Creating Videos

You have long understood how to watch an individual image get drawn using the various *Drawing Modes* in the web model but here we focus on a different way to create a series of images that could be viewed as a video rather than as a single static image. The method we model here is based on the often-used idea of allowing ***P*** to vary while holding ***n***, ***S*** and jumps fixed. This, for example, is what led to the discussion of waves of images.

What happens to images as *P* varies? As noted elsewhere, VCF = GCD($J_1+J_2+\ldots+J_k,n$) and v_{used}= $k \cdot n$/VCF, some of which may be used multiple times, in the jump set context. Since ***n*** and jumps are fixed, so is v_{used} and more concretely, *so is the vertex frame*. What varies as ***P*** varies is SCF = GCD($P,S \cdot v_{used}$) and hence lines in the image = $S \cdot v_{used}$/SCF.

Avoiding unwanted distractions. If you start from a random image in the web model and decide to scroll across ***P*** for "nearby" images, you are likely to run into images where the number of lines varies often and dramatically due to variations in SCF. To take an example that has been used often in multiple jump settings, if v_{used} is a multiple of 12 then as you scroll across images you will run into SCF that varies widely. Take, for example, the 144 line (24,3,11,J(15,2)) image discussed here. With ***P*** = 11, SCF = 1, but the next 15 values of SCF are: 12, 1, 2, 3, 16, 1, 18, 1, 4, 3, 2, 1, 24, 1, and 2. This means that the number of lines in the image varies from 144 to 6 and more importantly, the overall image density is constantly changing due to the great deal of commonality between numbers in this instance. This makes scrolling over ***P*** seem like a very disjointed experience.

Scrolling made easy. If you have a keyboard with directional arrow keys, click on the number in the *Points* area (so it turns blue) and use the ^ or v key to increase or decrease ***P***. You can hold this key down rather than tap it one ***P*** per tap.

Minimizing density changes. The commonality problems come from various sources. Note, in particular, that ***n***, ***k***, ***S*** and the sum of J_i all contribute to determining $S \cdot v_{used}$. *One thing to do is to think strategically about these values.* By making ***n***, ***S*** and ***k*** all powers of a *single prime number* the resulting image will have SCF > 1 only if ***P*** is a multiple of that number.

Examples. Take, for example, 5. The resulting image will only have SCF > 1 if ***P*** is a multiple of 5. Put another way, 4 out of 5 images will have the same image density and every fifth will be 1/5, 1/25th or 1/125th as dense.

The bottom left image is 625 line (5,25,49,J(1,2,5,3,1)). The next ***P*** has SCF = 25, but the next four after that are SCF = 1. The 250 full density images are different but closely related to nearby images (250 = (625-125)/2 since there are 125 multiples of 5 between 1 and 625 and since the images are symmetric about the porcupine midpoint, ***P*** = 312).

The bottom middle image is 3125 line (5,125,151,J(1,1,2,3,5)) has five times the density of the prior image, but the same rule applies. Every 5th ***P*** has lower density, but the rest (1250 in all) are full density. The porcupine is at ***P*** = 1562.

The bottom right is based on 7. The (7,49,33,J(1,1,2,3,5,1,6)) image has 2401 lines (2401 = 7^4). There are 1029 full density images with a reduced density every 7th ***P*** from 1 to porcupine at ***P*** = 1200. Note multiple 7,2 and 7,3-stars here.

Additional points. The above examples should provide you with an idea of how to proceed. Here are additional links and ideas. The first image is based on 11s. The second image is based on 13s. If you violate the *single prime rule* by adding a second prime, it simply means that density changes at multiples of both of these numbers, so you lose density at multiples of 5 and 101 for ***S*** = 101. Finally, if ***P*** is prime and you scroll on ***S***, lines change but stay at full density until ***S*** = ***P***.