Styles of visual note representation in an MML Application

Damien O'Neil

Introduction

Music Markup Language (MML) is a markup language that uses SVG to render sheet music on the screen. Now with a lot of writings and also in music there can be different way's in which a character or in the case of music a note can be depicted to the user. The character style can be changed by applying a font to a particular piece of text and the style of the text will be different depending on which font is applied. This can be said the same for music, a user may prefer to add a different style to the notes, according to his personal preference or even for more ease of reading.
This document will outline the uses of a grid in the MML language. It will show how the grid works and the basic language behind the grid.

The uses of creating a style in a grid

This MML grid was defined so that a user will be able to create a style for the individual notes. The main design of notes in this case will be around a grid format.
The user can specify how they would like the grid to look, i.e. the user can specify how many cells they would like in the grid, by specifying rows and columns. The user can also specify the size of the cells that are used throughout the grid.

```
<grid>
    <height>
        50
    </height>
    <width>
        75
    </width>
    <cols>
        7
    </cols>
    <rows>
        8
    </rows>
</grid>
```

Figure 1. Sample grid specified by the user
(A more in depth explanation will be provided on the elements later in the document.)

Once the grid has been specified to the user's requirements objects can then be added to the grid. Elements can then be added to particular cells by specifying different shapes, and the cells to where the elements are to be placed (Figure2.). This is a lot easier than specifying actual co-ordinates.

```
<rectangle>
    <position col = "5" row = "3"></position>
    <size cols = "1" rows = "1"></size>
    <angle>20</angle>
    <rounded x = "20" y = "20"></rounded>
</rectangle>
```

Figure 2. Sample Element used within the MML grid language
(A more in depth explanation will be provided on the elements later in the document.)

How the Grid Works with the Elements

Now what we have is a grid with an element, some shape at a particular place, at this point it means nothing to us so how can we make meaning of these elements.
This is done by grouping different elements together as a symbol. This symbol will then become the basis for a note that will be used within an application used to render MML. What we have now is a way of creating a new style for a note. But there are many different notes and glyphs within music, e.g. we have a quarter length note, a whole length note, a treble clef, a bass clef, and rests with different lengths and so on. Many of these different notes and glyphs can then be created within a single style by specifying many different "symbols" within one document.

```
<symbol id="quarter">
    <rectangle>
        <position col = "5" row = "3"></position>
        <size cols = "1" rows = "1"></size>
        <angle>20</angle>
        <rounded x = "20" y = "20"></rounded>
    </rectangle>
    <line>
        <strokewidth>
            3
        </strokewidth>
        <angle>
            20
        </angle>
        <length>
            2
        </length>
        <updown>
            down
        </updown>
    </line>
</symbol>
```

Rectangle Element

Line Element

Figure 3. Sample symbol made up of multiple elements.

We now have a grid and we have some elements which need to be placed on the grid. This can be done by specifying the position of a particular element. In the case of the rectangle in figures 2 and 3 a position element has been used. This will tell the application that this element is to be placed in a cell that the user specifies. The two attributes within the "position" opening tag specifies where exactly the element is to be placed. To allow the user to specify which cell to place the element two arguments are required which row and which column in the grid the element is to be placed. This is done be two separate attributes, firstly the "col" attribute which will specify the column in the grid and secondly the "row" attribute which will specify the row in the grid.
So what we have is an empty element with two attributes, which looks like this:

<position col="5" row="3"></position>

In the case of the example shown in figure 2 and 3, with the position been set to col="5" and row="3" and example outcome can be seen in figure 4.

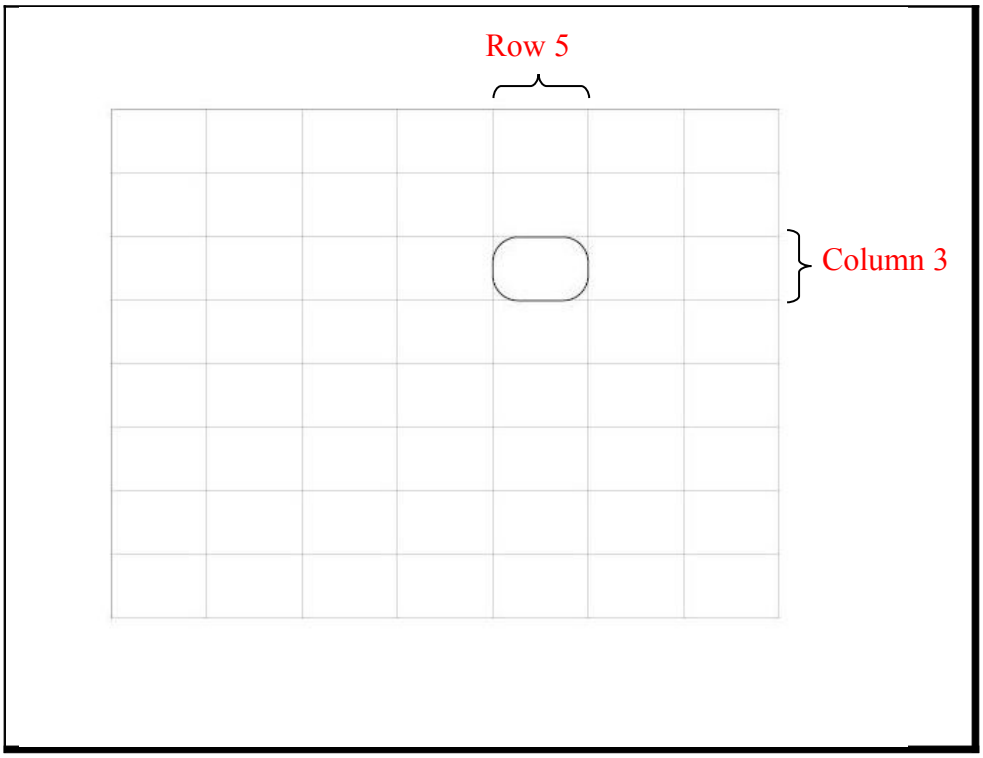

Figure 4. Outcome of position of row="3" and col="5"

Another use of the rows and columns within a grid is to specify the width and height of an element, in the case of figure 2 and 3 a "size" element was used again with two attributes, one to specify how many columns across the element should be ("rows") and one to specify how many rows in height the element should be ("cols").

The language Behind the Grid

Because MML was developed using XML it would feel wrong if anything else was used to create the styles, therefore XML was used to develop the grid also.
XML is a markup language that is the basis of most markup languages and allows a user to specify their own language. Like any markup language the grid style has a predefined set of elements and attributes. Most of the elements and attributes that are used will be the same for each element in a symbol, i.e. a rectangle will have certain elements and attributes that a line might have, but each of these elements also has their own unique elements and attributes that are used to uniquely define each element.
As mentioned before elements that are defined are essentially shapes that are added together to form a symbol. Shapes that can be used are: circle, rectangle, ellipse, line, polygon and polyline.

Syntax behind the language

When starting any XML document the first line must contain the XML version information, so in this case the very first line of code in our style will be:

<?xml version="1.0" encoding="iso-8859-1"?>

After that then the tags for the type of document is to be used, for example in the case of an html document the tags <html></html> are used, in this case we will use the tags:

<style></style>

Now we know what the document is all other elements for the document must be defined inside that opening and the closing tags of the style element.

The Grid

We must now define the grid. This is done by defining a grid element <grid></grid>. Certain aspects of the grid need to be defined; these will be defined by using different elements within the grid element.
The height of the cells need to be specified, this is specified within the element as a pixel size. This is specified by using a “height” element with the size inside of the opening and closing tags.
<height>50</height> will tell the system that the cells in the grid are to be 50 pixels high.
The width of the cells also need specifying, like the height element the size is in pixels and is specified within the opening and closing tags. The “width” element will be used for this.
<width>75</width> will tell the system that the cells in the grid are to be 75 pixels wide.
Now we know how big each cell is going to be, but a grid isn’t only made up of a single cell but a number of cells arranged in columns and rows. SO what we need to do is specify how many rows and how many columns are going to exist in our grid. This is done by using the elements “rows” and “cols” respectively. The number of rows or columns will be specified inside of the opening and closing tags for each of these:

<cols>7</cols> will tell the system that the grid will have 7 columns, and <rows>8</rows> will tell the system that the grid will have 8 rows.

What we have for the grid is a way to specify the height and width or a cell and a way to specify how many cells we have.
What we have at the moment is (figure 5):

```
<?xml version="1.0" encoding="iso-8859-1"?>
<style>
    <grid>
        <height>
            50
        </height>
        <width>
            75
        </width>
        <cols>
            7
        </cols>
        <rows>
            8
        </rows>
    </grid>
</style>
```

Figure 5. Grid syntax

When this is put through the system a grid is drawn so that the user can track their progress as they continue to add elements to their styles (figure 6).

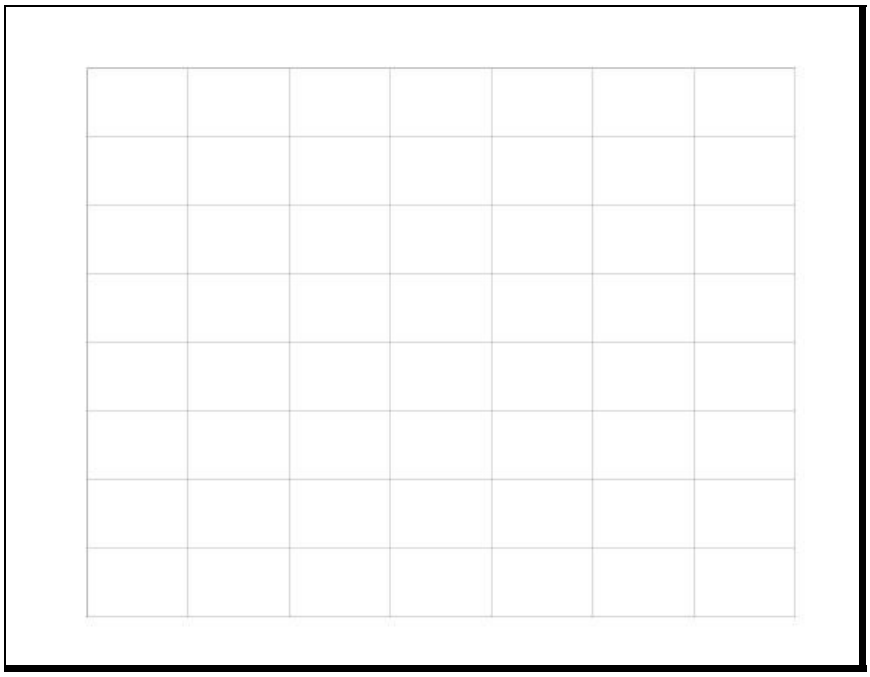

Figure 6. Outcome of the grid created in figure 5. Grid created with 8 rows and 7 columns.

The Elements

As was mentioned earlier the different elements that make up a note are made of different shapes. These shapes share some common attributes as well as having their own individual attributes that uniquely define them from each other.

Common Attributes

Because we are using a grid it would have been silly to have each element having their own individual attributes to define how the element is to be placed on the grid, therefore common attributes where used to define how the elements should be placed.

Common attributes include the position of the element, the size of the element, the rotation angle of the element, the fill colour, the line colour and an offset attribute.

- **Position attribute**
  As previously mentioned the position is defined by specifying the cell in which the element will lie, this is done by specifying the column and the row of the grid where the element is to lie.
  <position col="3" row="2"></position> will place the element on the grid at the 3rd column and the 2nd row. The outcome of a rectangle place in column 2 and row 3 can be viewed at figure 7.

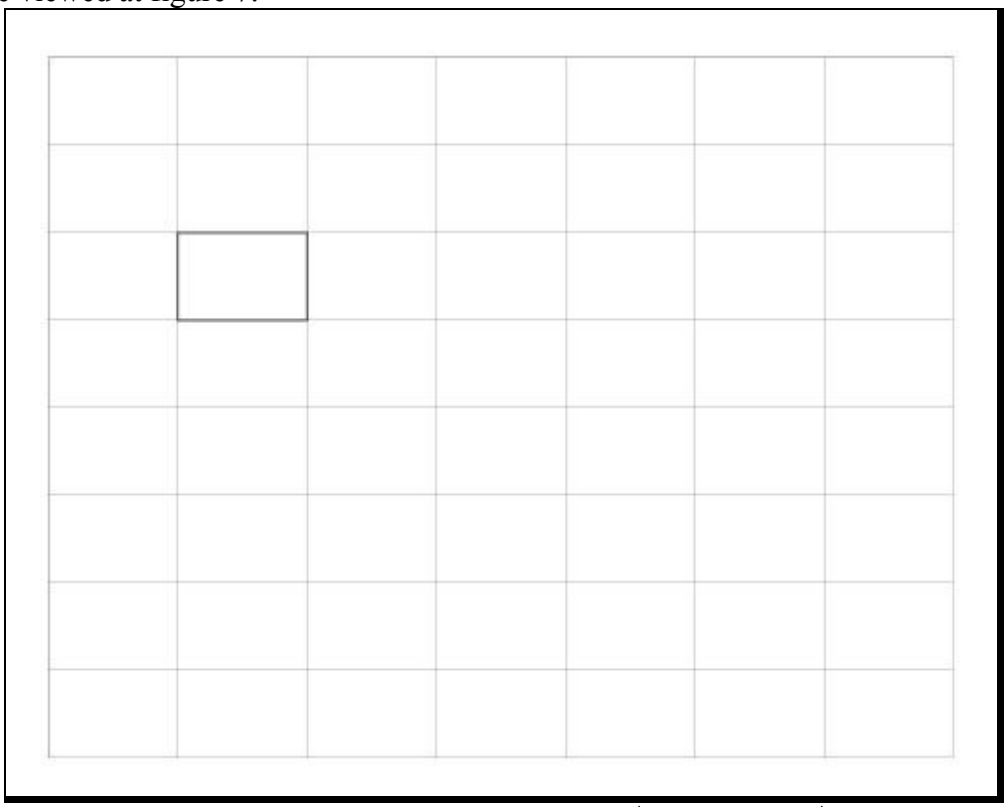

Figure 7. Grid with a rectangle place on the 2nd row and the 3rd column

- **Size attribute**

With certain elements such as rectangle, circle and ellipse, a size attribute can be used this will specify how many cells that the particular element will take up. By default the size of an element is set to 1. If an element that is two columns wide and three rows high is required then the following syntax would be used: <size cols=”2” rows=”3”></size> and this will tell the system to make the element to do that, In the case of a circle where the width and height is essentially the same, if the user specifies both the width and the height to be different then the circle element will become an ellipse, which will have a different height to the width and will no longer look like a circle.

- **Angle attribute**

  The angle attribute is used to specify how the element is to be rotated, by default there is no angle of rotation. If an element is to have an angle of 20 then the element will be rotated by 20°.
  The following syntax can by used for an angle to rotate an element: <angle>20</angle>, this will rotate the element by 20°. As shown in figure 8.

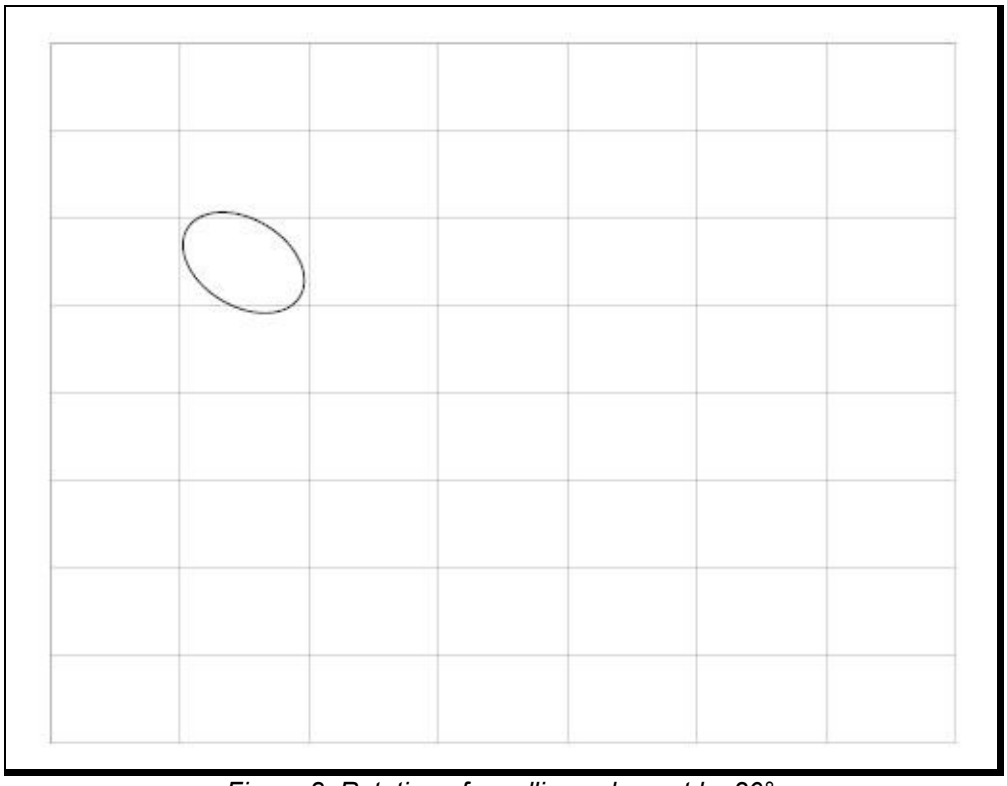

Figure 8. Rotation of an ellipse element by 20°.

- **Offset attribute**
  The offset attribute will place an element at a position away from the original cell by using specified x and y coordinates. If an offset has a positive x value greater than 0 then the element will be placed that number of pixels to the right of its original position. If it has a negative value it will place the element to the left of the original position.

If the offset y value is positive then the element is placed below its original position and a negative value will place the element above its original position.
Syntax for the offset is as follows: <offset x=”25” y=”50”></offset> this will place the element 25 pixels to the left of is original position and 50 pixels below its original position.

- **Fill attribute**
  The fill attribute specifies whether the area between its outline is to be filled and can specify a colour in which the area is to be filled. The syntax for the fill attribute is as follows: <fill color=”green”></fill> this will fill the area with a green colour.
  Example can be seen in figure 9.

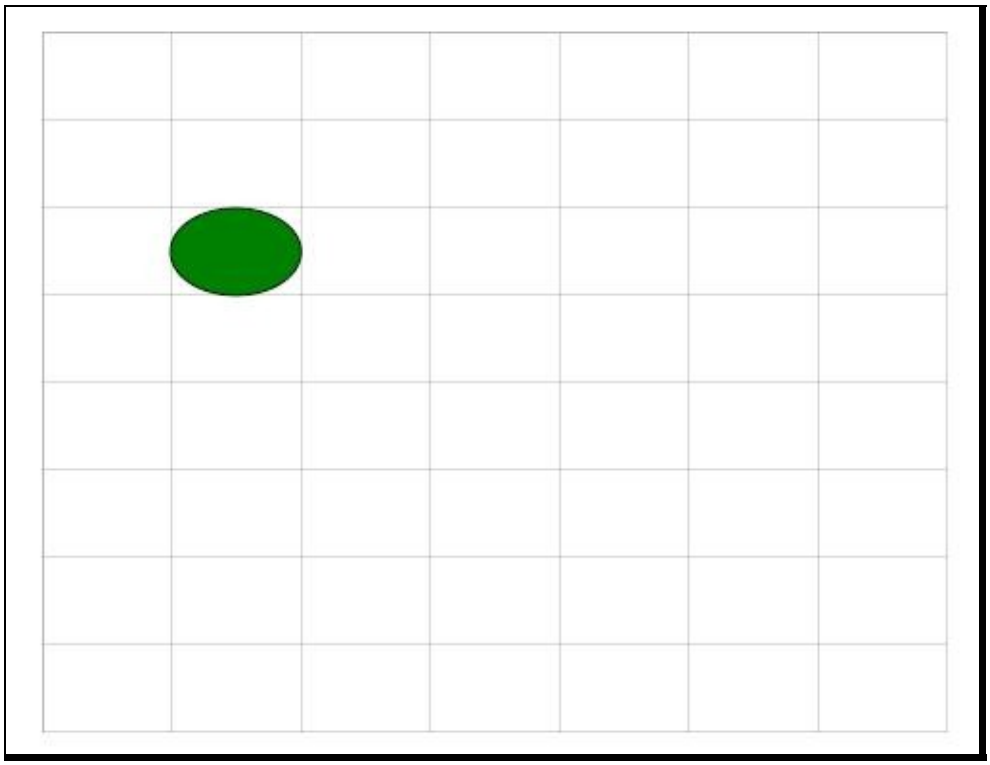

Figure 9. An element with a green fill

- **Stroke attribute**
  The stroke attribute is used to specify the colour of the outline of an element (figure 10). In the case of a polyline and a line this will determine the colour of those lines(figure 11).
  The syntax for this is as follows: <stroke color=”blue”></stroke>, this will cause the line or outline of an element to be blue.

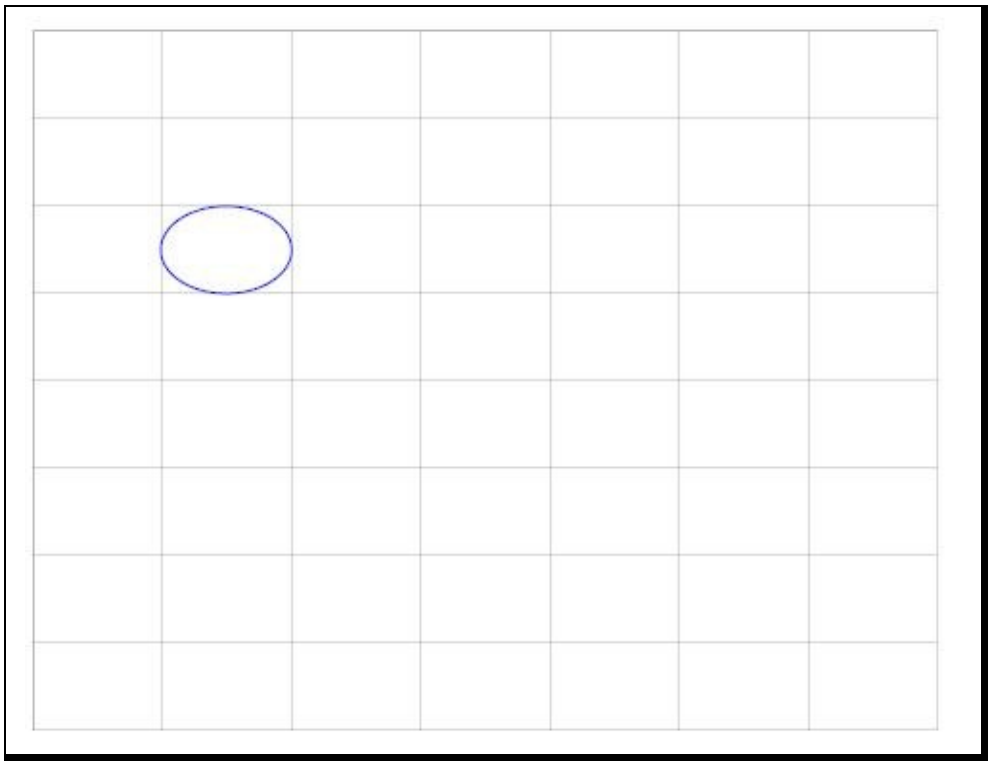

Figure 10. An element with a blue outline

Figure 11. A polyline that has a blue stroke.

Individual Attributes

Most elements have their own individual attributes that is used to differentiate from each other.

- **Circle Element**
  The circle element is firstly defined by using the “circle” tags, <circle></circle> and any other attributes that will be used to define the circle are placed inside the opening and closing circle tags(figure 12).

```
<symbol>
    <circle>
        <position col = "2" row = "3"></position>
    </circle>
</symbol>
```

Figure 12. The code to produce a circle in the 2nd column and the 3rd row of the grid

The circle element doesn’t have any individual attributes if it was to have any different attributes then it would become an ellipse so for this reason it just inherits only the common attributes (figure 13).

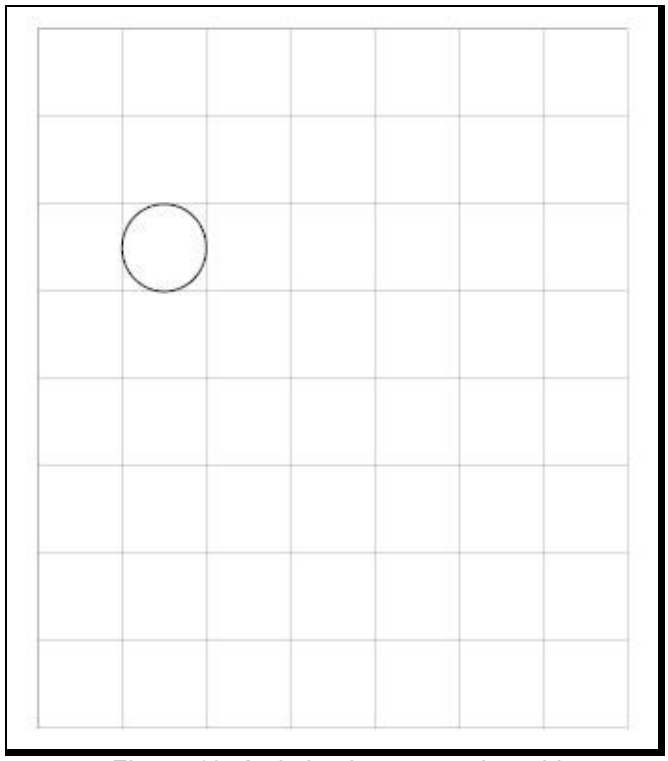

Figure 13. A circle element on the grid

- **Ellipses element**

An ellipse element if defined by using the "<ellipse>" opening and closing tags, and any of the defining attributes are to be placed within these opening and closing tags (figure 14).

```
<symbol>
    <ellipse>
        <position col = "2" row = "3"></position>
        <rounded x="20" y="20"></rounded>
    </ellipse>

</symbol>
```

Figure 14. The code to produce an ellipse in the 2nd column and the 3rd row of the grid

To differentiate from the circle element we've added a rounded attribute to this is used to specify how rounded the corners of the ellipses will be. Syntax is as follows: <rounded x="20" y="20"></rounded> this will define a rounded edge that is more defined if it hadn't been specified. An ellipse is also used in the place of a circle in the event that the grid cells do not have the same width and height.

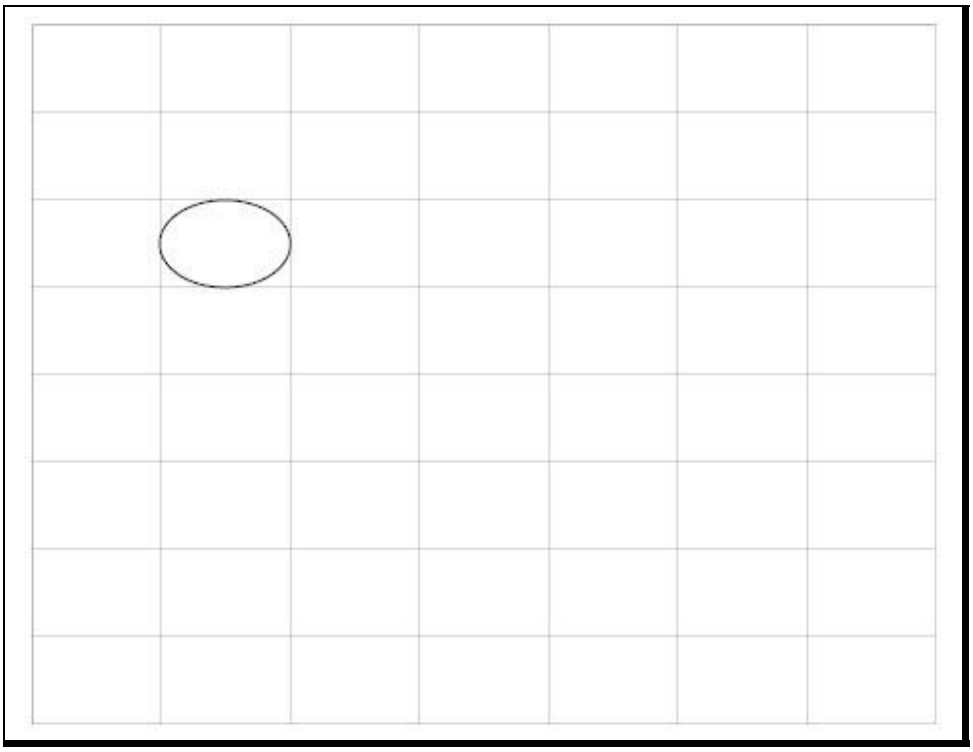

Figure 15. An ellipse element on the grid.

- **Rectangle Element**

A rectangle element is defined by using the “<rectangle>” opening and closing tags, with any defining attributes being placed within these tags (figure 16).

```
<symbol>
    <rectangle>
        <position col = "2" row = "3"></position>
        <angle>0</angle>
        <offset x="0" y="0"></offset>
        <rounded x="20" y="20"></rounded>
    </rectangle>
</symbol>
```

Figure 16. The code for a rectangle element in the 2nd column and the 3rd row of the grid

Like the ellipse element the rectangle may have rounded corners, this can be defined by using the rounded attribute with the syntax <rounded x=”20” y=”20”></rounded> as used in figure 16.
The code produced in figure will produce a rectangle with rounded corners as seen in figure 17.

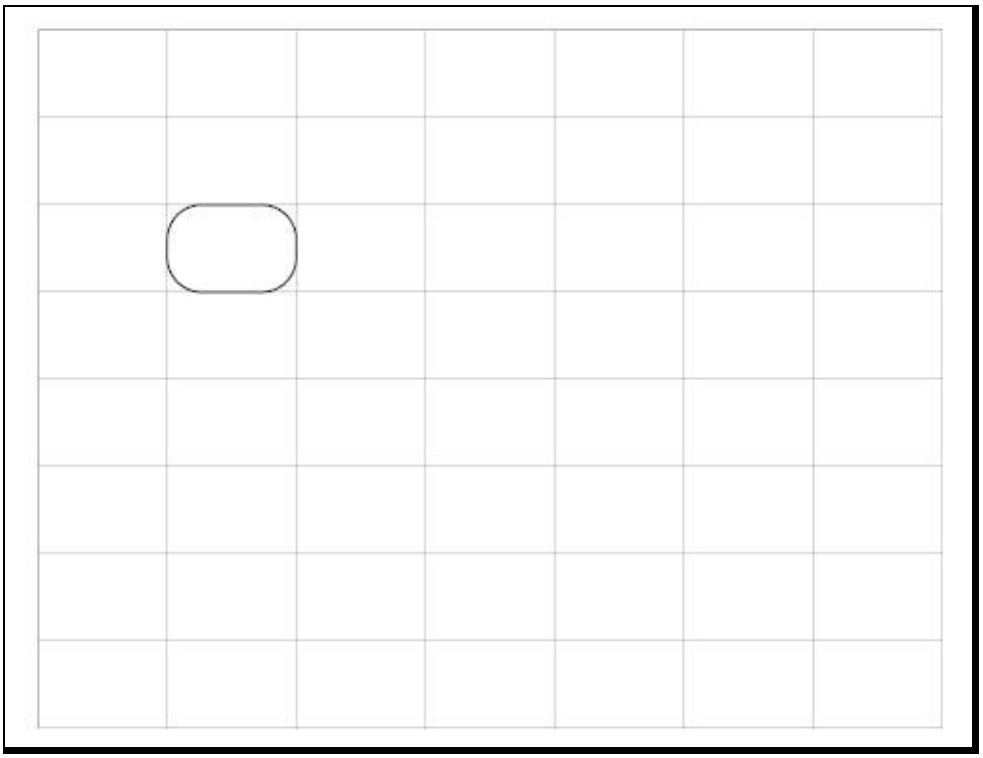

Figure 17. A rectangle with rounded corners.

If the rounded attributes are not used then a rectangle with corners will be displayed as in figure 7.

If the grid cells have the same width as the height then the rectangle will automatically become a square as shown in figure 18.

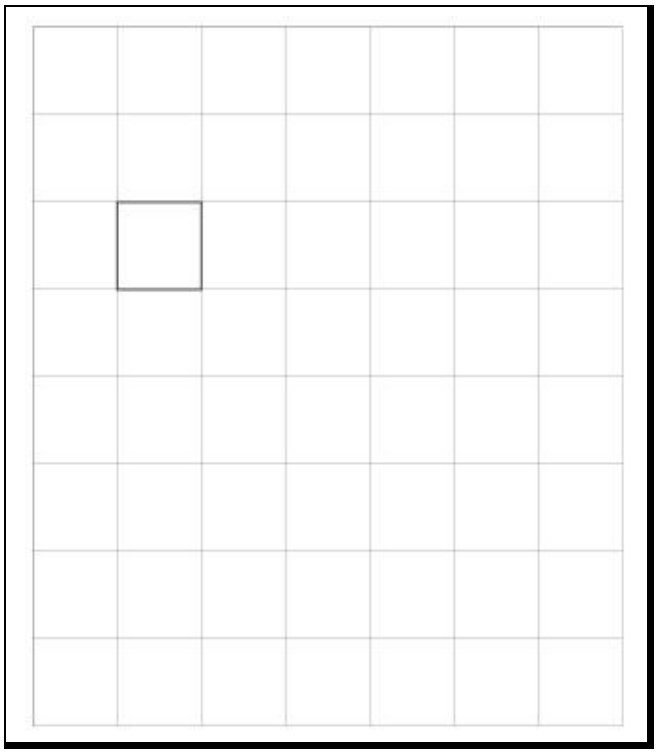

Figure 18. A square produced by cells that have the same width as height

- **Line Element**
  The line element is defined by using the "<line>" opening and closing tags, with any defining attributes within these tags (figure 19).

```
<symbol>
    <line>
        <position row="2" col="2"></position>
        <size></size>
        <angle>0</angle>
        <length>2</length>
        <offset x="0" y="0"></offset>
        <strokewidth>3</strokewidth>
        <updown>down</updown>
    </line>
</symbol>
```

Figure 19. The code that will produce a line

The line element introduces three new attributes a "length" attribute that specifies how many cells long a line is going to be, a "strokewidth" attribute that specifies how thick the line is going to be, and an "updown" attribute which defines whether the line is going to be the drawn upwards or downward from it's original position. The code in figure 19 will produce the line shown in figure 20.

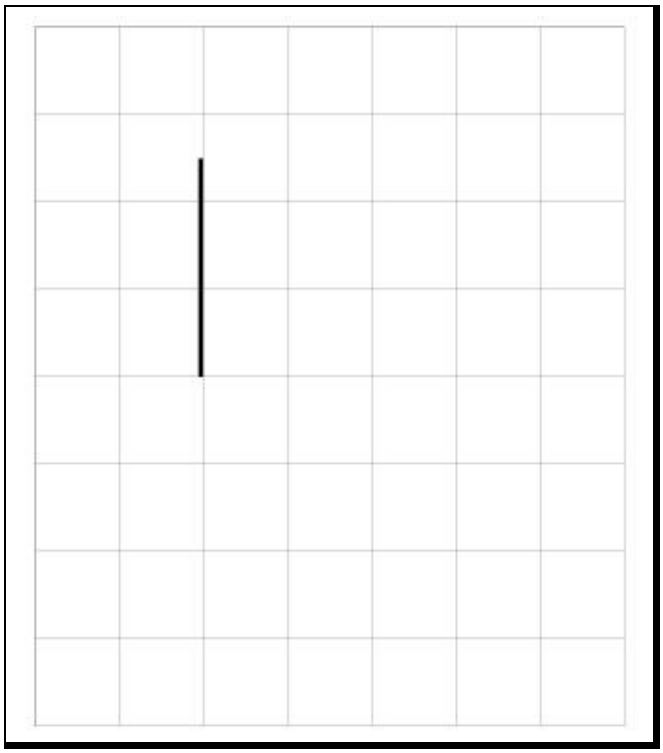

Figure 20. A Line element drawn on the grid

- **Polygon**
  The polygon is used to specify a shape that has many joining sides. A Polygon element is defined by using the "<polygon>" opening and closing tags with any defining attributes within these tags (figure 21).

```
<symbol>
    <polygon>
        <point x="25" y="50"></point>
        <point x="225" y="150"></point>
        <point x="125" y="70"></point>
        <point x="175" y="100"></point>
        <point x="325" y="150"></point>
        <point x="75" y="250"></point>
    </polygon>
</symbol>
```

Figure 21. The code that will produce a polygon with 6 sides.

Unlike the other elements only points are required to specify where each point of the polygon is to be placed. A polygon may have as many points as required and are specified by pixels and not by cells. Figure 22 below shows the outcome of the code displayed in figure 21.

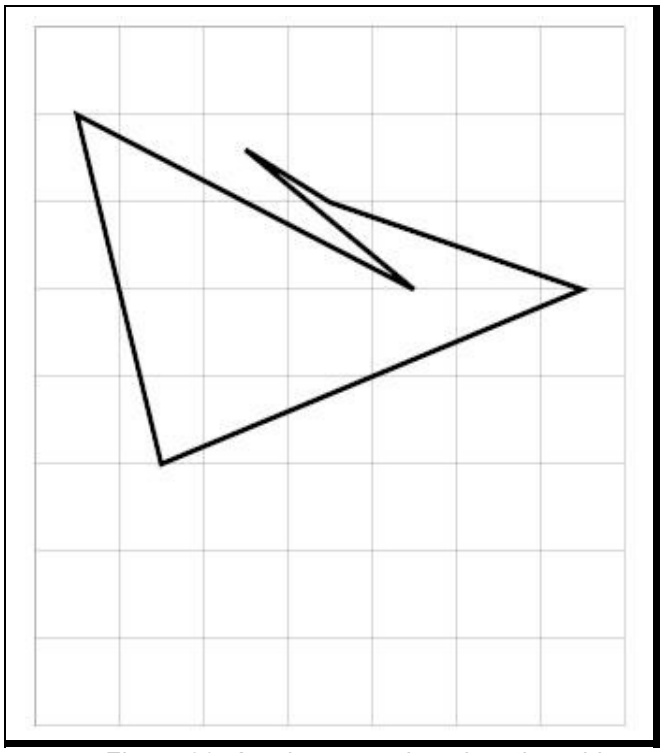

Figure 22. A polygon produced on the grid.

- **Polyline**
  Similar to the polygon in both code and outcome a polyline doesn't join its ending point with its starting point to form a completely closed object.
  A polyline can be defined by using the "<polyline>" opening and closing tags, with the points being specified within these tags (figure 23).

```
<symbol>
    <polyline>
        <point x="25" y="50"></point>
        <point x="225" y="150"></point>
        <point x="125" y="70"></point>
        <point x="175" y="100"></point>
        <point x="325" y="150"></point>
        <point x="75" y="250"></point>
    </polyline>
</symbol>
```

Figure 23. The code that will produce a polyline with 6 lines.

Like the polygon a polyline may have as many points as required and are specified by pixels and not by cells. Figure 24 below shows the outcome of the code displayed in figure 23.

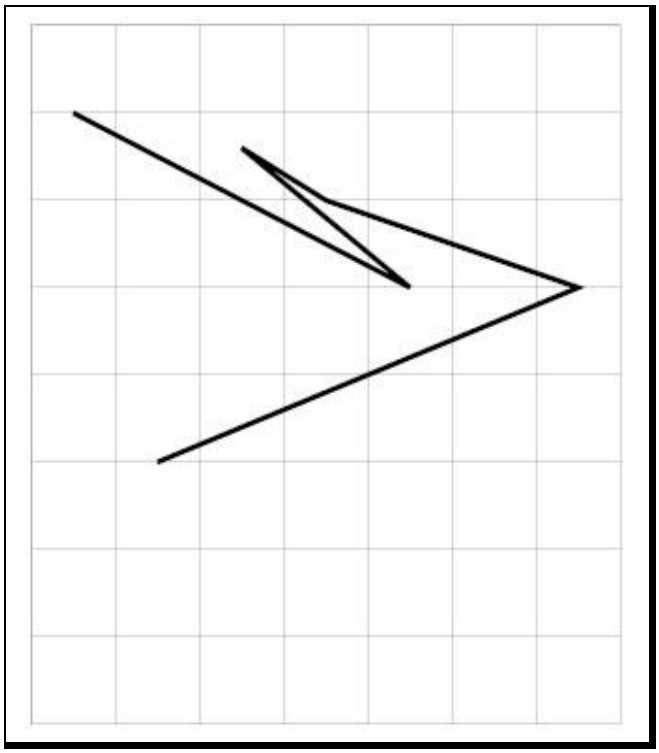

Figure 24. A polyline produced on the grid.

Multiple elements to make up a note

As mentioned previously multiple elements can be added together to make up a note. An example of a rectangle and a line and many different attributes used to make a quarter note is shown in figure 25 and 26.

```
<symbol>
    <rectangle>
        <position col = "5" row = "3"></position>
        <size cols = "1" rows = "1"></size>
        <angle>20</angle>
        <stroke color="green"></stroke>
        <fill color="orange"></fill>
        <rounded x = "20" y = "20"></rounded>
    </rectangle>
    <line>
        <stroke color="green"></stroke>
        <angle>20</angle>
        <length>2</length>
        <updown>up</updown>
        <strokewidth>3</strokewidth>
    </line>
</symbol>
```

Figure 25. The code to create a quarter note using different elements and different attributes.

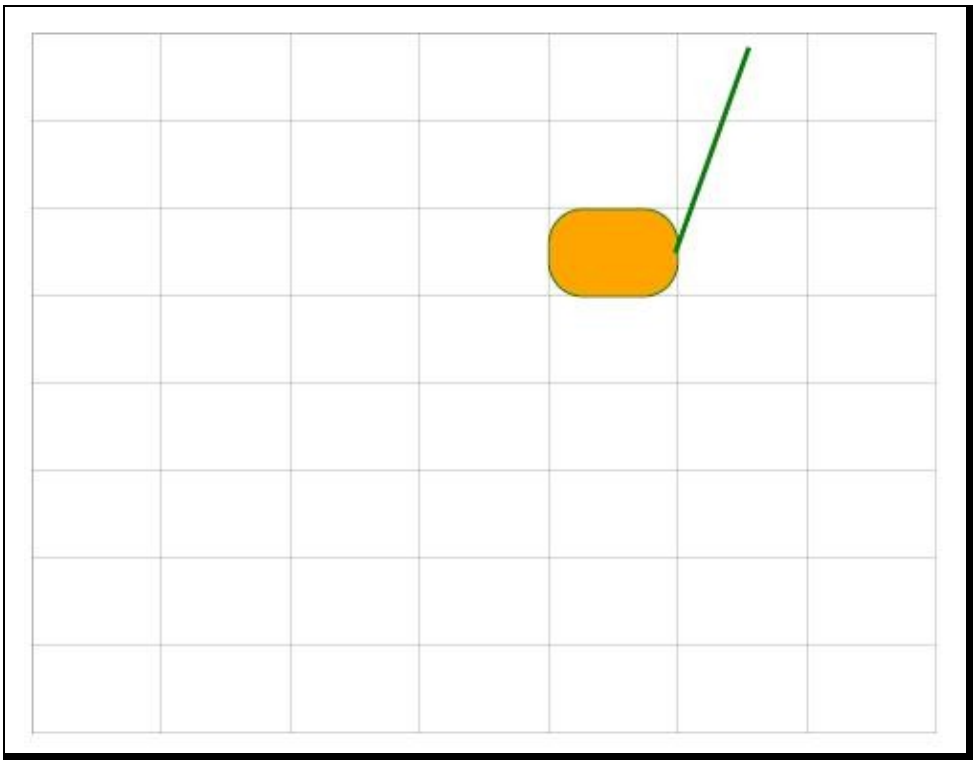

Figure 26. A quarter note produced by the code in figure 25.