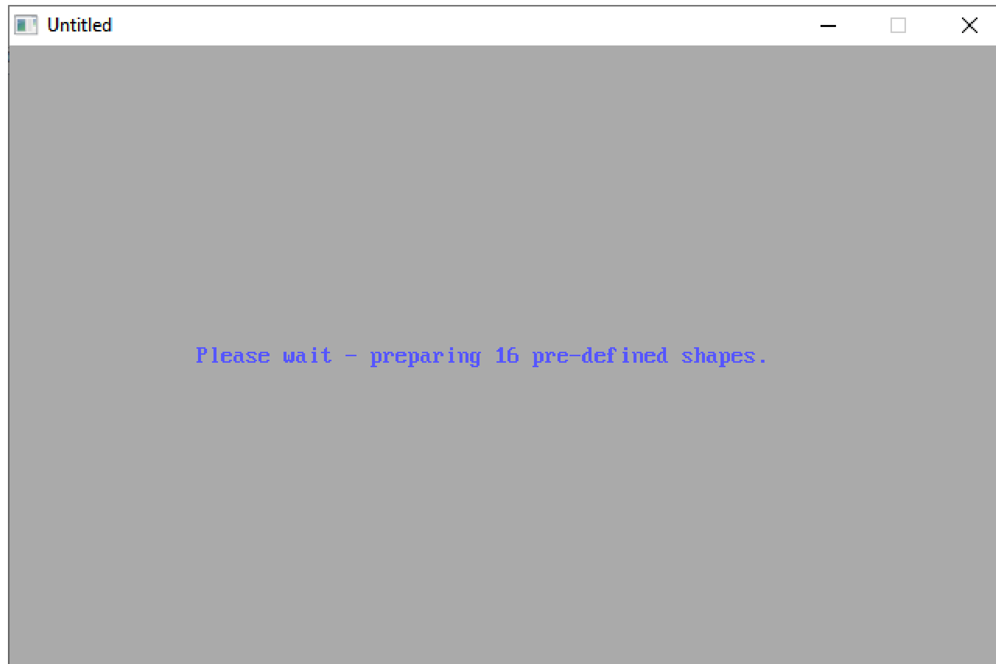
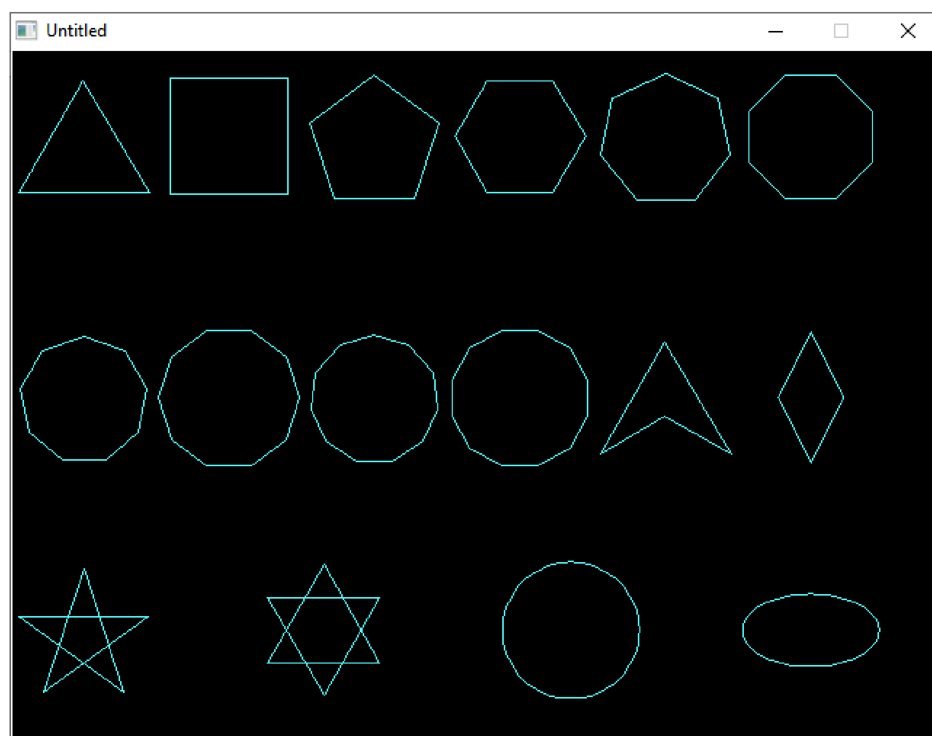
# Notes on the Demo

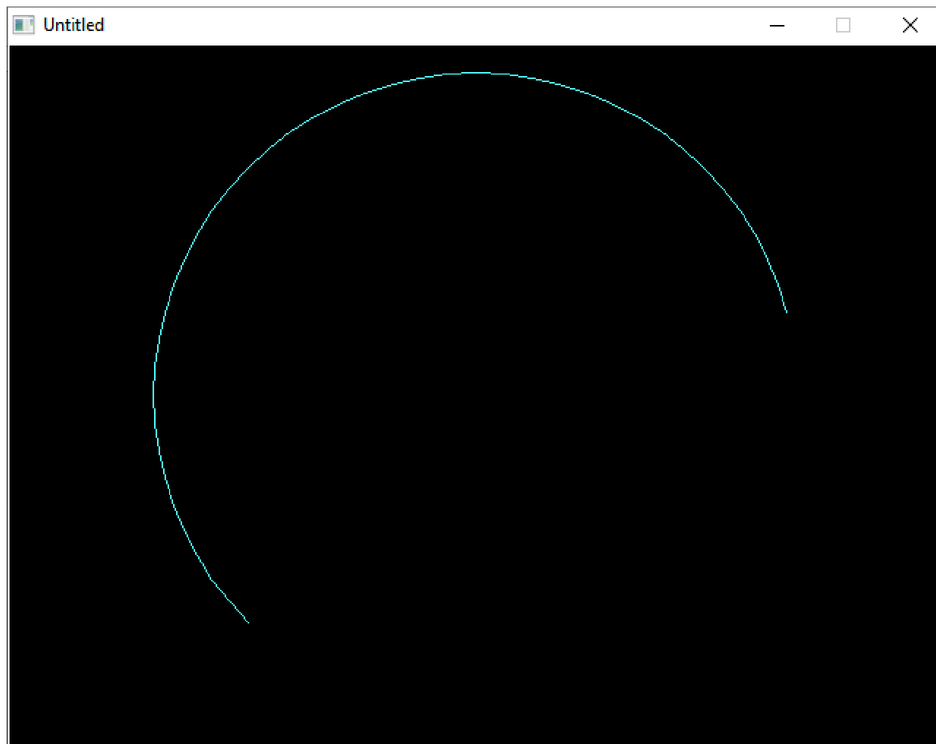
I have scattered half-second delays throughout the demo program so that you can see things happening. I have also included an annoying BLEEP in a number of places in the demo. It is there to let you know that the program is waiting for you to press any key in order to progress throughout the demo. I will now explain what you are actually seeing in the demo.




I actually added a three second delay, so that you could read this. Otherwise, you would never have seen it as the predefined shapes will have loaded before you read the first word. How technology has progressed in the last 20+ years.

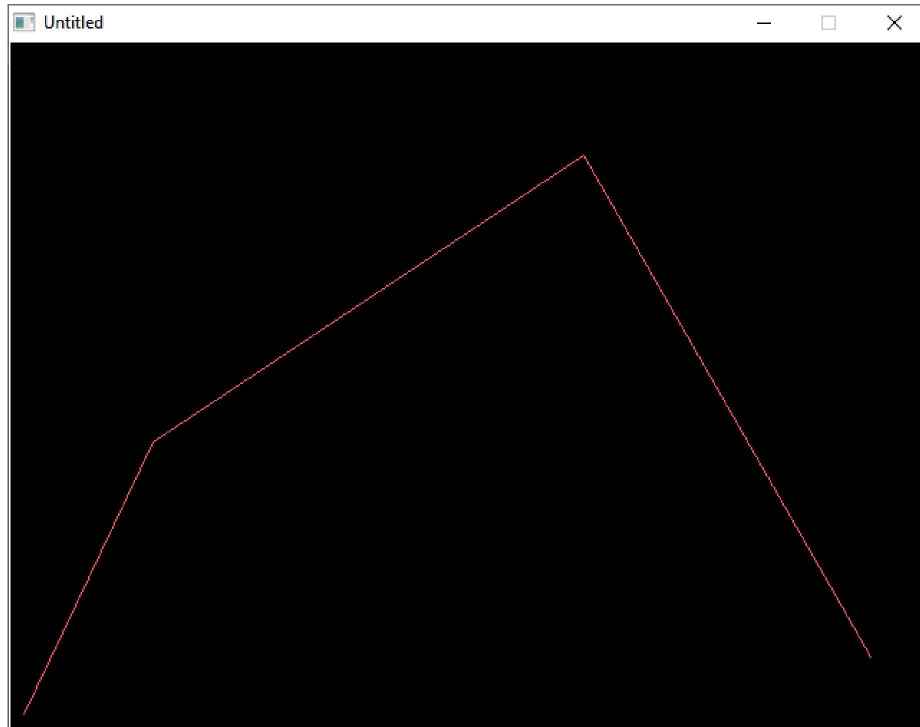


The above are derived from the pre-defined shapes found in the SHAPES2D.DAT file.

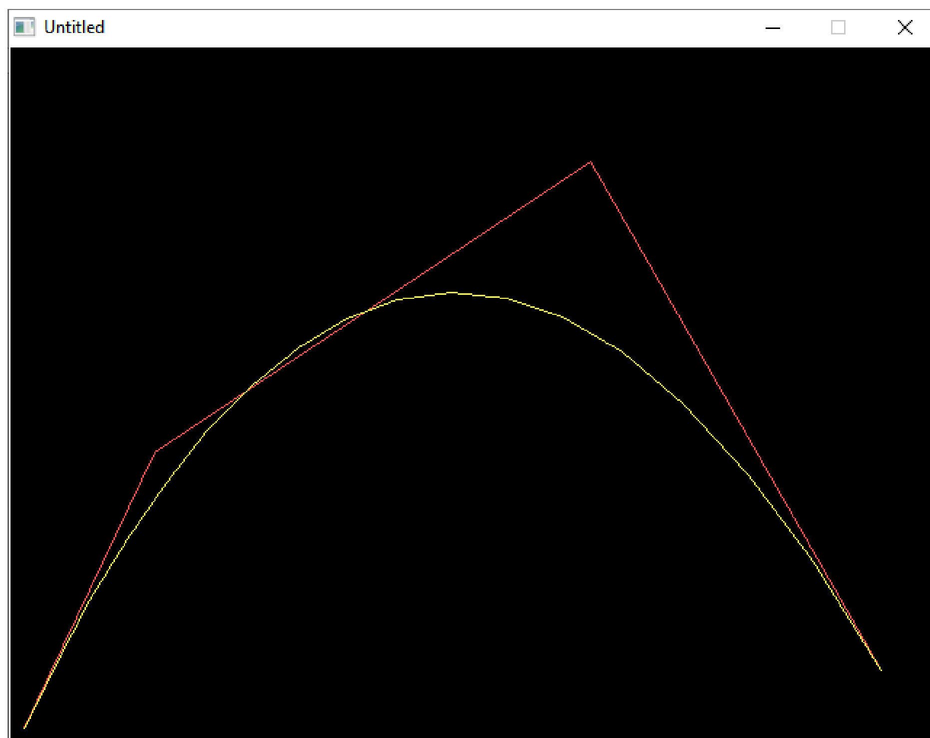



This is an arc (a section of a circle) calculated on the fly just before being displayed.

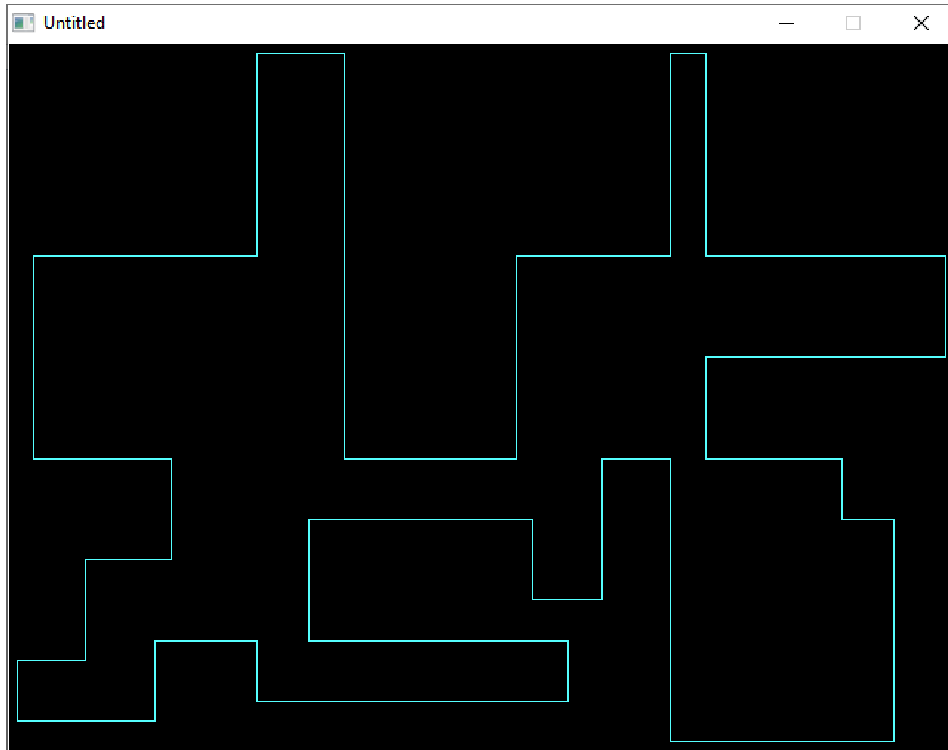
This is more interesting. First take 2 circles and calculate the best tangents to use to connect them so as to produce a smooth object. Then using these tangents, calculate arcs to join the tangents, storing the data in an array. Finally draw the result.




This is not what it seems. What we actually have here are 4 points in 2D space. In order to make them more visible they have been joined by three lines. We use these points to calculate




A smooth, non-circular curve that starts at the first point and ends at the last one. The other two (highest) points are used to control the shape of the curve. Expanding on this idea…

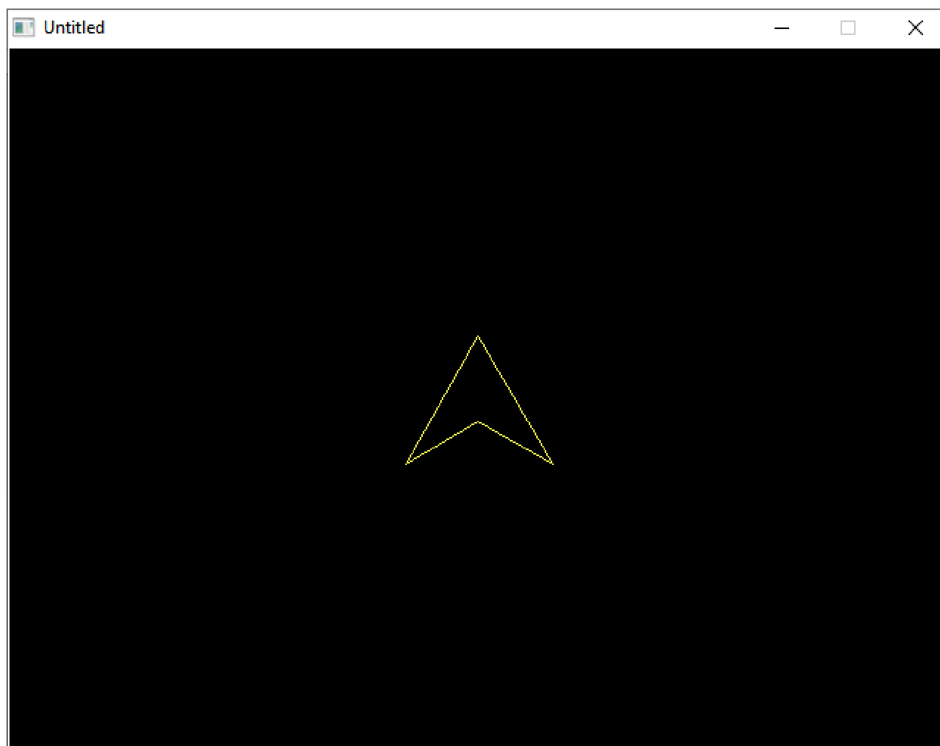



Incidentally, the shape above is stored in G2DEMO.DAT.

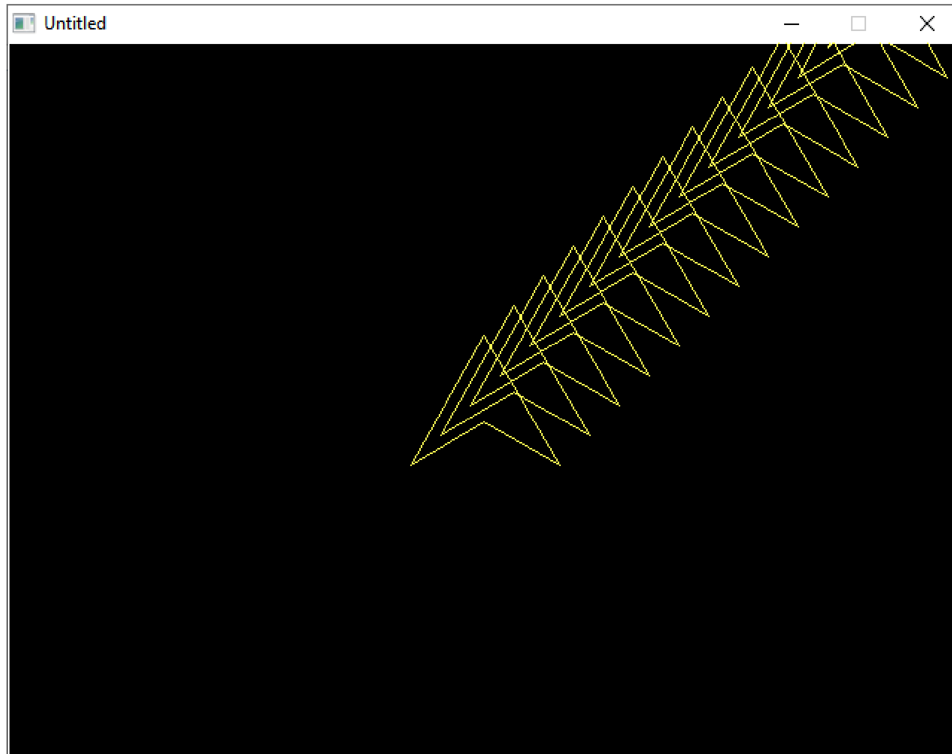
No this isn't some form of modern art. Each corner of that figure is a control point. I just drew lines so you can get an impression of where they are. So…

Here is the result. All we need to do now is remove the control points (and lines) so that you can more clearly see the resulting curved shape…

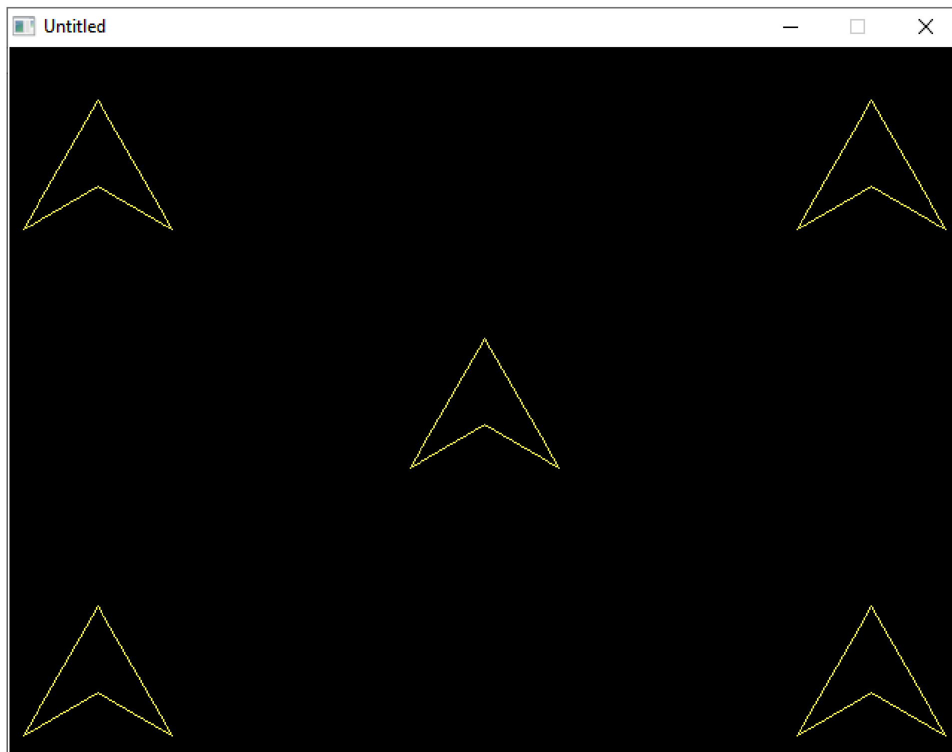
Notice that there are no obvious joins. All the curves flow into each other. And that's the thing. While you may believe that you can see some straight lines, they are in fact all curves. It's just the resolution that gives you the wrong impression.



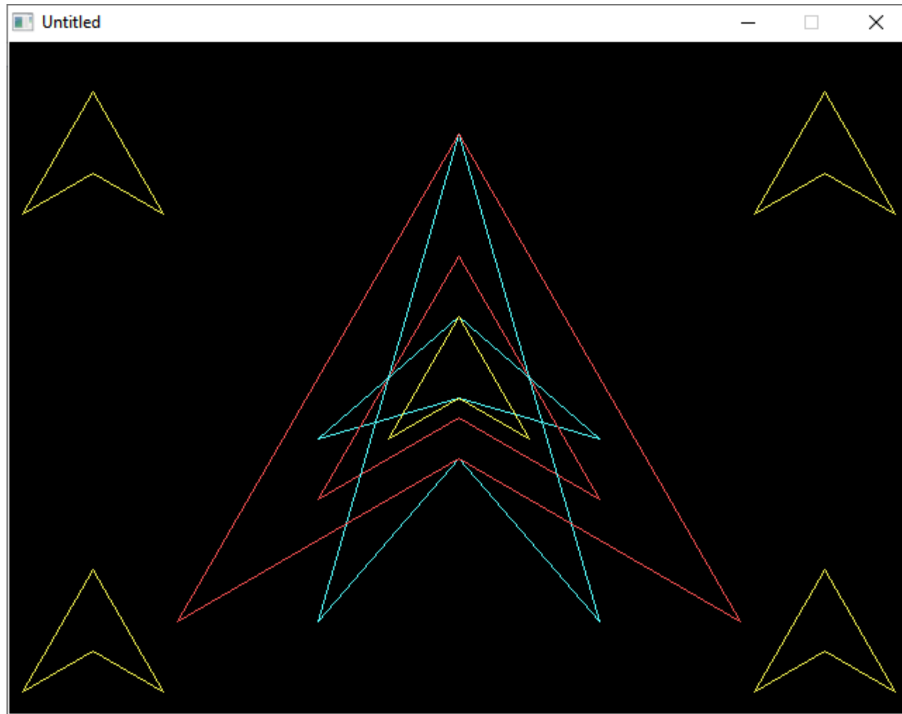

Yes, it's just one of those arrow shapes you saw at the start, so press any key…



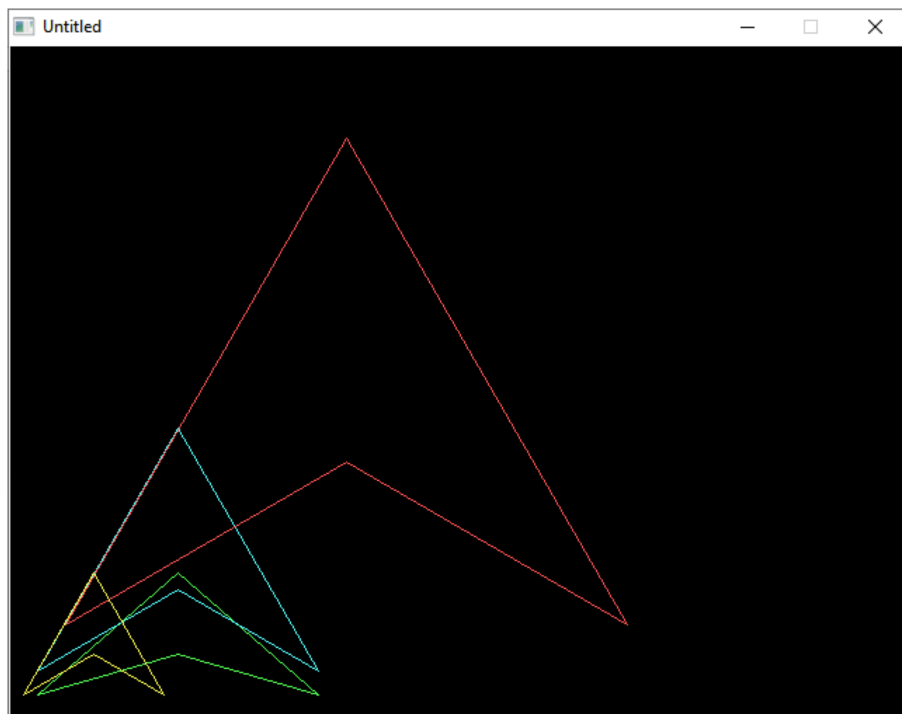

This is simply the result of drawing the shape and moving it, repeated a number of times.




As we have the shape stored in an array, we can use that data to repeatedly use it to stamp copies in various places on screen.

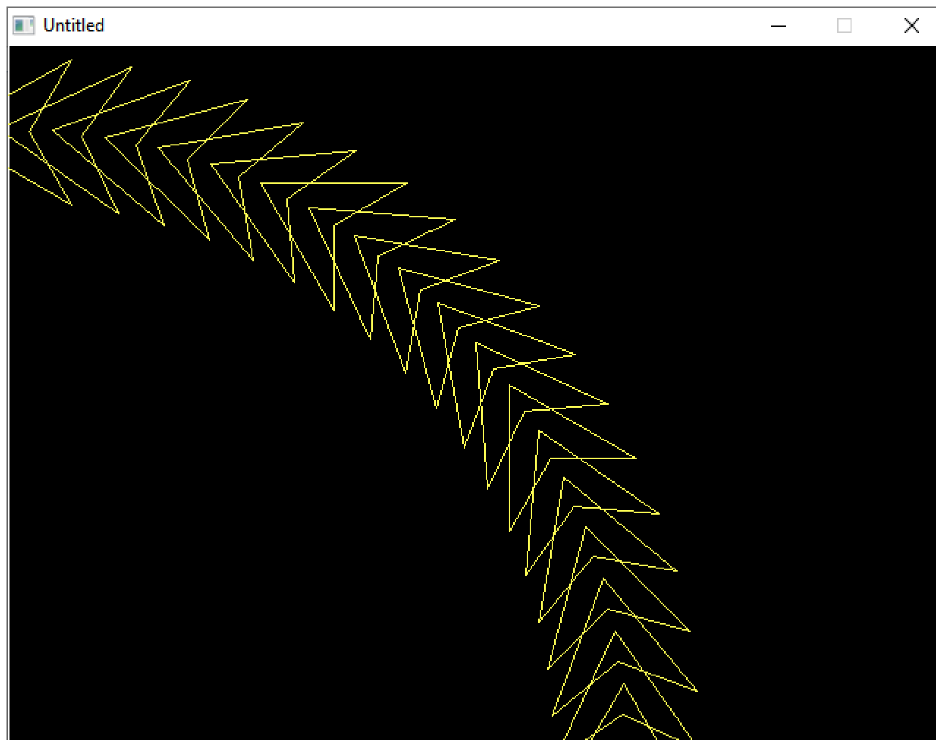


If you look at the centre of this image you will see that the original arrow is still there. Overlayed on this are a number of copies of it which have had their size increased in various ways without changing their centres. FWIW, I call this form of size adjustment inflation.

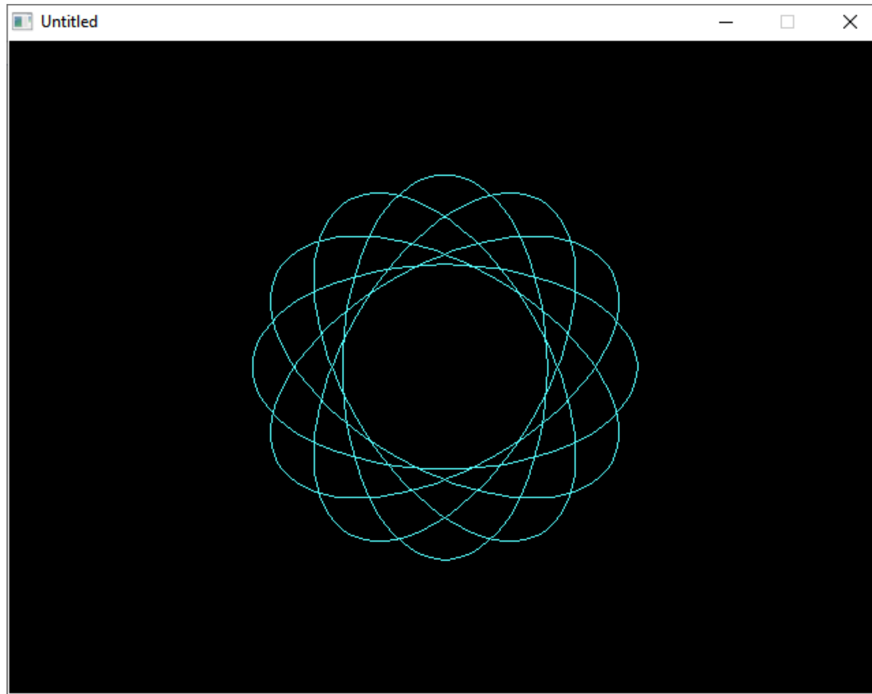


Whereas these have also been changed in size, the difference is because the standard routine doesn’t preserve the arrows’ centres.

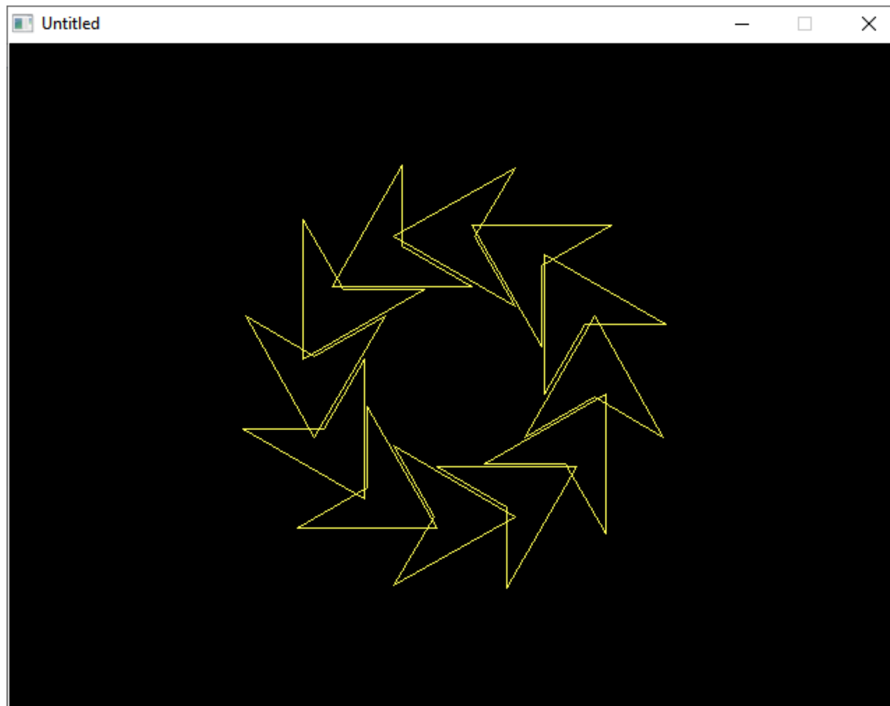
This on the other hand is not resized as such. Instead, a different technique, called shearing, has been used. To put it simply, shearing is a way to distort a figure in a repeatable manner.



We now move onto rotation. Without additional steps, a rotation occurs around a special point in 2D space. It is called the origin and is simply the point located at 0,0. In QB64 (and most other programming languages) this also happens to be the bottom-left hand corner of the screen.

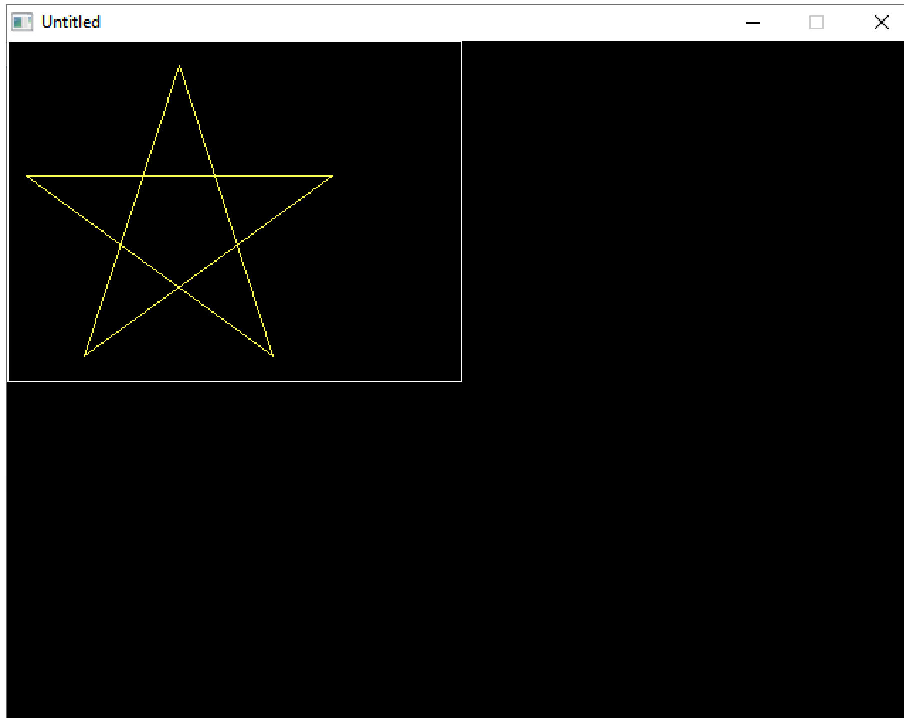



One of the ways we can improve on the standard method is to make the graphics object rotate about it's centre (which is why I store the location of the centre of each graphics object in the same array as the rest of their data.

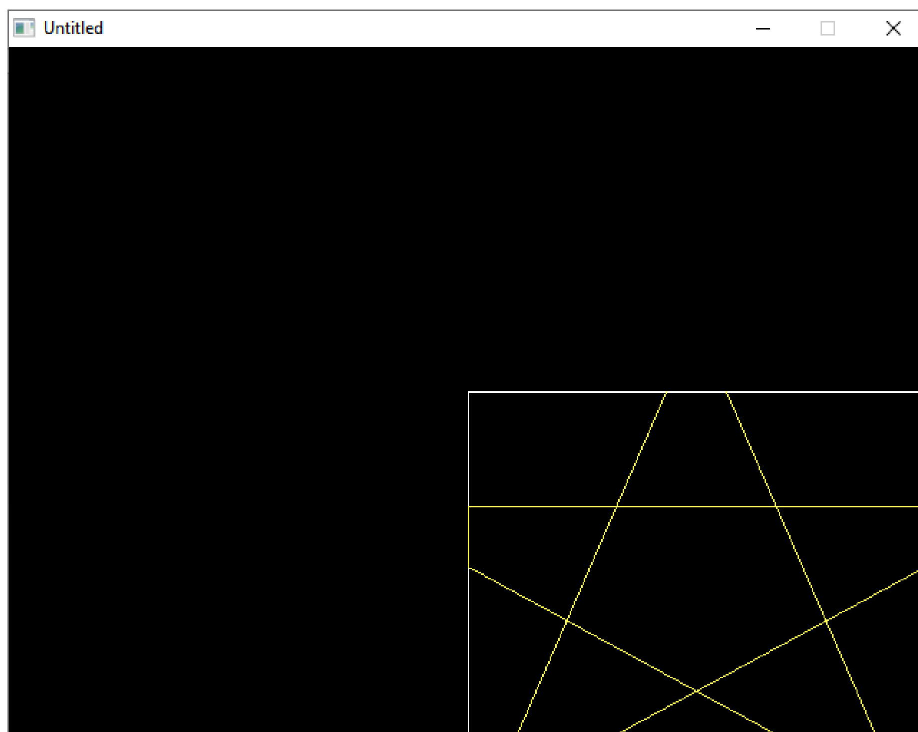



Another way that we can rotate things is to do so around a common centre. FWIW, I call this particular method orbiting.

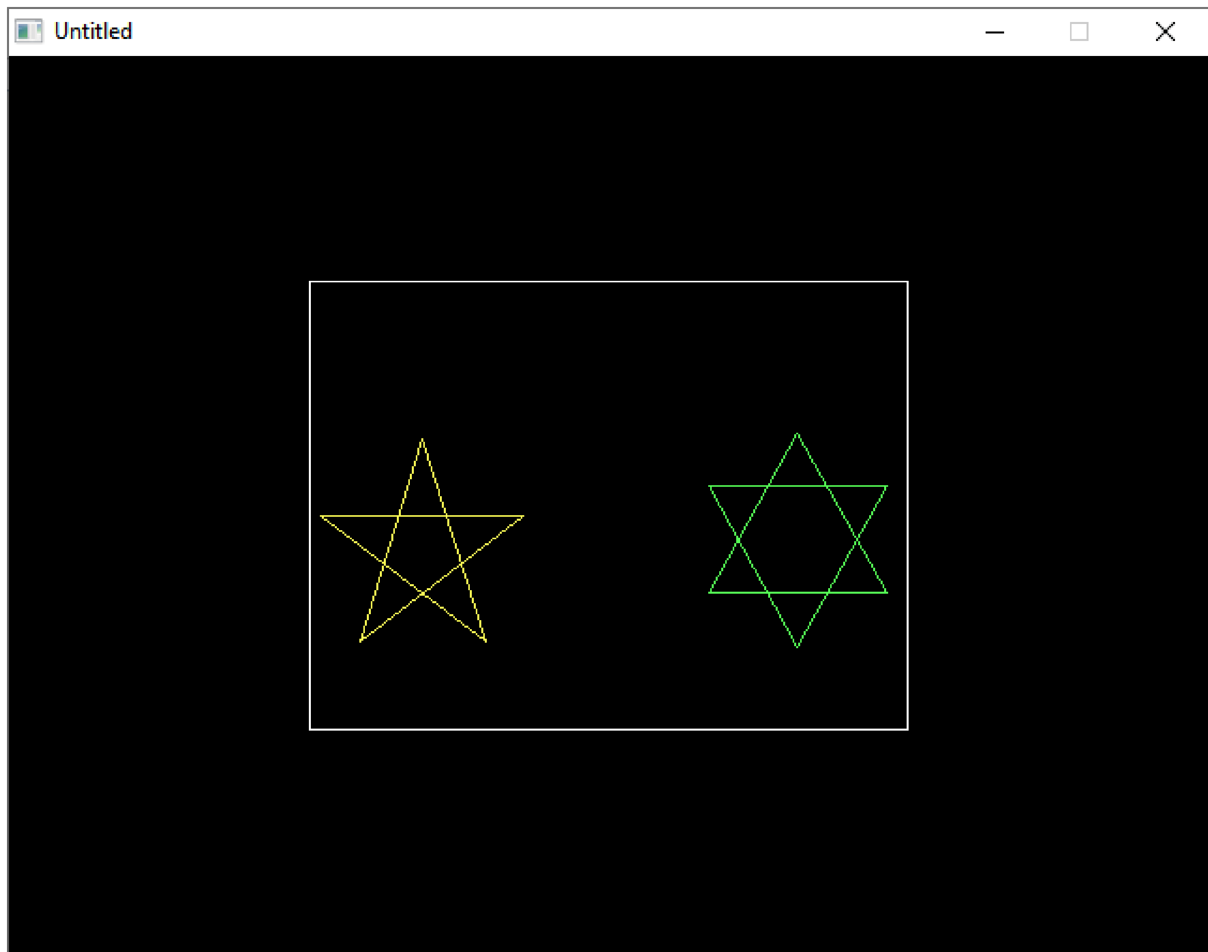
Now onto the last three screen that deal with controlling what is actually displayed without breaking the stored shapes.




If you look at the BI file you will see that there are three user-defined TYPEs. These are used in combination in order to control what parts of the objects are actually displayed. In order to make it clearer I have used the data contained in those types to draw a box around the above pentagram.




This is the same pentagram. It is just we have moved the “box” to the bottom-right corner and “zoomed-in” to the pentagram. However, neither of the above images show the whole of the story as they are both “zoomed-in” .




When we "zoom-out" all the way, it is revealed we were previously concentrating on less than half the picture.

A quick note about the above picture. The pentagram and the star of David use exactly the same points as the pentagon and the hexagon previously shown. It's just that the above two make use of extra information in the form of connection lists – lists which define where lines will be drawn.

And that's the demo done with. The demo shows a good proportion of what can be done with the library but by no means all. Go through the all the code reading the comments and experiment. Just try stuff out.

I am convinced that anyone who knows their way around QB64 and the Inform utility could use this library to make a simplified version of Inkscape or a simple CAD program using it.

Anyway, have fun.

TR