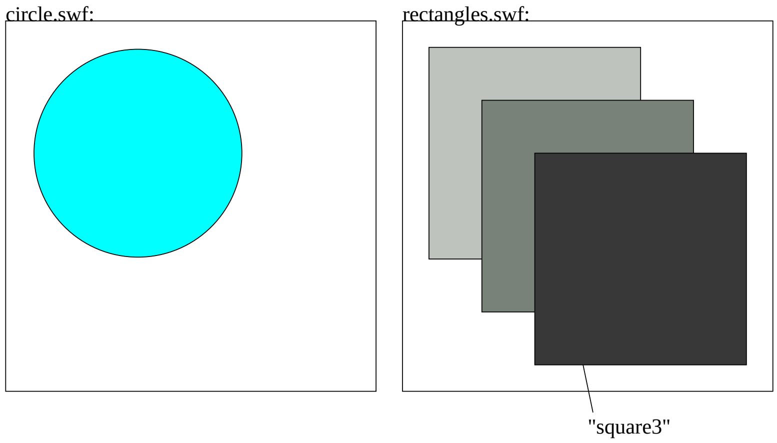
C Appendix C: SWFCombine Manual

SWFCombine is a tool for merging separate SWF files or modifying SWF parameters.

C.1 Named Objects

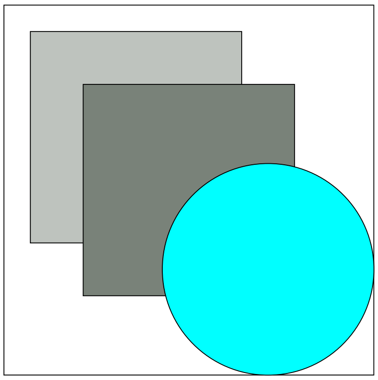
A simple example is as follows. Suppose you have two SWF files, ”circle.swf” and ”rectangles.swf”, depicted below. Both these SWF “movies” only contain a single frame.



In ”rectangle.swf”, the 3rd square object is **named**, with the name ”square3”. You can give all objects in SWF Movies names. These names can be used by SWFCombine to apply merging operations:

```
$ swfcombine rectangles.swf square3=circle.swf
```

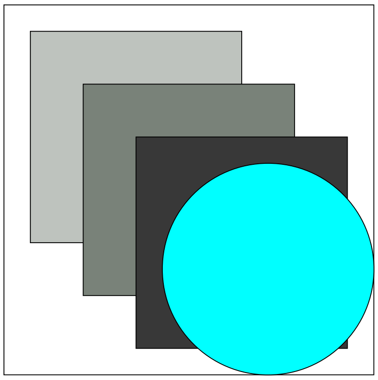
gives: (see next page)



Now, both movies became one, with both the circle and the squares in it. (Except the named one). In effect, square3 has been **replaced** with the circle. You can also **overlay** instead of replace. This is what the -l (–overlay) option is for:

```
$ swfcombine -l rectangles.swf square3=circle.swf
```

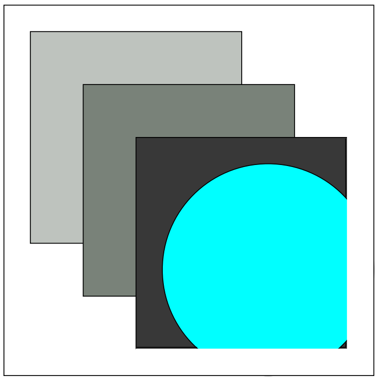
gives:



Here, both the circle and the rectangle movie are fully visible. In the cases where the named object has a ”real” shape (as opposed to being an invisible marker for the sole purpose of telling swfcombine what to do), this is usually what you want.

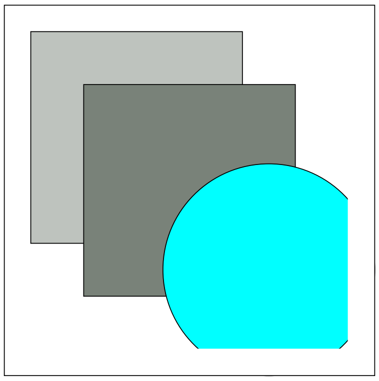
In this case, the "marker" only tells swfcombine *where* to insert circle.swf. (i.e., at which x and y position, and at which depth [5]) However, the named character can also be used for other purposes, like clipping, using the -c (–clip) option:

```
$ swfcombine -c -l rectangles.swf square3=circle.swf
```



The second movie lies as well *above* and *inside* the named object. Notice that we used both the -c and -l options. If we had used only -c, the rectangle wouldn't have been visible, while still clipping the circle:

[5]The depth of an SWF character is it's priority: The character with the highest depth shadows all characters with smaller depths. The character with the smallest depth lies behind all other characters.



C.2 Stacking, concatenating, merging

The following chapter made use of named objects to combine movies. The motivation for this was the fact that it should be possible to specify positioning of the second object. If, however, this is not needed, it is also possible to combine movies without using named objects.

The simplest example is the -T (–stack1) option:

```
$ swfcombine -T square.swf circle.swf
```

This just places the movies on top of another. Notice: The movies will play independently from each other. For example if movie 1 has 3 frames and movie 2 has 10 frames, then movie 1 will loop for 3 times before movie 2 loops the first time.

There's an alternative to the -T option, the -t (–stack) option. This option follows a quite different principle: Every movie is placed in it's own frame. Notice that, therefore, only the first frame of each movie will be visible!

Both -T and -t can be used together with -m (–merge).

C.3 Modifying single SWF streams

swfcombine can do other useful things, too. For example, it can compress SWF files using the Flash MX Zlib compression feature. To compress an SWF file, use:

```
$ swfcombine -dz input.swf -o output.swf
```

You can also uncompress already compressed files with

```
$ swfcombine -d input.swf -o output.swf
```

The reason, we need to give the -d option in both cases is that swfcombine would throw an error due to the fact that there's only one input file. With using -d (–dummy) you tell swfcombine, that you indeed just want to work with one file.

Other uses:

```
$ swfcombine -d -r 100 input.swf -o output.swf
```

sets the framerate of the SWF file to 100.

```
$ swfcombine -d -X 2000 -Y 2000 input.swf -o output.swf
```

sets the bounding box to 2000*2000.