Rounding image corners in CF8

Posted At : 20 October 2008 09:39 | Posted By : Gareth
Related Categories: cfimage, coldfusion

I'm currently working on a site where the staff can upload images for display. Part of the design requirement, is that the corners of these images are rounded off. Step forward coldfuson 8...

Image support in coldfusion was long overdue. Although there were some very good 3rd party components (such as the Alagad Image Component), there should have been something bundled with coldfusion many versions ago. Coldfusion 8 comes with the new <cfimage> tag and dozens of image manipulation and drawing functions. So drawing rounded corners should be easy?

Unfortunately not...

What I wanted to build was a generic function that could apply rounded corners of any colour and size to a coldfuson image object.

My first thought was to create partially transparent corner images in photoshop and paste them on top of the target image. Although this would have worked for one project, I knew that further projects would require the same functionality, but with different sized corners and different colours. I didn't even bother trying this method.

I started looking at image functions such as **ImageDrawRoundRect()** and **ImageDrawArc()**. The problem with all of these though, is that they masked the wrong part of the images - all that was showing of the original image was the corners. I wanted to somehow reverse them.

After some playing around, I discovered that you can create a new image object with a transparent background. Next was to draw a rounded rectangle over it. The surprise came when I tried the **ImageNegative()** function on it - it reversed the image so that the centre was transparent as required. This could then be placed on top of the target image as shown in the code below:

```
img = imageNew(sourcepath);
   ImageSetAntialiasing(img);


   corner = ImageNew("",ImageGetWidth(img),ImageGetHeight(img),"argb");
   ImageSetDrawingColor(corner,red);
   ImageSetAntialiasing(corner);
   ImageDrawRoundRect(corner, 0, 0, ImageGetWidth(img), ImageGetHeight(img), diameter, diameter, true);
   ImageNegative(corner);
   ImagePaste(img, corner, 0, 0);
```

After a little testing, it soon became obvious that this technique seemed to ignore the colour setting of the corners - it seems that the negative of transparent was always white. While this would be ok for most projects, there would be some where I needed corners of a different colour because the page had a non-white background.

After further experimenting, it seems that the strange **ImageXORDrawingMode()** function was actually of use:

```
img = imageNew(sourcepath);
   ImageSetAntialiasing(img);


   corner = ImageNew("",ImageGetWidth(img),ImageGetHeight(img),"argb");
   ImageSetDrawingColor(corner,red);
   ImageSetAntialiasing(corner);
   ImageDrawRoundRect(corner, 0, 0, ImageGetWidth(img), ImageGetHeight(img), diameter, diameter, true);
   ImageNegative(corner);
   ImagePaste(img, corner, 0, 0);


   ImageXORDrawingMode(corner,"ffffff");
   ImageDrawRect(corner, 0, 0, ImageGetWidth(img), ImageGetHeight(img), true);
```

This method met all my requirements, but somewhere along the line, the anti-aliasing was lost. For larger corners the results were ok, but a smaller radius could get very blocky.

Final Function

The final attempt took a very different approach. I realised that the **ImageDrawRoundRect()** function could also be used without being filled. You could also vary the width of the line.

The basic idea was to draw a rounded rectangle on top of the target image. The rectangle would be larger than the image, so that only the curved corners would overlap. Careful calulation of the rectangle's dimensions and thickness of line were required. The method worked well though after some adjustments. The final working function is shown below:

```
<cffunction name="roundImageCorners" returntype="any" access="public" output="false" hint="I take an image, object and return it with the
corners rounder off">
      <cfargument name="image" type="any" required="true" hint="The image to be modified" />
      <cfargument name="radius" type="numeric" required="false" default="50" hint="Radius width of the corners" />
      <cfargument name="bgcolor" type="string" required="false" default="ffffff" hint="Colour of the rounder corners" />

      <cfscript>
         var t = "";
         var pen = structNew();
         var root2 = sqr(2);

         ImageSetAntialiasing(arguments.image);
         t = arguments.radius * (root2 - 1);
         ImageSetDrawingColor(arguments.image,arguments.bgcolor);
         pen.width = t;
         ImageSetDrawingStroke(arguments.image,pen);
         ImageDrawRoundRect(arguments.image, 0-(t/2), 0-(t/2), ImageGetWidth(arguments.image)+t, ImageGetHeight(arguments.image)+t,
arguments.radius+(t/2), arguments.radius+(t/2), false);

         return arguments.image;
      </cfscript>
   </cffunction>
```