоль которой необходимо отрисовать шаблоны, составляет 250 мм, а длины начального, основного и конечного шаблонов равны соответственно 10, 30 и 10 мм, то нетрудно подсчитать, что при отрисовке будут отображены 1 начальный, 7 основных и 1 конечный шаблоны (суммарной длиной 230 мм). При этом остается неопределенность в рисовании 20-мм линии, разрешить которую предлагается введением промежуточного шаблона. Он необходим в том случае, когда длина линии не соответствует сумме длин начального, конечного и N основных шаблонов. Например, для дорожных ограждений промежуточным шаблоном будет являться отрезок прямой линии, идущей по верху. Будем считать также, что промежуточный шаблон конечной (но достаточной) длины, а при отрисовке спецлинии будем выбирать лишь какую-то его часть в зависимости от того, насколько суммы длин не совпадают. Еще одним отличием от предыдущего варианта является и то, что здесь присутствуют примитивы, которые должны изгибаться вдоль траектории (верхняя линия), но есть и такие, которые должны отображаться без искажений (окружности).

1.3. Резюме

Таким образом, можно выделить ряд элементарных шаблонов, совокупность которых составляет требуемый составной шаблон. Рассмотрим их в порядке вхождения:

1) начальный шаблон;
2) промежуточный шаблон;
3) основной шаблон;
4) промежуточный шаблон;
5) конечный шаблон.

Каждый из элементарных шаблонов в свою очередь состоит из базовых примитивов (прямые, кривые Безье, сплайны), которые характеризуются набором точек (X, Y) в системе координат элементарного шаблона и толщиной. При этом для базовых примитивов заводится 2 списка: для изгибающихся и неизгибающихся примитивов. Для создания произвольного шаблона как совокупности элементарных разработана специальная утилита *SpecLinePatterns*.

2. Особенности реализации

2.1. Хранение шаблонов

Разработанные шаблоны хранятся в базе данных. Такой вариант хранения был выбран потому, что обычно в инструментах проектирования схем дислокации автомобильных дорог в основе лежит база данных с дорогами и элементами, входящими в нее. Если база данных отсутствует, то создавать ее только для хранения шаблонов спецлиний нецелесообразно, поэтому в качестве альтернативы структуру шаблонов можно хранить и в файлах. Если начинать с самого низкого уровня (примитивов), то каждый из них «умеет» записывать себя в поток и читать из него, а затем потоки двух списков примитивов записываются в поток элементарного шаблона.

2.2. Возможности GDI+ в контексте отображения специальных линий

Для решения задачи отображения разработанных шаблонов необходимо рассмотреть 2 возможности библиотеки GDI+ [1, 2], а именно возможность рисовать прерывистые линии произвольной конфигурации, задавая длины штрихов и пропусков вдоль пера (рис. 3), а также делать пропуски в исходном пере заданной толщины (рис. 4.А), задавая ширины видимых и невидимых частей пера поперек (рис. 4.Б).

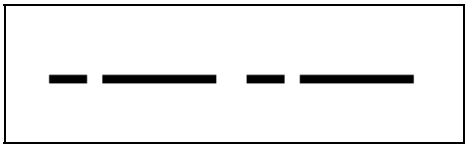

Рис. 3. Прерывистая линия

Рис. 4.А. Исходное перо

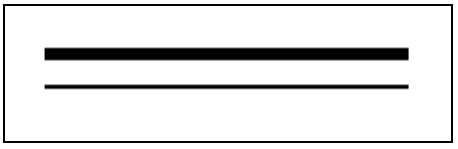

Рис. 4.Б. Перо с пропусками

Оперируя терминологией библиотеки GDI+ [3], рассмотрим объект *TGPPen* (перо для рисования). В простейшем случае конструктор пера выглядит как

TGPPen.Create (*MakeColor* (*A*, *R*, *G*, *B*), *Width*),

где 0<=*A*<=255 – степень прозрачности, *R*,*G*,*B* – значения трех базовых цветов, а *Width* – толщина пера. К объекту *TGPPen* можно применять различные методы, мы рассмотрим 2 из них:

TGPPen.SetDashPattern (*@dp, count*),

где *dp* – массив чередующихся значений вида «длина штриха – длина пропуска – длина штриха – ….», count – количество элементов массива.

TGPPen.SetCompoundArray (*@cp*, *count*),

где *cp* – массив чередующихся значений вида «толщина видимой части – толщина невидимой части – толщина видимой части – ….», *count* – количество элементов массива. В последнем случае массив строится по принципу накопления значений и для *i*-го элемента массива выполняется 0<=*cp*[*i*]<=1. Например, массив ср = [0,0, 0,3, 0,9, 1,0] задает перо, вид которого приведен на рис. 4.Б (0,3 – толщина первой видимой части, 0,6 – толщина скрытой части, 0,1 – толщина второй видимой части пера).

2.3. Базовые классы объектов

Рассмотрим основные классы, которые были разработаны для представления элементов шаблона и самого шаблона. Классом представления шаблона является следующий класс:

```
TROPattern = class (TObject)
public
    //длина и высота шаблона в мм
    property Width: Single read FWidth write FWidth;
    property Height: Single read FHeight write FHeight;
    //высота шаблона в пикс.
    property PixelHeight: Integer read FPixelHeight;
    //список изгибаемых примитивов
    property PenPrimitivesList: TObjectList read FPenPrimitivesList;
    //список неизгибаемых примитивов
    property NoTransformPrimitivesList: TObjectList read FNoTransformPrimi-
        tivesList;
    //процедура отрисовки шаблона
    procedure Draw(aGraphics: TGPGraphics);
    //функция заполнения массивов dp и cp
    function CalcDashValues: Boolean;
    //функция создания текущего пера
    function CreateNotNullPen(pen: TGPPen; var Index: Integer): TGPPen;
published
    constructor Create(w,h: Single);
    destructor Destroy; override;
end;
```

Базовым классом для примитивов, которые изгибаются вдоль траектории заданной линии, является класс

```
TROPenPrimitive = class (TObject)
public
    //массив вещественных точек, задающих траекторию примтива
    Points: TPointFDynArray;
    //шаблон, в который входит примитив
    property Owner: TROPattern read FOwner write FOwner;
    //толщина примитива
    property Width: Single read FWidth write FWidth;
```

```
    //абстрактная процедура отрисовки
    procedure Draw(aGraphic: TGPGraphics); virtual;
published
    constructor Create(aOwner: TROPattern; pts: TPointFDynArray; w: Single);
end;
```

В конкретных классах изгибаемых примитивов переопределяется только процедура отрисовки *Draw*. При этом цвет примитивов не является аргументом метода, вместо этого считается, что все изгибаемые примитивы рисуются основным цветом линии. Для неизгибаемых примитивов имеется возможность задания при отрисовке цвета пера и кисти (к примеру, если мы захотим нарисовать границы окружности дорожного ограждения другого цвета или даже закрасить их). Рассмотрим класс для неизгибаемых примитивов.

```
TRONoTransformPrimitive = class (TObject)
public
    //массив вещественных точек
    Points: TPointFDynArray;
    //шаблон, в который входит примитив
    property Owner: TROPattern read FOwner write FOwner;
    //толщина примитива
    property Width: Single read FWidth write FWidth;
    // точка привязки в системе координат шаблона
    property BasePoint: TGPPointF read FBasePoint write FBasePoint;
    //абстрактная процедура отрисовки
    procedure Draw(AGraphics: TGPGraphics; aPen: TGPPen; aBrush: TGPBrush;
                                        p: TGPPointF; Alpha: Single); virtual;
published
    constructor Create(aOwner: TROPattern; pts: TPointFDynArray; w: Single);
end;
```

Процедура *Draw* также перекрывается в подклассах конкретных примитивов.

Заметим, что кроме *pen* и *brush* метод *Draw* имеет в качестве аргументов также значения *p* и *Alpha*. Это значения текущей точки вывода очередного неизгибаемого примитива и значение текущего угла поворота, поскольку линия может быть ломаной или сплайном. И если в случае с изгибаемыми примитивами всё более или менее просто (средства библиотеки GDI+ помогают нам изгибать шаблон вдоль траектории, не интересуясь самой траекторией), то в случае с неизгибаемыми необходимо знать точку вывода и угол поворота.

2.4. Применение методов GDI+ для решения задачи

Для того чтобы успешно применить возможности библиотеки GDI+ для отрисовки произвольных спецлиний, предлагается сначала отрисовать все примитивы, входящие в состав шаблона, во вспомогательный битмап, после чего произвести разбор получившегося изображения. Для этого предназначен метод *CalcDashValues* класса *TROPattern*, представляющего шаблон. Опишем вкратце мезанизм его работы. На начальном шаге заводится двумерный динамический массив, количество строк которого соответствует высоте битмапа (пикс.), а количество столбцов является переменной величиной. В цикле проходим по всем строкам битмапа и на каждом шаге считаем количество пикселов <с изображением> – <без изображения> – <с изображением> и т.д., формируя структуру для задания в GDI+ пунктирного пера. Поскольку изображение шаблона часто бывает и белого цвета (на-

пример, линия дорожной разметки), предлагается сначала заполнить битмап цветом какого-либо редко используемого фона. Очевидно, что и сравнивать очередной пиксел на предмет отсутствия в нем полезного изображения следует тогда с цветом выбранного фона. Итак, будем сичтать, что данные для заполнения *dash*-массивов получены. Однако для разных строк эти массивы чаще всего различны.

Поэтому следующим важным этапом рисования спецлинии является создание текущего пера (метод *CreateNotNullPen*). В качестве аргументов он имеет на входе базовое перо для определения таких характеристик, как исходная толщина, цвет и т.п., а также индекс текущей строки (в качестве указателя на нужный *dash*-массив). В простейшем случае данная функция для каждой строки возвращает объект типа *TGPPen*, для которого заданы *dash*- (на основе рассчитанных данных *dash*-массива) и *compound*-характеристики. Элементы массива *CompoundArray* (рис. 4.Б) определяются тривиальным образом, особенно учитывая, что он состоит всегда из четырех элементов.

cp[*0*]:= *0,0*;
cp[*1*]:= *0,0*;
cp[*2*]:= *i* / *pattern.FPixelHeight*;
cp[*3*]:= (*i* + *1*) / *pattern.FpixelHeight*

Если рассматривать исходное (толстое) перо, толщина которого равна высоте шаблона в пикселах, то значение (*cp*[*1*] – *cp*[*0*]) задает толщину первой видимой части (считая сверху) – она равна нулю, значение (*cp*[*2*] – *cp*[*1*]) определяет размер «пропуска» (до текущей строки), а (*cp*[*3*] – [*cp2*]) – толщину текущей линии, которую мы и будем выводить (соответствует одной строке битмапа, в который мы отрисовали шаблон).

Таким образом, вместо отрисовки спецлинии одним пером для достижения нужного результата линия рисуется несколько раз разными перьями, но так, что ни одно не «затирает» другое. Алгоритм рисования спецлиний может быть и усовершенствован. Для этого создавать различные перья следует не для каждой строки битмапа, а только тогда, когда соседние строки перестают быть одинаковыми. Тогда можно сэкономить на времени отрисовки за счет незначительного усложнения логики. На рис. 5 приведен пример растра с изображением шаблона. Двигаясь сверху вниз, выделим непустые строки с разным изображением. Получим строки с номерами 5 – 9 и 13 – 17 включительно. Кроме этого, отличное от остальных (но одинаковое между собой) изображение будут иметь строки с номерами 10 – 12. Таким образом, для рисования данного шаблона потребуется 11 различных перьев, одно из которых (среднее) будет в 3 раза толще остальных.

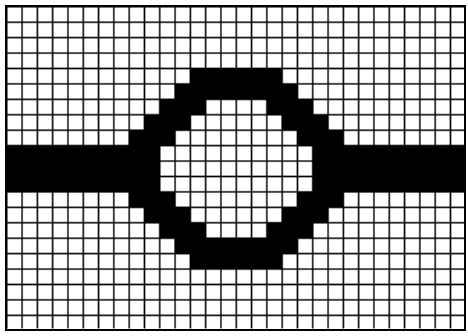

Рис. 5. Пример растрового изображения шаблона

Что касается отрисовки неизгибаемых примитивов, для которых рассмотренный метод не подходит, то процедуры их отрисовки относительно некоторой базовой точки реализованы в классах, которые их представляют. Вопрос состоит только в том, как определять базовые точки. Для первого отображения базовая точка известна (одно из свойств объекта «Неизгибаемый примитив»). Для определения всех остальных, зная длину шаблона, мы просто прибавляем ее к начальной, учитывая траекторию линии. В случае сплайнов или кривых Безье предлагается использовать для расчетов апроксимацию ломаными отрезками (визуально это незаметно).

Соответственно нам необходима вспомогательная процедура, которая бы по заданным точкам ломаной определяла бы положение базовых точек (закрашенные круги на рис. 6) независимо от траектории ломаной.

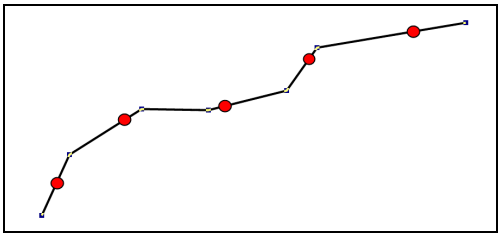

Рис. 6. Положение базовых точек

Таковой является процедура *Skip* (*Points*: *TPointFDynArray*; *L*: *Single*; *var X*: *Single*; *var Y*: *Single*; *var i*: *Integer*; *var Alpha*: *Single*), где *Points* – массив точек, задающих ломаную; *L* – длина шаблона; *X*, *Y* – координаты очередной базовой точки; *i* – последний рассмотренный (на текущем шаге) индекс точки в массиве *Points*; *Alpha* – значение текущего угла поворота, которое определяется направлением отрезка ломаной. Механизм работы процедуры состоит в проверке условия «умещается ли длина шаблона в текущий отрезок ломаной». Если результат положительный, то местоположение следующей точки определяется просто по формулам

$$X := X + L1 \times Cos(Alpha),$$

$$Y := Y - L1 \times Sin(Alpha),$$

иначе в цикле производим перебор отрезков ломаной до тех пор, пока не найдем нужный, изменяя при этом значения *i* и *Alpha*.

Заключение

Процедура отрисовки спецлинии *DrawSpecialLine* получает в качестве входных параметров объект *TGPGraphics* (созданный на основе контекста устройства вывода), объект *TGPPen* (перо для рисования), объект *TGPBrush* (кисть), 4 объекта-шаблона *TROPattern* (начальный, основной, конечный и промежуточный шаблоны) и массив точек, задающих ломаную, которая апроксимирует исходную траекторию с заданной точностью. Вначале отрисовываются все неизгибаемые примитивы, при этом попутно запоминаются координаты точек ломаной, которые соответствуют начальному шаблону, затем промежуточному, основному и т.д. Затем, зная координаты для вывода каждого шаблона (речь идет уже о примитивах, изгибаемых вдоль линии), в цикле для каждого шаблона рассчитываются значе-

ния массивов *Dash* и *Compound*, создается текущее перо (для рисования одной или нескольких строк растрового изображения шаблона) и производится собственно отрисовка изгибаемых составляющих шаблонов, повторяющих заданную траекторию. При этом начальный, конечный и промежуточный шаблоны рисуются однократно, а основной шаблон в общем случае многократно в зависимости от длины исходной кривой (ломаной).

Рассмотренный подход позволяет рисовать практически любые спецлинии, а рассмотренную логику вывода неизгибаемых примитивов можно расширить и для изображений (например, может понадобиться вдоль заданной траектории выводить пиктограммы).

ЛИТЕРАТУРА

1. *Брусенцев В.* GDI+: графика нового поколения [Электронный ресурс]. URL: http://www.rsdn.ru/article/gdi/gdiplus1.xml; http://www.rsdn.ru/article/gdi/gdiplus2.xml; http://www.rsdn.ru/article/gdi/gdiplus3.xml
2. *Поляков А.*, *Брусенцев В.* Программирование графики: CDI+ и DirectX. СПб.: БХВ-Петербург, 2005. 360 с.
3. *MSDN* → Win32 and COM Development → Graphics and Multimedia → GDI+.

Приступа Андрей Викторович
Томский государственный университет
E-mail: pristupa@sibmail.com
Петрухин Александр Васильевич
ООО «Индор-Диагностика» (г. Томск)
E-mail: diag@indor.ru

Поступила в редакцию 7 мая 2009 г.