Why RunLength encoding?

animation usually cuts corners, doesn't animate wherever possible

800 x 450 = 360000

800 x 450 x 149 = 53640000

33640550 total equal value pairs

33640550 / 53640000 = .62715417598, or 62.72%

Method 1:

2 bytes to record every run (1 or greater)

use 1 byte to record runlength (150 frames, 7 bits (128 max runlength) not enough)

largest run was 150

another byte to record the actual number

we know we have 62% run coverage, so worst case, we have a <38% increase in size

if 62% are all runs of length 2, and we double the number of bytes needed to store the 1 length runs

technically, 38% increase is unachievable since there can only be a max of 50% runs of length 2

best case, we have a < 24% reduction in size

this would be if all the entire 62% run coverage were long runs. However, 38% of data is still doubled in size, so 100-38\*2 = 24. Then we have 1 frame's worth of data (each pixel taking 2 bytes) that covers 62% of our data

With method 1, upper limit on file size is 38% increase, lower limit is 24% decrease

Method 1: went from 52735 kilobytes to 39765 kilobytes, 25.6% reduction in size

not sure why...

inefficiency in decompression time from reading it into 3d array, then outputting

Method 2:

2 bytes to record a runlength

n total pixels, n/8 bytes to determine runlength vs a run of non matching

this "information" byte will be every 8 bytes

12.5% increase bytes to include this information, but cuts the theoretical 38% increase from doubling amount of bytes used by 1 length runs. Overall, theoretically should result somewhere from 49.5% decrease in size to 12% increase in size from the original.

overall, is a better way to do it because of how animation works. If one pixel is changing, it is likely to change over a period of time, then stagnant for a period of time, and so on.

However, this "information byte" could easy cut into runlengths. This would split that one runlength into two, which would add 1 redundant byte. Worst case, if the information byte split a runlength every single time, we would see 12.5% increase bytes from information bytes and 12.5% increase from redundant bytes, meaning we only get a 37% decrease in size, and potentially 24.5% increase in size.

Method 3: doing RL along the xy plane instead of z plane- bad, need to test just how bad

Method 4: splitting the RL byte to record nonmatching runs (0-127, max run length 128) and matching runs (128-255, max run length 129). This means that runs of longer than 129 will be split, but these are uncommon.

why 129 and not 128? a matching runlength is always great than 1, so we can do runLength-126 to find true runlength.

inefficiency in compressing because we need to put runlength first instead of last. This way, we have to read values to determine how long of a runlength, then write runlength, then write actual values