УДК 004.424

РАЗРАБОТКА ВЕКТОРНОГО РЕДАКТОРА В СРЕДЕ DELPHI

Гаскаров А.Г.,
научный руководитель канд. физ.-мат. наук Дмитриев В.Л.
Стерлитамакский филиал Башкирского государственного университета

В настоящее время достаточно большое количество графических редакторов могут работать с двумя видами изображений: растровыми и векторными. Современная векторная графика – не просто геометрические фигуры с элементами текста. Векторные программы – это мощные инструменты с возможностью создания фотореалистических коллажей. Граница между векторной и растровой графикой постепенно исчезает. И то, что раньше было возможно только в растровых редакторах, теперь доступно пользователям таких пакетов, как Adobe Illustrator, Expression, Canvas, InkScape, Scribus, и др.

Многие современные графические редакторы для своего нормального функционирования предъявляют высокие требования к компьютерам, например, мощный процессор, большой объем оперативной памяти. Отчасти это связано с тем, что многие разработчики современного программного обеспечения намеренно забывают об оптимизации кода программ, рассчитывая в основном лишь на мощности современных систем. Такой подход оправдан, очевидно, тем, что он существенно сокращает время разработки программы за счет использования стандартных вычислительных алгоритмов. Но с точки зрения быстродействия, а в некоторых случаях, и качества, такой подход нецелесообразен. В результате оптимизации кода программы, предъявленные требования могут быть существенно уменьшены. Для этого надо знать принципы работы таких графических редакторов. С этой целью в представленной работе рассматривается возможность создания графического векторного редактора на языке визуального программирования Delphi, обладающего достаточным инструментарием для реализации наиболее распространенных задач.

Кроме того, в случаях, когда часто приходится рисовать какие-либо простейшие фигуры и различным образом менять их форму, использование программ, требующих значительных системных ресурсов, нецелесообразно; можно использовать ряд более простых продуктов, в том числе и программу, описанную в данной работе.

К числу необходимых функций, которыми должен обладать простейший графический редактор, относятся:

- построение простейших геометрических фигур и линий (прямоугольник, эллипс, прямая, ломаная, и другие);
- написание какого-либо текста;
- возможность изменять форму, цвет (контура и заливки), толщину контура, стиль заливки уже построенных фигур;
- возможность перемещать и копировать построенные фигуры;
- возможность сохранять результаты работы в виде отдельного файла в нескольких форматах;
- возможность открыть и редактировать ранее сохраненную работу.

Для работы с прямоугольниками, эллипсами, прямыми, ломаными, текстом удобно описать новые классы объектов. Назовем их соответственно TRectangle, TEllipse, TLine, TBrokenLine, TText. Также необходимо создать соответствующие списки классов TListRectangle, TListEllipse, TListLine, TListBrokenLine, TListText. Кроме того, для работы с разнообразными стилями заливки и стилями линий описываются соответствующие новые типы. Так как часть свойств и методов одного класса может повторяться в другом классе, то необходимо использовать механизмы наследования.

Ниже для примера приведена часть кода, содержащая описание двух классов – TLine и TListLine (подробных описаний методов соответствующих классов не приводится, т.к. они достаточно громоздки и сложны).

```
TLine = class
   private
     Old_X,Old_Y: Integer;
     function Patch(xx,yy: Integer): Real;
     procedure PenBrushStyle(Sender: TImage);
     procedure SetLineProperty(TLS: T_LineStyle);
     procedure ViewCollection1(Sender: TImage); //свободная волна
     procedure ViewCollection2(Sender: TImage); //пилообразная линия
     procedure ViewCollection3(Sender: TImage); //винтовая линия
     procedure ViewCollection4(Sender: TImage); //спираль
     procedure ViewCollection5(Sender: TImage); //роза
     procedure ViewCollection6(Sender: TImage); //синусоида
   public
     x1,y1,x2,y2: Integer;
     Color: TColor;
     Style: T_LineStyle;
     Width: Byte;
     District: Word;
     MarkerWidth: Word;
     Marker_OK: Boolean;
     L_C1: L_Collection1;
     L_C2: L_Collection2;
     L_C3: L_Collection3;
     L_C4: L_Collection4;
     L_C5: L_Collection5;
     L_C6: L_Collection6;
     constructor Create(xx,yy: Integer); overload;
     procedure Move(Sender: TImage; xx,yy: Integer; Size: Byte);
     procedure RePaint(Sender: TImage; Col: Boolean);
     procedure Marker(Sender: TImage);
     function Resize(xx,yy: Integer): Byte;
     procedure Constr_Copy(VR_TR: TLine);
 end;

 TListLine = class
   private
     List: TList;
   public
     Count: Integer;
     SelectItem: Integer;    //определяет ближайшую фигуру при ее выделении
     Number_Select: Integer; //хранит номер выделенной фигуры
     constructor Create;
     procedure Add(xx,yy: Integer);
     function Path_Selected(xx,yy: Integer):Real;
 end;
```

Любая линия или фигура может быть скопирована и в дальнейшем вставлена на любом этапе работы с программой. Кроме того, реализована возможность расширенной вставки – в этом случае вставлять можно любую из ранее скопированных фигур пяти видов. Для реализации этой возможности описан следующий тип:

```
F_Clipboard = record {Хранит информацию о типах скопированных фигур}
      F_Clip_1: TRectangle;
      F_Clip_2: TEllipse;
      F_Clip_3: TLine;
      F_Clip_4: TBrokenLine;
      F_Clip_5: TText;
      F_Type: Byte;      //тип последней скопированной фигуры
      F_ExtType: array[1..5] of Boolean; //типы уже скопированных фигур
end; {F_Clipboard}
```

При отрисовке линий поддерживаются широкие возможности настройки их свойств, что зачастую приводит к тому, что линия становится похожей более на какую-либо фигуру (рис. 1). Для ломаной линии предусмотрены следующие возможности: добавление и удаление узлов, замыкание линии. Кроме того, ломаная линия может быть трех типов (простая ломаная, линия четвертого порядка, кривая Безье).

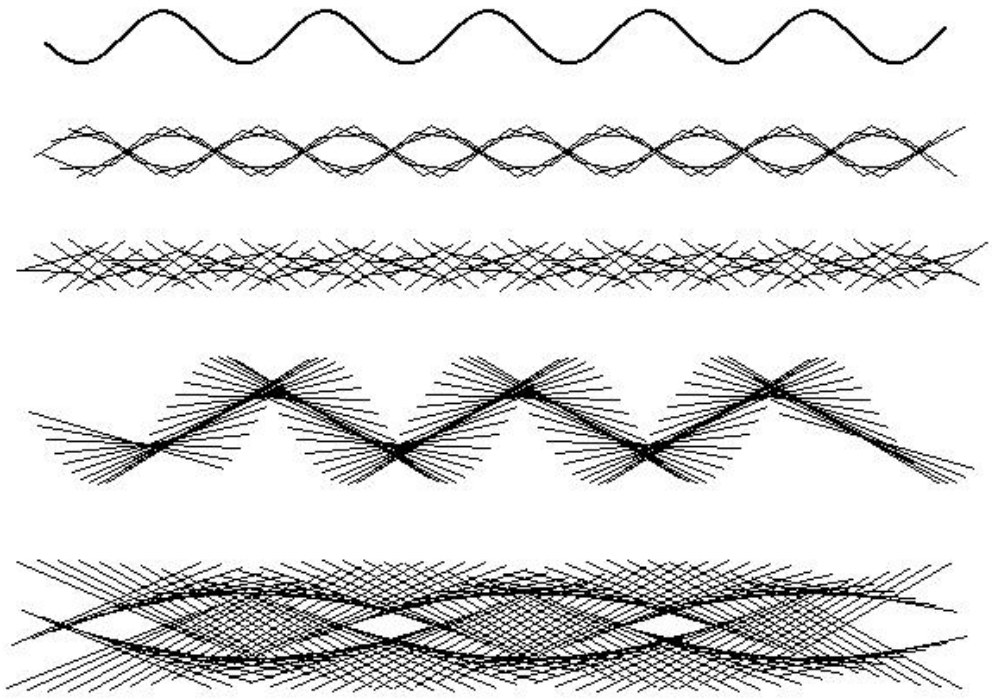

Рис. 1. Некоторые варианты линии в зависимости от настраиваемых параметров.

Для фигур предусмотрены три стиля заливки: заливка линиями, градиентная заливка, текстурное заполнение. В первых двух случаях предоставляются широкие возможности настройки свойств заливки. В третьем случае происходит заполнение выбранной текстурой (текстура хранится во внешнем файле), добавлена возможность пользователю самостоятельно добавить новые текстуры в программу.

При работе с текстом доступны настройки следующих его свойств: название шрифта, высота и ширина букв, угол поворота, цвет, стиль начертания (жирный, курсив, подчеркнутый, зачеркнутый). Над текстом могут быть проведены все те операции,

которые доступны другим объектам, в частности это: копирование, вставка, перемещение, выделение, порядок следования.

Результаты работы сохраняются в файл внутреннего формата программы (собственная разработка автора) с возможностью последующего редактирования. Поддерживается экспорт в наиболее популярные графические форматы BMP, JPEG (можно задать качество изображения); при этом можно сохранять изображение в оттенках серого.

Внешний вид окна программы с простейшим готовым изображением приведен на рис. 2. В левой части окна программы располагается панель инструментов, позволяющая работать с основными объектами редактора. Также здесь расположены кнопки экспорта, сохранения и открытия файлов. С правой стороны окна располагается панель свойств, позволяющая производить настройку параметров добавленных фигур. В нижней части окна программы располагается строка статуса, отображающая некоторую информацию относительно выбранных линий и фигур, а также показывающая текущие координаты мыши.

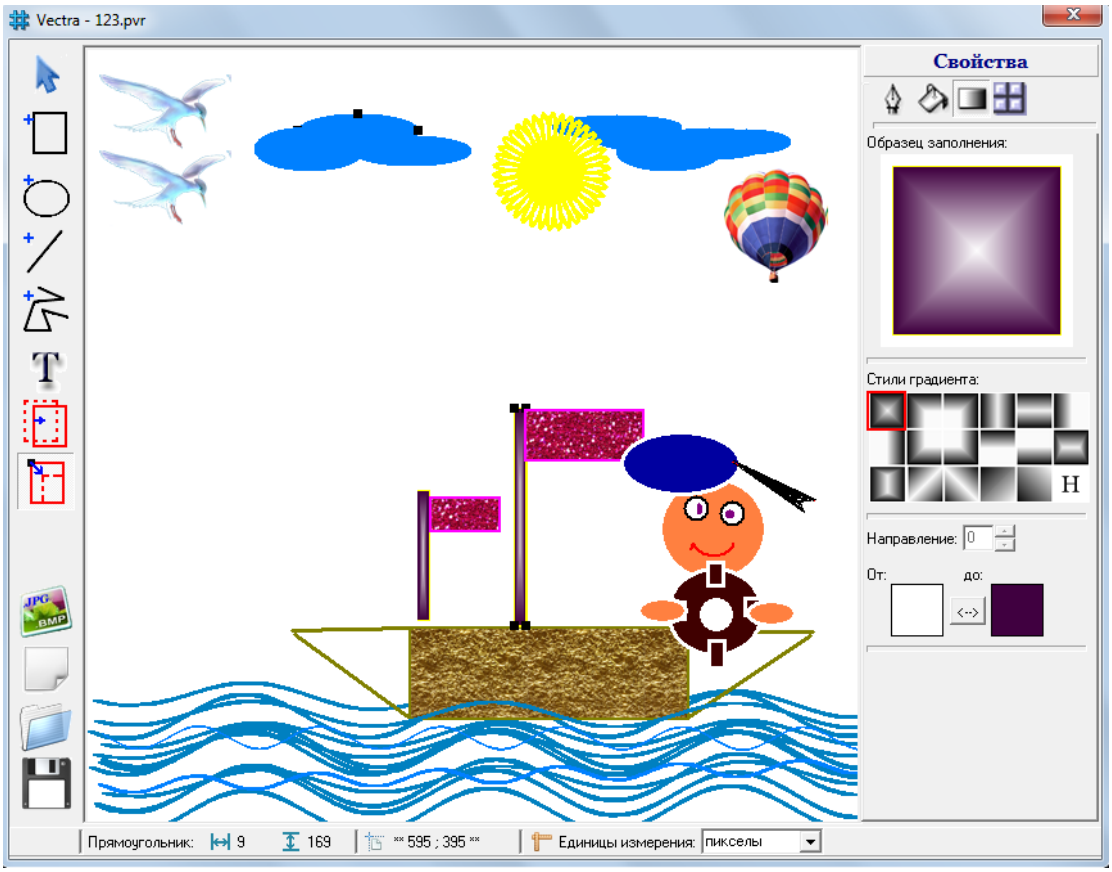

Рис. 2. Внешний вид окна векторного редактора с готовым изображением.

Алгоритмы и классы, разработанные в программе, также могут служить основой для разработки более сложных программных продуктов, а также использоваться при обучении программированию больших программных комплексов. Написанная программа относительно легко поддается модификации, что позволяет добавлять в нее новые возможности.