NON-TESSELLATING REGULAR POLYGONAL GRIDS

MOOSA NASIR
OCTOBER 29, 2022

ABSTRACT. All known regular polygonal grid implementations are tessellating and either use triangles, squares, or hexagons. I produce a new grid system idea that uses non-tessellating polygons instead. **Keywords.** Grids, Non-tessellating, Traversal

A regular polygon that tessellates in a grid must follow the rule that the sum of the interior angles where the vertices of the polygons meet equals 360°. As shown in figure 1, the only regular polygons that follow this rule are triangles, squares, and hexagons, while shapes like pentagons or heptagons are shown to not be able to be tessellated. A note going forward, most examples will be done using pentagons but the information presented should work with any regular polygon.

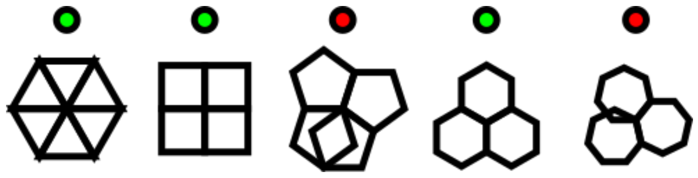

FIGURE 1. Examples of what regular polygons tessellate

Nothing stops us from creating a grid using non-tessellating polygons though. It makes sense that this idea would work, as every polygon on these grids would still be connected to n polygons where n is the number of sides polygon has. Further more after analyzing these grids, the polygons will be able to connect back to a chosen starting point as shown in figure 2, meaning grid traversal is possible. Visualization of these grids might pose a problem due to their complexity and intersection. Non-tessellating grid start off with one polygon, which will be the origin, and at every iteration, a new polygon is connected to an empty side, as shown in figure 3. Eventually connections will be formed between the overlapping polygons, allowing for certain paths to reach a point on the grid with less traversal. To traverse these grids we need a new format. This format will provide the same purpose as a point on a Cartesian plane, to show a path that traverses from the origin of the grid to another position on the plane. For example, if a point of (35, 54) is given, this means that you must traverse 35 squares right, and 54 squares up from the origin to get to the desired point.

Lets start off with a regular polygon. It has n sides and we can choose one of those sides to start our traversal. It will help to label these sides from 1 to n. If not noticed already, one can see that to connect regular polygons at their sides, a 180° rotation must be made to the polygon we want to connect. This makes things simpler, as each rotation can be represented by a 0 or 1, where 0 represents no rotation, and 1 represents a 180° rotation. When we

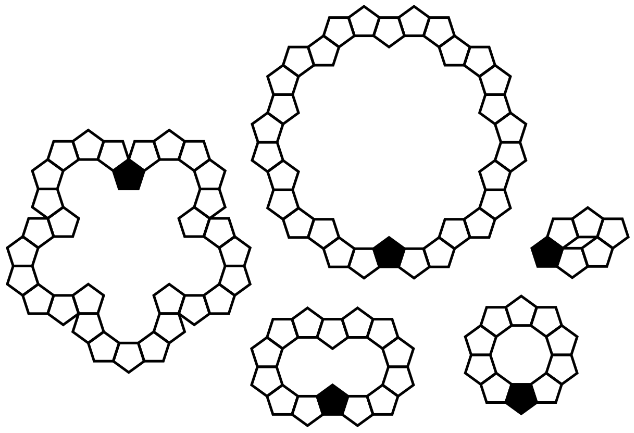

FIGURE 2. Examples of pentagons connected together from a chosen starting point, highlighted in black

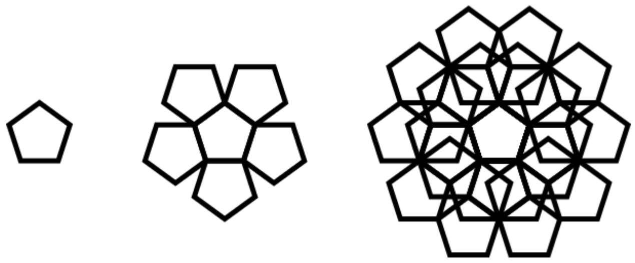

FIGURE 3. Three iterations of a pentagonal non-tessellating grid

rotate the polygon, the numbers on each side stay with them, This makes sure that our next traversal will not result in moving back to our previous position if the side number we chose to traverse to is not equal to the previous. After Looking at figure 4, the format would work nicely to show the starting rotation of the the origin, the number of sides on the polygon, and the list of side numbers which we traversed by. So in general, the format of a traversal on a non-tessellating regular polygonal grid would be:

$$\overset{r}{\underset{n}{}} s_1 \to s_2 \to s_3 \to \cdots s_{i-2} \to s_{i-1} \to s_i$$

Where r is the rotation of the polygon (0 or 1), n is the number of sides in the polygon, s is the side number traversed to, and i is the number of side traversals. We can see an example of this in Figure 4, where it shows the visualization of the traversal $\overset{0}{\underset{5}{}} 1 \to 3 \to 2 \to 5 \to 3$. This can be said as "A traversal of pentagons where the origin has no rotation and traverses from side 1 to side 3 to side 2 to side 5 to side 3".

This format looks to be the best way to describe a traversal on these grids, and allows for many patterns and general algorithms to be figured out. Most algorithms presented below have the goal of shortening the number of needed traversals. A simple algorithm is that if two of the same numbers are adjacent in the list of traversed sides numbers, they cancel each other out as shown in figure 5. This is because when we traverse the two same side numbers,

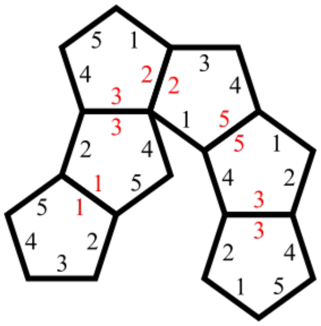

FIGURE 4. An example traversal of ${}^{0}_{5}1 \to 3 \to 2 \to 5 \to 3$

the first traversal goes to a new polygon, but the second traversal goes back to the previous polygon due to the rotation in the new polygon reversing the side numbers.

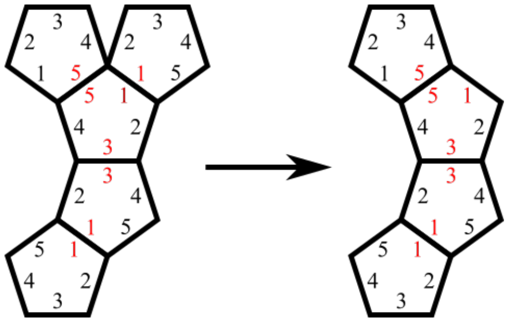

FIGURE 5. The traversal ${}^{0}_{5}1 \to 3 \to 1 \to 1 \to 5$ is shown equal to the traversal ${}^{0}_{5}1 \to 3 \to 5$

This is a good way of decreasing the number of side number traversals but an important algorithm in this non-tessellating grid system would be a way to find the most efficient path of a traversal. This will allow for us to know the smallest amount of side traversals needed to get to a certain polygon from the origin. After some experimentation, we can see that most simple traversals can be simplified by checking some criteria. But first, we need to notice a something. Though only the origin is given a rotation value, each side number traversal has its own rotation as well. When mapping these rotations, we can see that a traversal like ${}^{0}_{5}1 \to 3 \to 2 \to 5$ for example becomes ${}^{0}_{5}1 \to^{1} 3 \to^{0} 2 \to^{1} 5$ where the superscript to the left of the side numbers represent the rotation of the polygon at a certain point in the traversal. Knowing this we can formulate an algorithm. When two similar ordered subsets are found in a list of side numbers of a traversal, and they both start at opposite rotations, we can cancel out both subsets and reverse the ordered subset between them to get the most efficient traversal. For example, if we have the traversal ${}^{0}_{5}5 \to 3 \to 1 \to 2 \to 3 \to 5 \to 3 \to 1$ we can simplify like so:

$$\overset{\boxed{0}}{\underset{5}{}}\underline{5 \to {}^{1}3 \to {}^{0}1} \to {}^{1}2 \to {}^{0}3 \to \underline{{}^{\boxed{1}}5 \to {}^{0}3 \to {}^{1}1} =$$

$${}^{0}_{5}\cancel{5 \to {}^{1}3 \to {}^{0}}1 \to {}^{1}2 \to {}^{0}3 \to \cancel{{}^{1}5 \to {}^{0}3 \to {}^{1}}1 =$$

$${}^{0}_{5}reverse({}^{1}2 \to {}^{0}3) =$$

$${}^{0}_{5}3 \to {}^{1}2$$

We get that the most efficient path to the end polygon location in our original traversal is ${}^{0}_{5}3 \to 2$. This is visualized in figure 6 and proves the algorithm works. Something to note though is that these traversals can get very complex, and might not always have both ordered subsets of side numbers at the ends of the set. In that case, you would check to see if a certain subset meets the requirements of this algorithm and if so, solve for the reversed middle subset and include that with the rest of the set. This is pretty important for this algorithm to work.

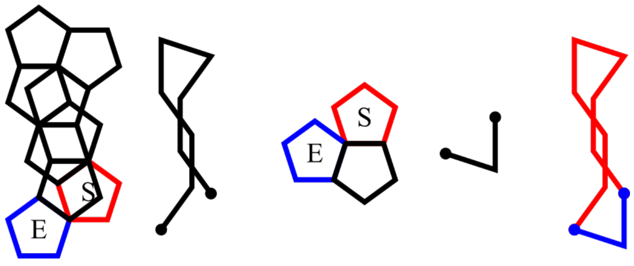

FIGURE 6. The traversal ${}^{0}_{5}5 \to 3 \to 1 \to 2 \to 3 \to 5 \to 3 \to 1$ is shown equal to the traversal ${}^{0}_{5}3 \to 2$ as they end in the same position. S represents the starting point and E represents the ending point.

Another thing that is important is converting these traversals to a Cartesian coordinate. This will be helpful when trying to create visualizations and drawings of a traversal. The way we will do this is by treating the traversals as vectors, and then getting the components of the vector sum. A few variables will be needed for this algorithm, including the interior angle between two vertices in the polygon, θ, the polygons side length, s, and the length of the polygons apothem, a. The interior angle θ is equal to $\frac{360°}{n}$ where n is the number of sides in the polygon. The length of the polygons side is just used in case we want to scale the visualization of the polygons and the length of the polygons apothem is $a = \frac{s}{2tan(\frac{180}{n})}$. The formula's for this conversion would be:

$$s_{Tx} = \Big((1-2r_1) \times (2a)sin(\theta s_1 - \frac{\theta}{2})\Big) + \Big((1-2r_2) \times (2a)sin(\theta s_2 - \frac{\theta}{2})\Big) +$$
$$\Big((1-2r_3) \times (2a)sin(\theta s_3 - \frac{\theta}{2})\Big) + \cdots + \Big((1-2r_{i-2}) \times (2a)sin(\theta s_{i-2} - \frac{\theta}{2})\Big) +$$
$$\Big((1-2r_{i-1}) \times (2a)sin(\theta s_{i-1} - \frac{\theta}{2})\Big) + \Big((1-2r_i) \times (2a)sin(\theta s_i - \frac{\theta}{2})\Big)$$

Where s_{Tx} represents the x-component of the sum of side vectors, r_i represents the rotation at traversal i, s_i represents the side number at traversal i, and i represents the number of side number traversals in the whole traversal. For the y-component of the sum of side vectors we just change sin in the equation to cos. Our point on the Cartesian plane is (s_{Tx}, s_{Ty}). Note that if r is equal to 0 then r_i just goes $0, 1, 0, 1, 0 \ldots 1, 0, 1$.

For example, if we have the traversal ${}^{0}_{5}\,1 \rightarrow 4$ and our pentagon has a side length of 1, and apothem of 0.688, then s_{Tx} will equal 2.12 and s_{Ty} will equal 1.54 as shown in figure 7.

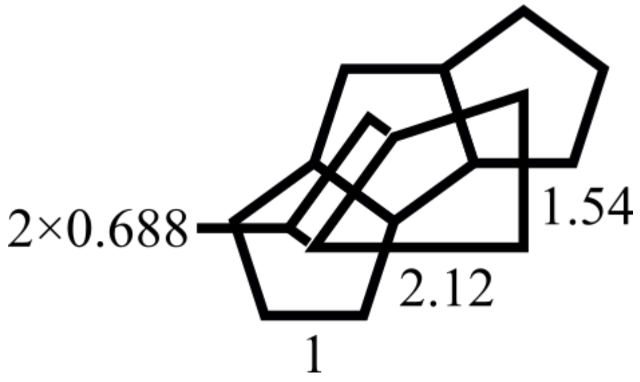

FIGURE 7. Graphical proof that ${}^{0}_{5}\,1 \rightarrow 4$ and (2.12, 1.54) both end in the same location

Another thing that would be possible is the use of iterative functions. For its format, It would look like this:

$${}^{r}_{n}\,S_n = \{...\}\; where\; S_0 = s_1$$

Where r is the rotation of the S_0, n is the polygons number of sides, S_n is the side number of the nth traversal, and s_1 is the first side number traversal. If at any point S_n becomes greater than the number of sides on the polygon or less than 0, we apply a modulus operation on the number of S_n. A possible function could be for example, ${}^{0}_{5}\,S_n = S_{n-1} + 2\; where\; S_0 = 2$. Figure 8 shows how this function will look.

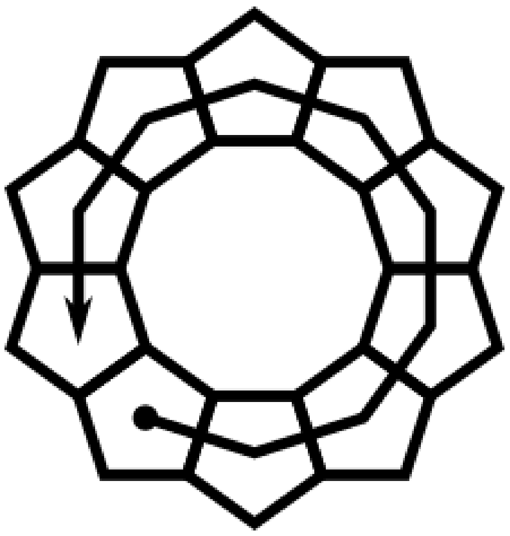

FIGURE 8. Function of ${}^{0}_{5}\,S_n = S_{n-1} + 2\; where\; S_0 = 2$

Going back to figure 3, we can see that a few sequences exist. For the case of the non-tessellating regular pentagonal grid, the total number of pentagon each iteration would be the sequence A020989 if we count every pentagon, and it would be the sequence A175898 if we count only unique pentagons. A pattern certainly can be found that explains the sequence for every non-tessellating polygon. Computing all unique points up to a certain iteration would be $O(2^n)$. Figure 9 shows all point up to iteration 9. We use points instead of pentagons to better visualize without intersection. You can see that in the middle a pentagon like shape can somewhat be seen, but as it gets to the outside, a decagon is formed. Circular patterns are also seen to be formed inside the points.

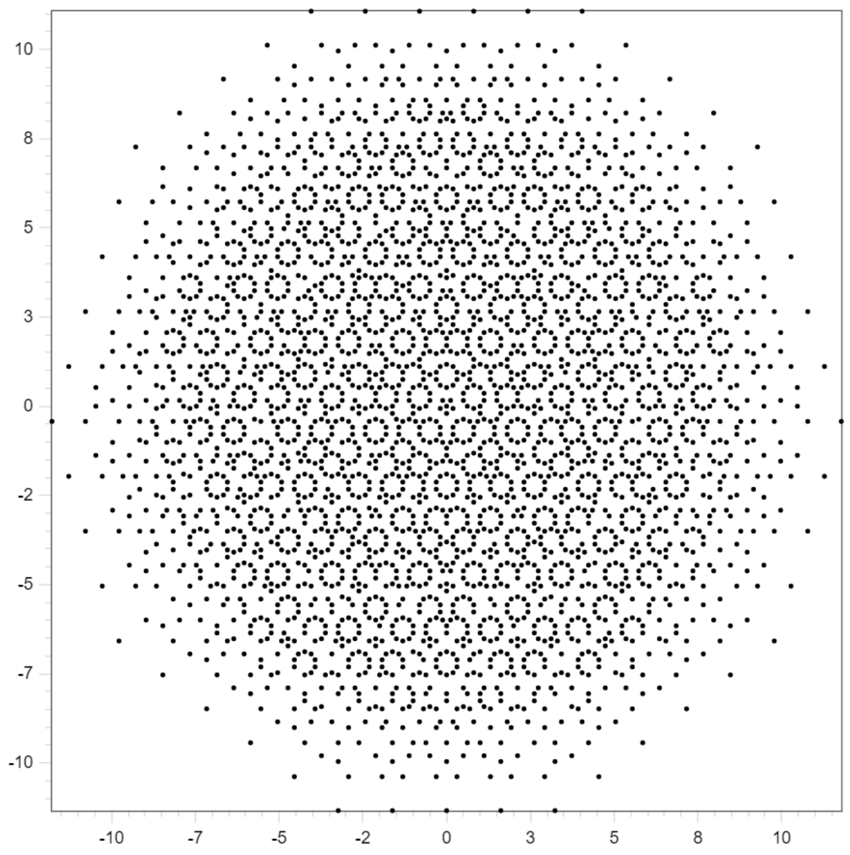

FIGURE 9. Plotting mid points of pentagons in a non-tessellating pentagonal grid up to iteration 9

In conclusion, non-tessellating regular polygonal grids are a new discovery with many unique properties no found in their tessellating counterparts. More must be done to fully understand how these grids work. A simple implementation of algorithms in this document can be found on GitHub at https://github.com/TealEgg/non-tessellating-regular-polygonal-grids.