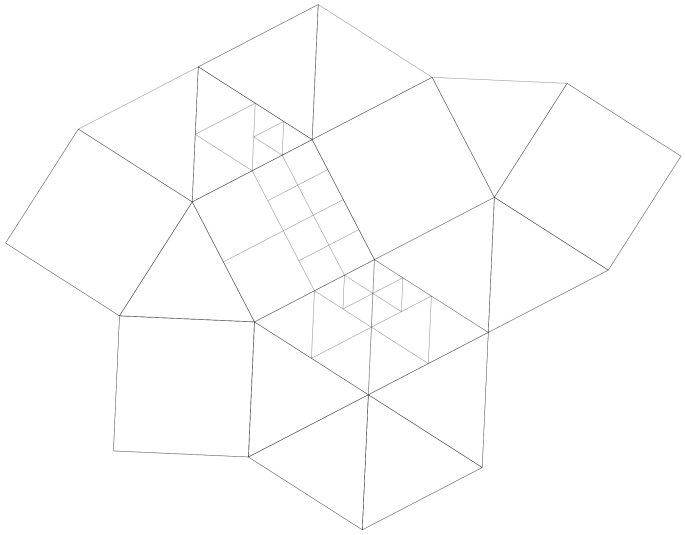


Contents

1 Overview

2 Domain Tiles Generation [Geometry Processing and Space Indexing]

2.1 Square Triangle Tilings

Square-triangle tilings allow for more varied geometry and therefore player freedom in a sandbox context, when compared to all square or all triangle regular tilings. This variety however comes with challenges both in generating the actual tiling and in the impossibility of random-accessing the single elements.

The rules ensuring a proper plane-filling tiling can be explained by looking at its vertices: squares account for 3/12 of a 360° angle, while equilateral triangles account for 2/12. This forces the set of faces surrounding a vertex to always

contain an even number of squares (i. e. 0, 2 or 4), with the rest made up by triangles.

The most straightforward way to attempt at generating such a tiling would be to use a Wave-Function-Collapse-like (WFC) algorithm, starting from a single tile and then propagating the generation past the edges, while using as rules the vertex completion right above. This method however has a few problems when we intend for parts of the world to be lazy-initialized, for instance generating a section of the map only when the player comes in close enough proximity. Heading far away from the spawn tile and then returning there from a different direction would generate a loop of collapsed tiles whose interior is a region that is very unlikely to admit a closable tiling, i. e. there may be regions of the map that can only be covered by tiles that intersect ones previously generated. A second issue with a WFC method is that linking tile collapse to player exploration results in a non-deterministic world generation, which may unnecessarily make other parts of the project less straighforward.

These problems are shared by any method attempting to expand the tiling past the border of previously determined tiles, so a flip in the overall approach is required.

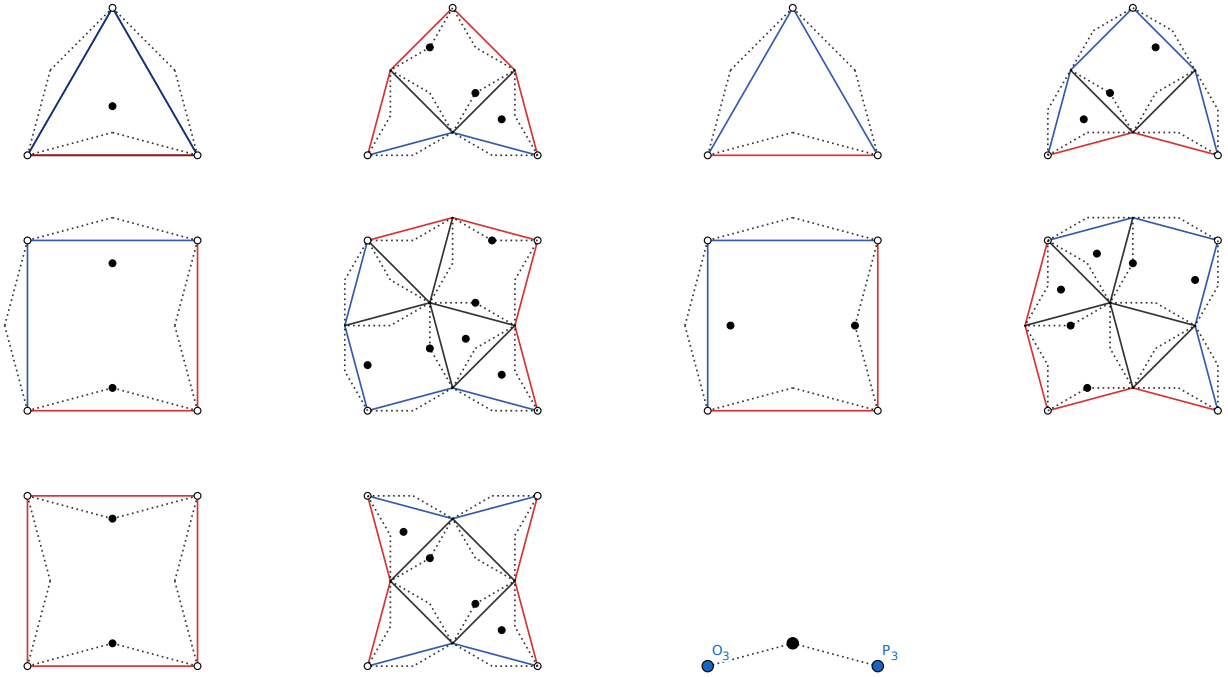
To ensure compatibility with lazy initialization and deterministic world generation, subdivision techniques provide a better approach: if we start with one "world tile" (either a square or a triangle) and a set of rules on how to subdivide each polygon as a collection of smaller polygons, ensuring that correct adjacency between tiles carries on to the next iteration, it is possible to apply those rules to the first tile, then to each of those it gets subdivided into, and so on until an arbitrary degree of subdivision is reached.

The power of this method relies on the fact that if we are able to keep track of successive subdivisions in a "subdivision tree", then it is not necessary to completely subdivide the whole world at once: if the player approaches areas of the world where the subdivision hasn't yet reached its maximal degree, it is sufficient to apply subdivision to an intermidiate node in the tree such that only regions in an immediate neighbourhood of the player get completely subdivided. This method thus allows for an adaptive resolution of the tilings that leaves areas far away from the player only partially subdivided, affording significant savings in memory occupancy and processing power.

2.2 Inflation Techniques [The Method]

Inflation techniques are subdivision algorithms that turn a given tiling into another with a higher resolution. Their name comes from the fact that each subdivision step is usually preceeded by an "inflation", i. e. a dilation of the

Figure 2.1: Inflation Rule



plane, by a constant value (the "inflation constant") that is intrinsic to the particular technique being used. The reason for such inflation is that it is usually desirable to ensure that parent and child tilings share the same edge-length. The inflation step however will never be performed in this project, since subdivision will be done "in place", so that vertices in the parent tiling are ensured to be vertices in the child as well.

The inflation technique used by this project, which has an inflation constant of $\gamma = \sqrt{2+\sqrt{3}}$, is an adaptation of the one found in [ZeUn06], which sadly doesn't actually work. Without diving into too much detail, the reason why the original technique outlined by the paper doesn't work is that, as written, adjacency between certain tiles is broken after the first full subdivision, making it impossible to arbitrarily repeat the process. My contribution consists in the introduction of a new family of triangles (which I call "true triangles"), a new property for edges (their coloring can be either "true" or "false") and in devising an update to each subdivision rule in order to account for the introduction of such additions. All of this will make sense as soon as you're through reading the actual subdivision rules detailed below.

To know how each polygon is subdivided it is necessary to distinguish two

different categories of triangles and of squares. Both categories of triangles are made up of two blue edges and one red edge, while squares are separated into the category of squares with four red edges ("red squares") and that of squares with two blue and two red edges ("red-blue squares"). Keeping figure 2.1 as reference, we can see that the color of the edges reflects the particular way they are split by performing a subdivision step: red edges turn into two edges sharing a vertex located in the interior of the parent polygon, while the corresponding vertex for blue edges falls outside the parent tile. Coloring the edges is thus a way of knowing how they will be subdivided, and each two adjacent tiles must share edges of opposite color, so that adjacency can carry on to their children.

We've seen how subdivision turns each edge into a pair of smaller edges sharing a displaced midpoint, but to know how these edges have to be colored (to be able to perform a new subdivision step) it is necessary to have a little more information. We will call "true" edges those which turn into edges of the same color and "false" edges those which swap colors when subdivided. We can see that to ensure correct color-pairing to the children edges, true edges must be paired with true edges between adjacent faces and the same applies to false edges. In summary, all of this means that each of a tile's edges has two properties, and between adjacent tiles a true blue edge must be paired to a true red edge, while a false red to a false blue.

The two categories of triangles mentioned above are determined by this second property of their edges: we'll call "true" triangles those whose edges are all true edges and "false" triangles those who only have false edges. In the figures showing each polygon's subdivision, false triangles are identified by a black spot in the middle, while for squares, their false edges are denoted by a black spot right next to that edge. Clearly, true triangles can only be adjacent to either other true triangles or any kind of square, but only through one of the square's true edges. At the same time, the exact opposite applies to false triangles. Figure [insert picture] shows how this rule induces, onto the tiling, two distinct graphs, whose nodes are the center point of the triangular tiles, and whose edges cross squares from side to side in accordance to their "trueness".

References

[ZeUn06] Xiangbing Zeng, Goran Ungar. Inflation rules of square-triangle tilings: from approximants to dodecagonal liquid quasicrystals. Philo-

sophical Magazine, Taylor & Francis, 2006, 86 (06-08), pp.1093- 1103.
ff10.1080/14786430500363148ff. ffhal-00513624f