# The Development of a Vectorial Drawing Application using Advanced OOP Techniques

**Ing. Dan Andrei Stanciu**
**Universitatea "Tibiscus", Timişoara**

REZUMAT. Aplicația *Vectorial Draw* a fost dezvoltată în special pentru realizarea unor rapoarte care pot fi apoi tipărite. Ea permite desenarea şi gestionarea de primitive grafice într-un context grafic. Pentru aceasta au fost implementate prin intermediul programării orientate pe obiecte o serie de clase care sunt capabile să trateze la un nivel cât mai avansat aceste primitive grafice.

## 1 Introduction

The *Vectorial Drawing* application, as the name suggests, is capable of rendering basic shapes on a drawing surface. It is not a professional vectorial drawing application; it does not offer as many drawing possibilities for it was not designed for this purpose – its main goal is the ability to create, manage, process, load and save a list of simple bi-dimensional shapes adding up into a report that can be printed and/or used by another application.

However, the application offers various means of drawing and managing shapes in order to obtain an accurate design on the drawing board.

## 2 Structural details

Shapes

The kernel of any vectorial drawing application consists in shapes. This is also where the design and development of this application starts. The *Vectorial Draw* application can handle these types of basic shapes:

- free-hands (manually drawn shapes)
- lines
- rectangles and squares
- rounded-edges rectangles
- ellipses and circles
- grids
- texts

Besides the obvious action of drawing these shapes, the management of the shapes on the canvas implies they can be aligned, resized, equally distributed, scaled and Z-ordered.

Selection

In every item management application, every item that composes the whole can be selected. The *Vectorial Draw* application permits the selection of shapes in a similar way many graphical object designers do. There can be multiple shapes selected, and one of those shapes must be the *hot* shape. The *hot* shape is the shape that guides the management actions that involve multiple shapes selected, like aligning and resizing. The *cold* shapes will change their position/size according to the *hot* shape.

The application offers useful possibility of *restrictive selection*. This means the user will only be able to select a certain type of shapes at a time, be it by dragging and composing a selection area or by pointing and clicking.

Background

As the *Vectorial Draw* application was initially created as a helper for printable reports design and especially for reproducing already existent printed reports, the custom image background feature came in handy. Besides the standard paper-white background, the user can select a bitmap image that can be a scanned version of a printed report, scaled to size. This makes the graphical objects very easy to place accordingly to the real standard model of the report, improving by this the final product and also eliminating the need of margin rulers to precise place shapes.

## 3 OOP design and implementation

Advanced object-oriented algorithms can easily apply on shape management, because it is probably the most understandable abstract model used for this purpose. This application extends that simple model to advanced operations with shapes.

Here is a representation of the class diagram of this model:

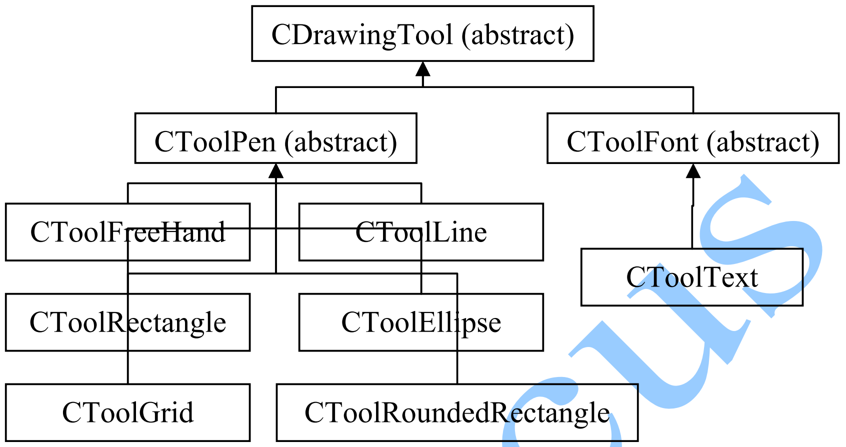


***Figure 1** The class diagram*

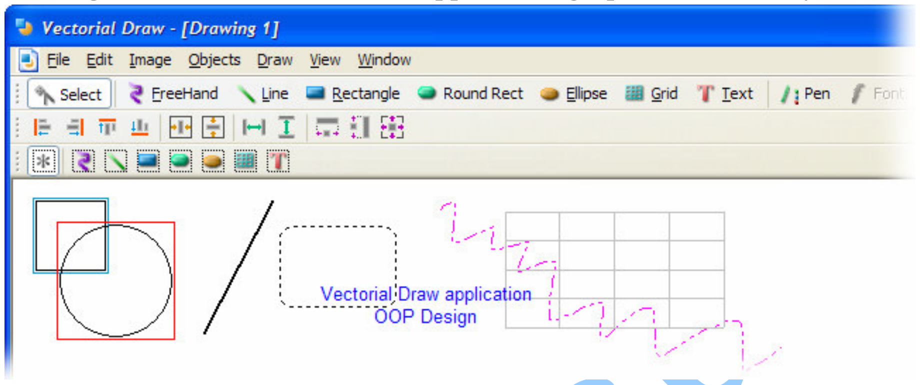
The base class, `CDrawingTool` is derived from the standard MFC class `CObject`. It is an abstract class because of several pure virtual functions that will be implemented in the corresponding subclasses. Some of these functions that have no implementation in the `CDrawingTool` class are:

```
virtual void DrawTool(CDC *pDC) = 0;
    // draws the shape on the device context

virtual CDrawingTool *Clone() = 0;
    // copies a shape (useful in copy-paste actions)

virtual CString GetToolCaption() const = 0;
    // retrieves the name of the tool
```

This class introduces as field members an `ID` for the object, the bounds of the object and also the object's name. It also offers the basic serialization function used for loading/storing shape objects from/to a stream.

The `CToolPen` and `CToolFont` classes are also abstract as they don't implement any of the abstract methods defined by `CDrawingTool`. They simply define some additional parameters common to each shapes class derived from them. Therefore, the `CToolPen` class introduces the color, style and width of the pen used to draw the shape, while the `CToolFont` class defines the properties of the font used to render a text object.

***Figure 2*** *The Vectorial Draw application graphical user interface*


Vectorial Draw - [Drawing 1]
File Edit Image Objects Draw View Window
Select FreeHand Line Rectangle Round Rect Ellipse Grid Text Pen Font
Vectorial Draw application
OOP Design

On the canvas and in the vectorial document (*.vd), the shapes are maintained into memory like this:

```
CTypedPtrList <CPtrList, CDrawingTool*> Items;
```

This defines a dynamic list of pointers to instances of the `CDrawingTool` class.

After the setting up the framework for the shapes by defining the base abstract shape tools classes, the actual definition and implementation of shape tools is as simple as it can be. For example, here is the implementation of the `CToolRectangle` class:

```
class CToolRectangle : public CToolPen
{
public:
    CToolRectangle() : CToolPen(TOOL_RECT) {}
    // default constructor
    CToolRectangle(CToolRectangle *aTool)
           : CToolPen(aTool) {}
    // copy constructor

    void DrawTool(CDC *pDC) {
           pDC->Rectangle(Bounds);
    }
    // effective drawing of the rectangle
    CString GetToolCaption() const {
          return "Rectangle";
    }
    // returns the caption of the tool

    virtual CDrawingTool *Clone() {
          return new CToolRectangle(this);
    }
    // copies of the object using the copy constructor
};
```

## Reference

[GHJV01] **Gamma, E., Helm, R., Johnson, R., Vlissides, J**. *Design Patterns – Elements of Reusable Object-Oriented Software,* Addison Wesley, 2001

[JK00] **Jamsa, K., Klander, L.** (2000), *Totul despre C şi C++ Manualul fundamental de programare în C şi C++*, Teora, 2000

[Sto00] **Stoicu–Tivadar, V**. (2000) *Programare orientată pe obiecte*, Orizonturi universitare, 2000

[Wil98] **Williams, M.** (1998) *Bazele Visual C++4*, Teora, 1998