**CGContextReplacePathWithStrokedPath with Cario**

**High Level Design Overview**

The cairo\_stroke\_to\_path method is currently not implemented in Cairo so another approach was needed to bring the stroked path functionality to CGContextReplacePathWithStrokedPath in a timely fashion.

The approach chosen was to use cairo\_copy\_path\_flat which creates a path with no curves. “Curves in the path will be approximated with piecewise-linear approximations, (accurate to within the current tolerance value). That is, the result is guaranteed to not have any elements of type CAIRO\_PATH\_CURVE\_TO which will instead be replaced by a series of CAIRO\_PATH\_LINE\_TO elements.”

This approach simplifies the current implementation such that curves do not have to be taken into account. However, there are several items that still need to be addressed:

1. Handling of multiple sub-paths
2. Path traversal changes between left to right vs right to left drawing.
3. End caps applied: Butt, Round, Square
4. Line joins applied: Miter, Round, and Bevel

The down side of this approach is that curves are just approximations and take up more memory because of the increased number of points involved in the path.

**Multiple Sub-Paths & Left to Right Drawing**

The following shows a path containing two sub-paths drawn from left to right in WinObjC. There is no visual difference in left to right vs. right to left paths.

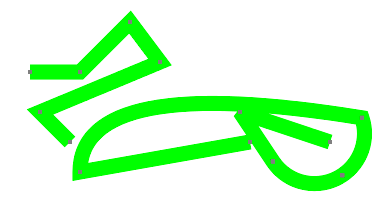


Figure : WinObjC Original Path

In addition to the path, little squares are drawn at each point used in creating the path to illustrate the differences in the underlying path. The first sub-path contains just lines, the second sub-path contains lines and curves.

Figure 2 shows the WinObjC output of CGContextReplacePathWithStrokedPath in WinObjC with Miter line joins. One can clearly see the curve approximations as it takes many points to reproduce it correctly.

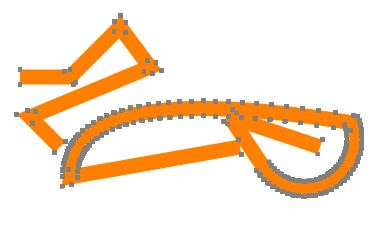


Figure : WinObjC Stroked Path

Figure 3 shows the iOS output of CGContextReplacePathWithStrokedPath. This is shown in slightly different scale as it’s a screen capture from an iOS device. However, it can clearly be seen the end results are very similar. The iOS implement just requires less points as it’s using true curves.

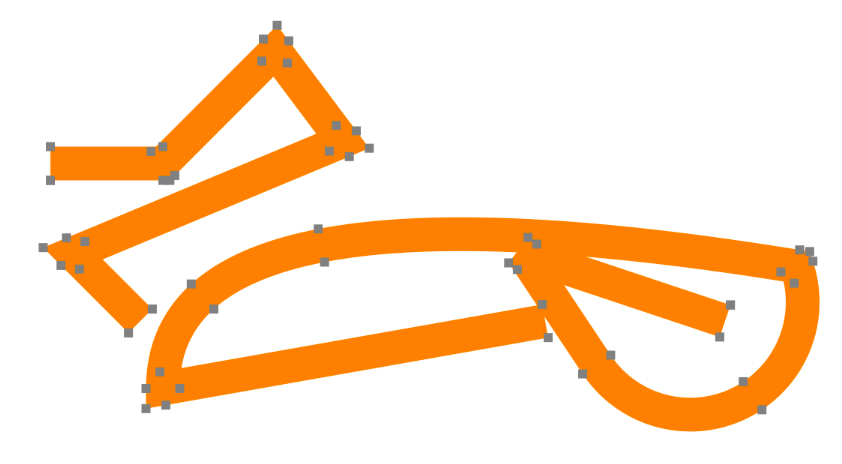


Figure : iOS Stroked Path (slightly different scale)

.

There is effectively no visual difference, other than the number of points required to create the path.

**End Caps**

Butt, Round, and Square end-cap styles can be applied to the beginning and end of each sub-path. They are implement by a simple Lambda method (addLineCap). Here are examples of all three end-cap styles in WinObjC:

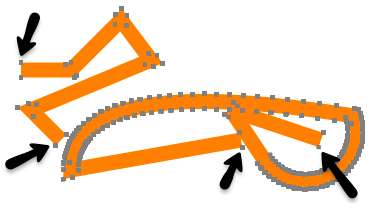


Figure : Butt End Cap

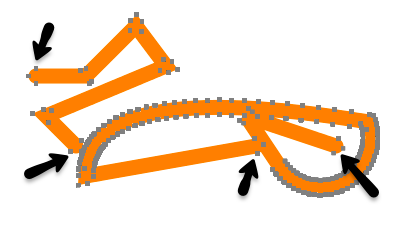


Figure : Round End Cap

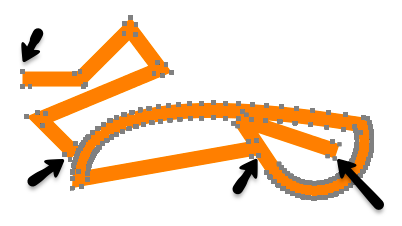


Figure : Square End Cap

**Line Joins**

Miter, Round, and Bevel line joins are supported via a Lambda method (addLineJoin). Parts of addLineJoin were taken from the Cario implementation in order to account for things like miter limit and the handling of acute angles (see cairo-path-stroke.c:\_cairo\_stroker\_join). Here are examples of the different line join techniques in WinObjC:

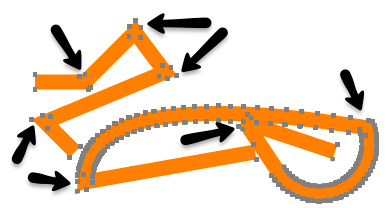


Figure : Miter Line Join

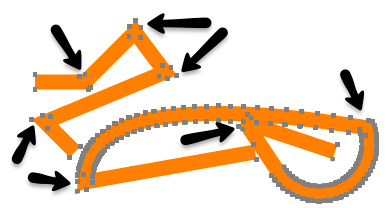


Figure : Round Line Join

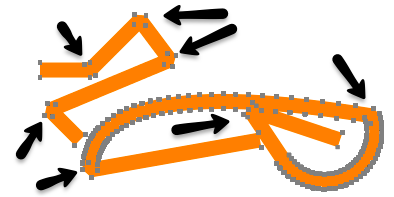


Figure : Bevel Line Join