Quasi.py: An Investigation into Tessellating Three-Space with Quasicrystals

Michael J. Mangialardi
Math 198: Hypergraphics 2006
University of Illinois at Urbana-Champaign

Abstract

The purpose of this exercise is to create a program that generates an aperiodic lattice of three-dimensional quasicrystals. This was accomplished using the DeBruijn dual method, which involves taking a star of vectors, each with direction normal to a face of a dodecahedron, and drawing planes normal to those vectors on unit intervals away from the origin. Then, at each point where three or more planes intersect, transport the original vectors and add them, and a quasicrystal is formed. By cycling through all points of intersection, an aperiodic lattice is formed.

Background

In 1974, mathematician and physicist Robert Penrose discovered that he could tessellate a two dimensional space quasiperiodically with two figures, a fat rhombus and a skinny one. This tessellation is called quasiperiodic because the pattern has no translational symmetry, yet it has five-fold rotational symmetry, something which was previously not thought to exist for tessellations. Penrose soon discovered a challenge in creating these tessellations: describing a foolproof rule set which does just that. He attempted to do so, along with the reclusive mathematician Robert Ammann, by decorating the tiles with bars that form continuous lines when the pattern was properly assembled. However, this method was not perfect, as it can come to pass that legal moves produce an error and must be replaced.

Then, in 1980, Dutch mathematician Nicolas DeBruijn came up with two methods of repeated steps that generate these non-repeating patterns. The first of these is the projection method, in which the vertices of a higher dimensional lattice of cubes is put through a mathematical test and those that pass are projected into lower-dimensional space and become the nodes of the quasicrystals. DeBruijn's second method, the dual method is the superior of the two and goes as follows. First, define a star of vectors that a perpendicular to the sides of a dodecahedron. Then, construct planes at unit intervals along the vectors. At each point where three planes intersect, transport the original vectors and perform the vector addition of the vectors, and either a fat or a skinny rhombohedron, analogous to the fat and skinny rhombi of Penrose's two dimensional tessellation, is drawn. If this is performed at every location where three planes intersect, a quasiperiodic lattice is formed.

Method

This program uses the DeBruijn dual method to generate the desired quasiperiodic lattice of rhombohedra. This method involves taking a star of vectors,

each with direction normal to a face of a dodecahedron, and drawing planes normal to those vectors on unit intervals away from the origin. Then, at each point where three or more planes intersect, transport the original vectors and add them, and a quasicrystal is formed. By cycling through all points of intersection, an aperiodic lattice is formed.

This program makes use of the syzygy library and the distributed scene-graph framework. This framework works by establishing a tree in which every node has the properties of its parent nodes and passes on its traits to its daughters; nodes correspond to things such as materials, indices, drawables, transforms, and other things necessary for computer graphics. What follows is a line-by-line explanation of how that method was implemented in quasi.py in order to draw that lattice.

Lines 1-5 import the syzygy library, as well as other libraries that are necessary for use in the program. Lines 6-32 are the default navigator as defined by Ben Schaeffer in his Cosmos.py. These lines define how the user navigates within the program.

Lines 33-73 are where the drawQuasiCrystal method is defined. This method, as its name implies, draws a quasicrystal. It takes as arguments the three vectors that will define the generated solid, the x, y, and z coordinates of the "zero" vertex of the quasicrystal, and the material node that will become the parent of the quasicrystal. This method first creates a node to function as the set of points defining the vertices of the rhombohedron to be drawn as a daughter of the material node passed as an argument. Then it creates a series of index nodes as daughters of the points node which will tell their corresponding daughter drawable nodes the order in which to connect the vertices. The method then creates a drawable node for every index node and defines those drawable nodes to be line strips, drawing the quasicrystal.

Lines 74-149 define the calculatePosition method, given three vectors, calculates the positions of intersection of the planes defined by those vectors, which give the positions of the rhombohedra. This method takes three vectors and the material node that will be parent to the drawn rhombohedra as arguments. Because each vector defines several planes, each spaced $N\tau$ away from the origin, where N is a non-zero integer and τ is the golden ratio, the first thing that this method does is create a set of nested while loops so that each permutation of planes can be accounted for. Then, for each plane, it generates its equation in the form:

$$Px + Qy + Rz = C$$

where:

$$C = Px_0 + Qy_0 + Rz_0$$

and P, Q, and R are the x, y, and z components of the vector normal to the plane. Then, Cramer's rule is used to determine if and where the planes intersect. If the determinant of the coefficients matrix is equal to zero, then the planes do not intersect and the method moves on and calculates the position of the rhombohedron and draws it using the drawQuasiCrystal method defined above.

Lines 150-171 define the addLights method. This method takes the root node as an argument and creates two white lights as daughters to the root node.

Lines 172-179 set up the distributed scene-graph framework, set the root node, and set buffer swap to manual. Lines 180-197 translate everything so that it is in front of the viewer, set the navigation node, call the addLights method, and create a material node and set the material properties of the rhombohedra. Lines 198-207 define the star of vectors that will be used to

calculate the position of the rhombohedra. Lines 208-249 call the calculatePosition method for each possible combination of three vectors and output a countdown so that the user knows that the program has not stalled during the lengthy process of calculating the positions. Finally, lines 250-259 establish the main loop of the program, swap the buffers, update the navigator, and set the viewer.

Execution

To run the program, syzygy version 1.0, Python version 2.4, and a UNIX shell must be installed Then, complete the following steps:

1. Open the UNIX shell.
2. Navigate to the directory in which syzygy is installed.
3. Start syzygy.
4. Navigate to the directory in which quasi.py is located.
5. Type `python quasi.py` into the UNIX shell

Conclusion

The greatest challenge in writing quasi.py is the fine tuning of the mathematics to generate the rhombohedra. There are several upgrades planned for future versions of quasi.py First, a proper algorithm for selecting where to draw rhombohedra needs to be developed because the current one, where the only criterion to be met is that the determinant of the coefficients matrix cannot be equal to zero, is too permissive. Second, it would be nice to draw the rhombohedra one at a time so that the user can see the structure grow and evolve. Finally, it has been proposed to allow the user to select rhombohedra one at a time and toggle whether the selected rhombohedron is viewed as a wire frame, as a solid, or as a partial solid.

I would like to thank George Francis and Tony Robbin for their help in developing the algorithm to calculate the position of the rhombohedra. I would also like to thank Ryan Mulligan and Kyle Wilkinson for helping to debug the code.

Sources

1. Robbin, Tony. *Shadows of Reality*. New Haven, Connecticut: Yale University Press, 2006.
2. Levine, Dov et al. "Quasicrystals with Arbitrary Orientational Symmetry," *Physical Review*. 32(8):5547-50, October 1985.