Partial Way Around Images: Systematically Encroaching on the *Donut Hole*

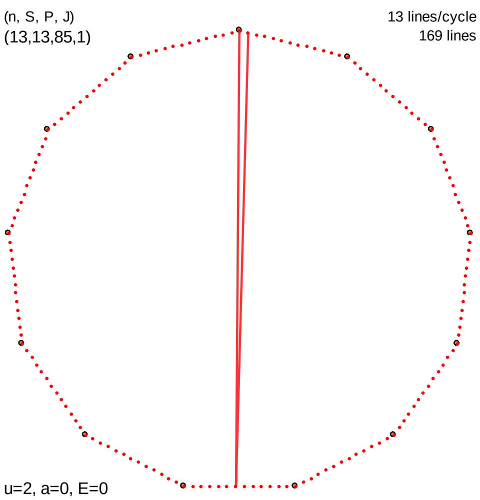
A simple adjustment to [the *no-donut-hole* ***P*** function] (noted below) produces images based on a starting point that is part of the way around the circle rather than near the top of the circle as conceptualized in the *no-donut* model. The adjustment replaces an integer number of times around the circle, ***T***, by a fractional number of times around the circle.

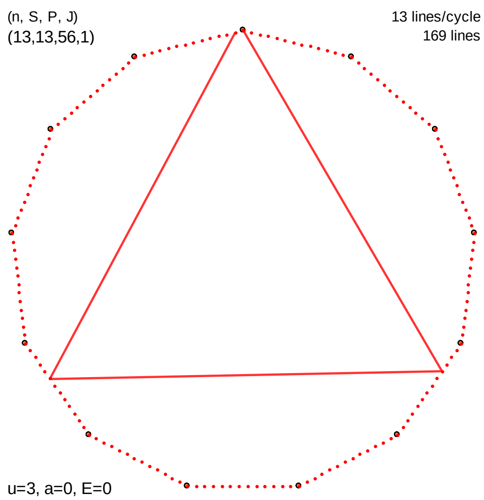
This is accomplished by introducing a new parameter ***u*** ≥ 1 in K39 in the darker blue cells J37:L39, controlled by ⬍ arrows in J37:J39. This parameter is used in conjunction with ***E*** ≥ 0 in L36, controlled by ⬍ arrows in M37:M39.

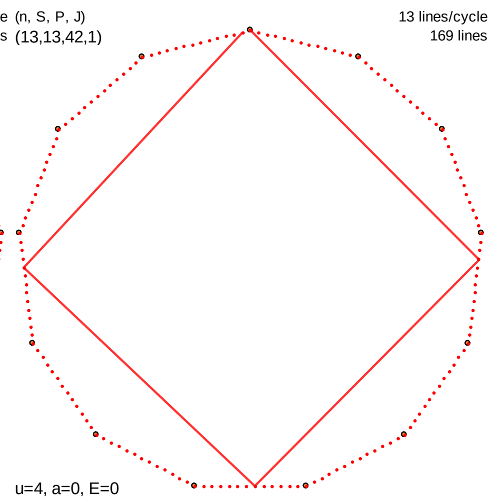
Define *T* and *P* as: $T = E + 1/u$. If $J < n/2$, $P = \text{ROUND}(T{\cdot}S{\cdot}n/J,0) + a$. If $J > n/2$, $P = \text{ROUND}(T{\cdot}S{\cdot}n/(n\text{-}J),0) + a$.

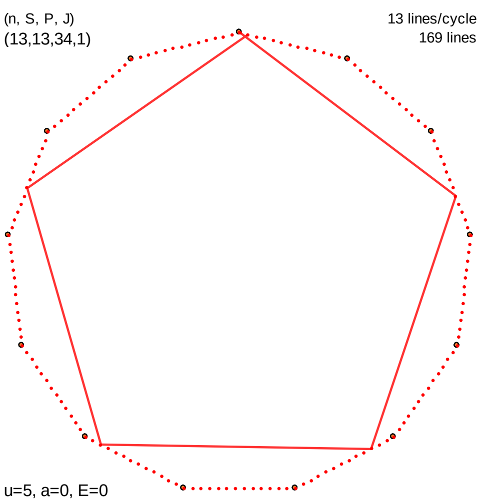
When ***u*** > 1 and ***E*** = 0, this finds ***P*** values that are approximately 1/***u*** of the way around the circle. Below are 6 examples chosen to emphasize the attribute of this version by setting ***J*** = 1. Note that with ***J*** = 1, **every** image other than the ***n***-gon itself encroaches on the *donut hole*. This generalizes the *no-donut* model because ***T*** is an integer if ***u*** = 1.

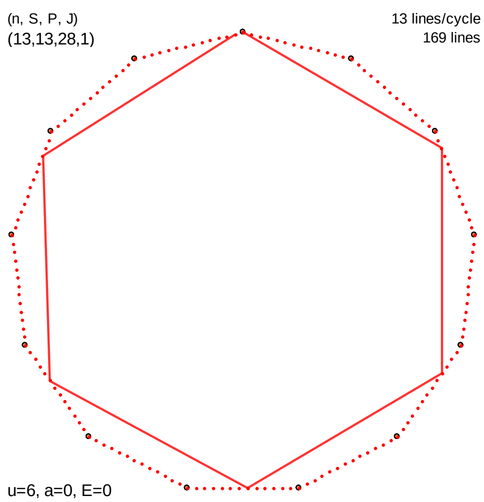
These are [single-step] images created using a single automated equation. This will not always be the case, but it is true for the images below because 13^2-1 = 12·14 = 2^3·3·7 and 13^2+1 = 170 = 2·5·17. This means that 19 single-step ***u***-gons are possible for ***u*** = 2, 3, 4, 5, 6, 7, 8, 10, 12, 14, 17, 21, 24, 28, 34, 42, 56, 84, and 85. This is another example of [difference between squares] at work. Of those shown, ***u*** = 3, 4, 6, and 7 are just under ↺ ***u***-gons; 2 and 5 are just over ↻ ***u***-gons.

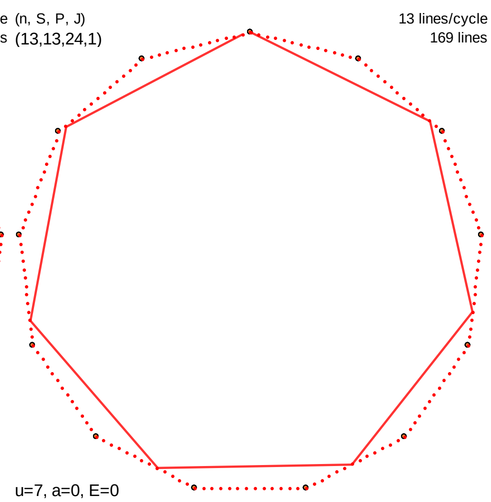








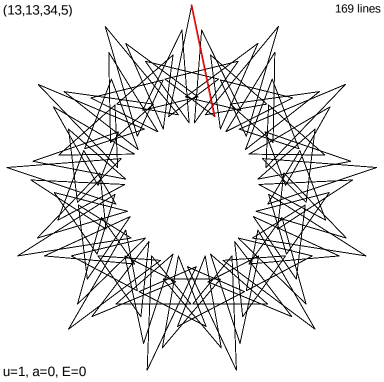




When ***J*** > 1 the fractional nature of the first endpoint is not quite as obvious, but some of the same rules apply as with the *no-donut* model. Note in particular that ***E*** = 0 and ***E*** = ***J*** produce the same image. [**MA.** If we think of ***P*** as a function of ***E***, ***P***(***E***), we see that ***P***(0) = ***P***(***J***) MOD ***n·S***.] Thus, we restrict ***E*** to 0 ≤ ***E*** < ***J***. However, unlike its whole number counterpart (where [***T*** and ***J-T*** produce the same static image]), these images **may be** distinct due to rounding issues.

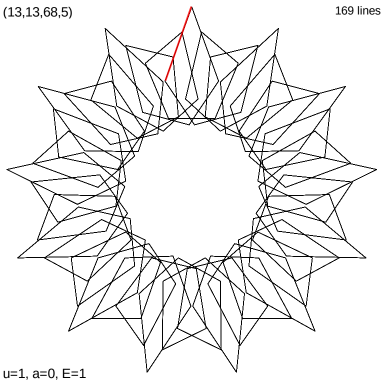
Hint. It helps to tie ***r*** to ***u*** in **Show first *r* lines** (type =K39 in C11) so that the number of lines shown changes as ***u*** changes.

(13,13,34,5) 169 lines



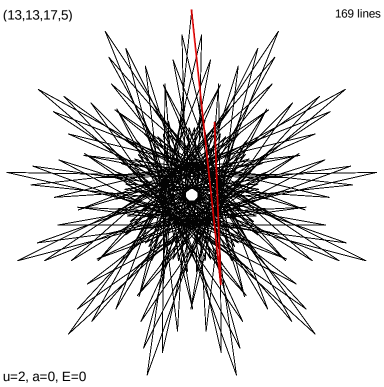
u=1, a=0, E=0

(13,13,68,5) 169 lines



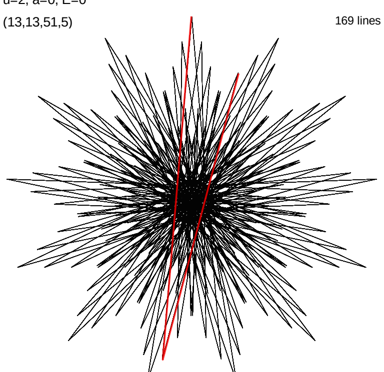
u=1, a=0, E=1

(13,13,17,5) 169 lines



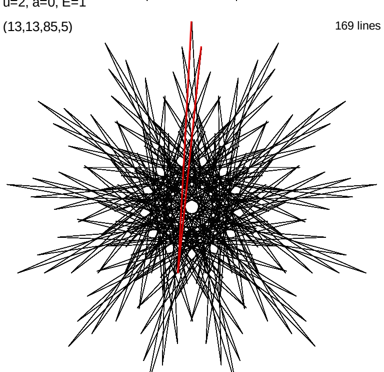
u=2, a=0, E=0

(13,13,51,5) 169 lines



u=2, a=0, E=1

(13,13,85,5) 169 lines



u=2, a=0, E=2

These images are based on ***n*** = ***S*** = 13, ***J*** = 5 and ***a*** = 0 for three values of ***u***.

The left top two images are *no donut-hole* versions given ***u*** = 1 and ***E*** = 0 and 1. The next two values of ***E*** produce the same static images as 1 and 0 but are drawn the other way and ***E*** = 4 produces a single point since ***P*** = 169.

The left bottom three images ***E*** = 0, 1, and 2 are all the static images produced given ***u*** = 2. The next two values of ***E*** produce the same static images as 1 and 0 but are drawn the other way.

The right images, based on ***u*** = 3, produces 5 distinct static images given ***E*** = 0, 1, 2, 3, and 4. If we had space to show ***u*** = 4, the same thing happens: five distinct images emerge via the partial way around automated ***P*** function.

The above is summarized in the table below. Note the highlighted cells which compare ***P*** with ***n·S-P*** since both produce the same static image but are simply drawn in the reverse direction.

	Given ***u*** = 1		Given ***u*** = 2		Given ***u*** = 3		Given ***u*** = 4	
E	***P***	169-***P***	***P***	169-***P***	***P***	169-***P***	***P***	169-***P***
0	34	135	17	152	11	158	8	161
1	68	101	51	118	45	124	42	127
2	101	68	85	84	79	90	76	93
3	135	34	118	51	113	56	110	59
4	169	0	152	17	146	23	144	25

It is worth noting that several of these images are single-step and are quite enjoyable to watch drawn using *Fixed Count Line Drawing* mode with varying *Drawn Lines*, *DL*, as noted with each image.

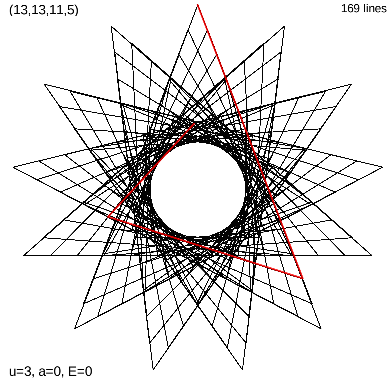
The top left is a swirling pentagram (set *DL* = 5).

The "snowflake" image at bottom left is completed in 13 cycles that look like a lopsided chopstick rolling around a circle. Set *DL* = 2 for best viewing.

The second from bottom at right appears to be a one-time around counterclockwise-drawn image that is similar to *Three Shape-Shifting Triangles* but not as complex. Actually, the first cycle ends at vertex 6 (set *DL* = 13 to see) but the second cycle ends at vertex 12 (set *DL* = 26 to see) and thus the illusion is created. Set *DL* = 3 for best viewing.

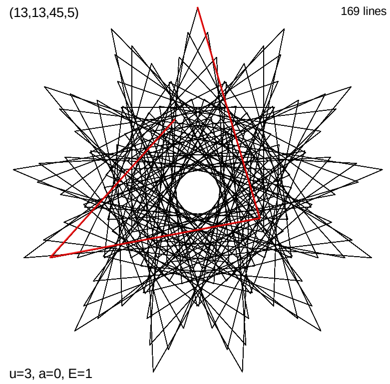
Finally, among the ***u*** = 4 images noted in the table, there is a single-step rotating quadrangle at ***P* = 42**. Set *DL* = 4 for best viewing.

(13,13,11,5) 169 lines



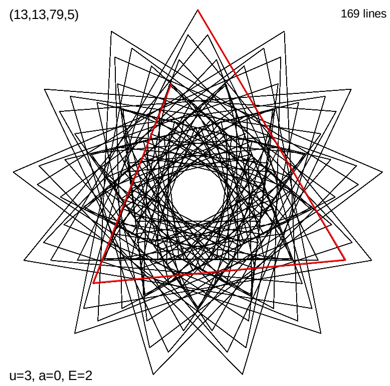
u=3, a=0, E=0

(13,13,45,5) 169 lines



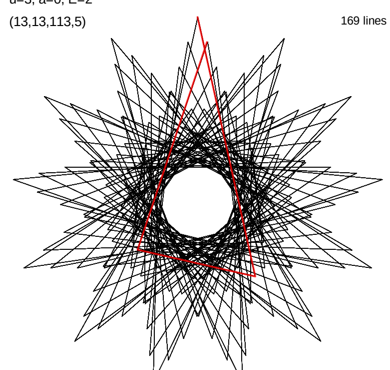
u=3, a=0, E=1

(13,13,79,5) 169 lines



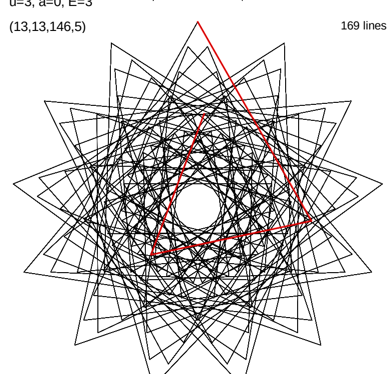
u=3, a=0, E=2

(13,13,113,5) 169 lines



u=3, a=0, E=3

(13,13,146,5) 169 lines



u=3, a=0, E=4