Brian Huntley

Estimated cache hit rate

	Row major	Column Major	Block Major
90 deg rotation	2	2	1
180 deg rotation	3	1	1

180 degree rotation column major - I expect this method to have the best hit rate. My UArray2 is designed as an array of arrays, where the outer array represents columns. This means that the values inside each column are stored near each other in memory. Therefore reading and writing from my UArray2 will all be a hit in this case.

180 degree rotation block major - I also expect this method to have the best hit rate. UArray2b will have blocks which will be able to fit into cache memory, and all the pixels in the blocks are near each other. Again, this will mean that reading and writing will all be hits, since the pixels are all grouped together.

90 degree rotation block major - I believe that this method will also tie for the best hit rate. Reading pixels will be all hits, as the pixels in each block are all located near each other in memory. Rotating the block 90 degrees may impact the performance per pixel when writing, but since all the pixels are in the same block that will not impact hit rate.

90 degree rotation column major - This method will have the second best hit ratio. As explained earlier, my UArray2 implementation allows values in columns to be localized. So reading values from the original will all be hits. However, since the image is rotated 90 degrees, the rows and columns get swapped. Writing the column into different rows will result in many cache misses since the pixels will now be spread across multiple different columns.

90 degree rotation row major - This method will perform the same as its column major variant. The reason is the same, just in the opposite direction. Reading pixels from each row will have lots of misses, as they are spread out across different columns. But when writing those pixels into the rotated UArray2, the pixels will all be next to each other in the same column.

180 degree rotation row major - I predict that this method will have the worst hit rate. Reading each pixel will result in many misses as each value read will be from a different column, meaning they are spread out far in memory. Writing the data into the rotated image will result in the same problem, as this is a 180 degree rotation. Each value will get written into a different column.

Cost per pixel estimate

	add/sub	mult	div/mod	compare	loads	Hit rate	stores	Hit rate
180 row major	4	0	0	0	1	0	1	0
180 col major	4	0	0	0	1	1	1	1
180 blk major	6	2	8	0	1	1	1	1
90 row major	4	0	0	0	1	0	1	1
90 col major	4	0	0	0	1	1	1	0
90 blk major	6	2	8	0	1	1	1	1

Expected speed estimate

	Row major	Col major	Block Major
180 degree	4	1	2
90 degree	3	3	2

The block major method has a higher cost per pixel than the other two methods, however I do not think this will be enough of a difference to negate its cache hit rate advantages. That being said, the column major 180 degree rotation will likely perform the best, as it has a low cost per pixel and the highest hit rate.