Voronoi Builder Plugin for CamBam

[Version 1.0.7a]

Purpose

This plugin provides a tool to create Voronoi patterns inside closed shapes. These patterns might then be used for creating textures or profiles for machining. A Voronoi diagram is defined as (Wikipedia):

> *"In mathematics, a **Voronoi diagram** is a partitioning of a plane into regions based on distance to points in a specific subset of the plane. That set of points (called seeds, sites, or generators) is specified beforehand, and for each seed there is a corresponding region consisting of all points closer to that seed than to any other. These regions are called Voronoi cells."*

The dual to the Voronoi diagram is the Delaunay triangulation of the seed points; this diagram can also be created.

This plugin was inspired by a previous Voronoi Script (by Bill Trondsen) and earlier work by Bob Mackay that had a similar purpose. This plugin is built on two third party (open source) packages:

- Voronoi creation by Burhan Joukhadar (www.codeproject.com/Tips/797123/Fast-Voronoi-Diagram-in-Csharp), a C# library for creating Voronio diagrams.
- Polygon clipping from Clipper by Angus Johnson (www.angusj.com), an efficient polygon clipping package.

It is also possible create 2.5D surfaces based on the Voronoi diagram, as a textured surface with various options for surface forms. These are created as image bitmaps that can be saved and imported to generate a height map surface in CamBam.

Installation

The VoronoiBuilder.dll file must be placed inside the CamBam plugins folder, and CamBam restarted. The Voronoi Builder option will then appear in the CamBam <Plugins> menu.

Operation

The Voronoi diagram is constructed inside closed shapes, these may be:

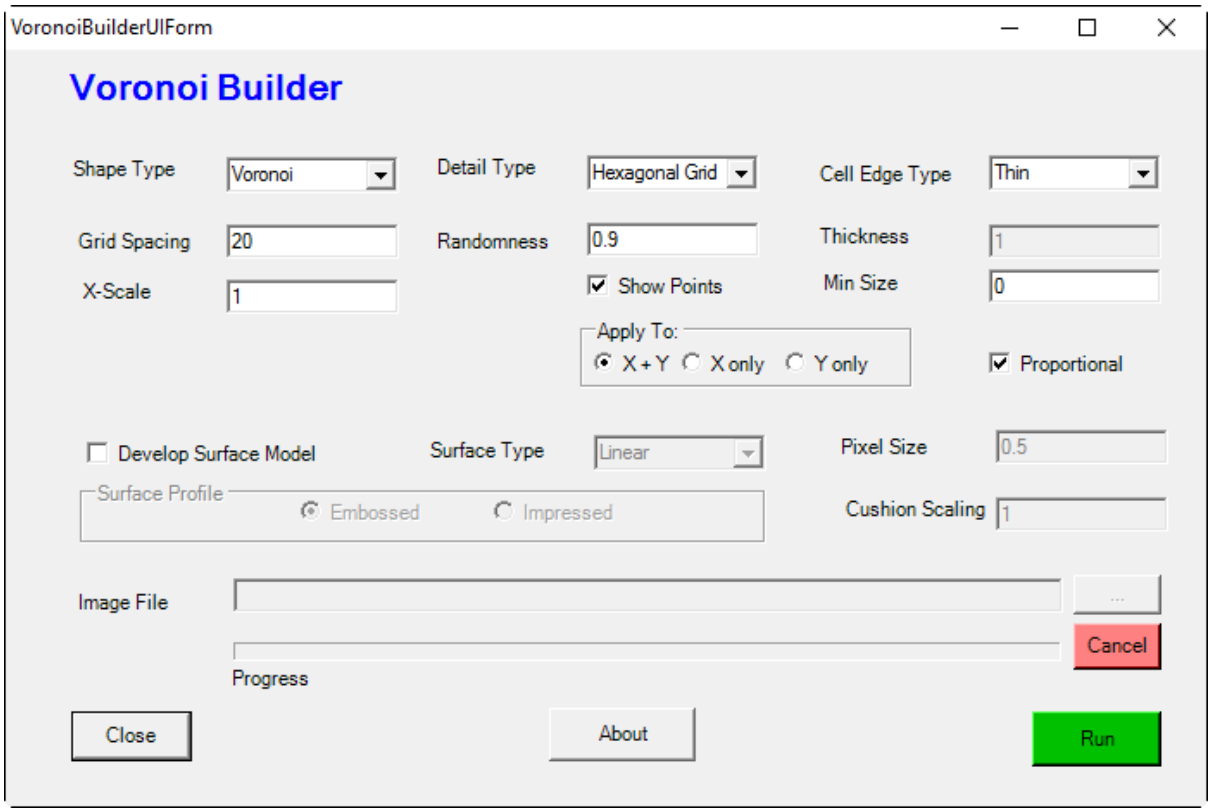
- Polylines
- PolyRectangles
- Circles
- Regions.

and an optional set of seed points manually defined as a PointList. If no seed points are defined, the plugin offers several types of computed sets of seeds: random, a square grid, a hexagonal grid or a triangular grid each with optional random perturbations.

A boundary that has a hole it must be converted to a Region. The bounding shapes, and any manually defined seed points, must be selected before launching the plugin.

The resulting diagram will be placed in a new Layer (named “Voronoi Layer”). For a Voronoi diagram layer there will be one closed Polyline for each Voronoi cell; for a Delaunay diagram there will be one open Polyline for each line in the diagram.

Here is the plugin dialog:



The Options are:

1. <**Shape Type**> selects either a Voronoi or Delaunay diagram.
2. <**Detail Type**>: specifies the nature of the seed points used to construct the Voronoi diagram, the options are:
    a. <**Random**>: uses a random pattern of seed points at a specified average spacing.
    b. <**Square Grid**>: uses a square grid of seed points at a specified spacing with a level of randomness.
    c. <**Hexagonal Grid**>: uses a triangular shaped grid at a specified spacing with level of randomness to create hexagonal shapes.
    d. <**Triangular Grid**>: uses a hexagonal shaped grid at a specified spacing with level of randomness to create triangular shapes.
    e. <**Diamond Grid**>: uses a diamond shaped grid at a specified spacing with a level of randomness to create diamond shapes.
    f. <**Octagonal Grid**>: uses an octagonal shaped grid at a specified spacing with a level of randomness to create octagonal shapes.
    g. <**Manual**>: uses a set of points defined, and selected, as a PointList by the user. The points should “cover” region of the boundary elements and extend outside the boundary to ensure that no artificial edges appear inside the boundaries.
3. <**Grid Spacing**>: the required spacing of seed points. For Random and Manual type seed points this value is computed from the *average* spacing of the random or manually defined seed points.

4. <**Randomness**>: the required randomness applied to the defined grid as a fraction of the grid spacing (values in the range 0 to 1.0 are useful).
5. **<Apply To>**: the randomness can be applied to both axes (the default), or just the X or Y axis as selected from the radio buttons.
6. **<Proportional>**: the randomness is applied in proportion to the distance from the point representing the minimum-X and minimum-Y values of the boundary shapes. By default the same randomness is applied to all points.
7. <**Cell Edge Type**>: has two options (for Voronoi diagram only):
    a. <**Thin**>: the edges of the cells are a single line, i.e. the edges of the Voronoi cells are coincident.
    b. <**Thick**>: there is a gap between the cells.
8. <**Thickness**> is the required thickness of the gap between the Voronoi (only) cells. Note that the gap between the Voronoi cells and the original bounding objects is 50% of this value.
9. <**Min Size**>: is a dimension parameter to filter out very small cells (perhaps too small to be machined). Currently this is the square root of the area of the cells. This seems to work reasonably well, but requires some experimentation, and may not be perfect for all situations. Manual removal of small cells may still be required.
10. <**X Scale**>: defines an X-axis scaling actor to distort the grid (see later Notes & Tips). Note that the results of such scaling are not easily predictable.
11. <**Show Points**> will add the computed seed points to the output layer as a PointList (for interest, and for possible re-use to repeatedly generate the same pattern).

The following controls allow the creation of a bitmap image to represent the Voronoi cells as a 2.5D textured surface

12. <**Develop Surface Model**> checkbox enables this option (only for Voronoi models)
13. <**Surface Type**> combo selects the type of surface required, the options are:
    a. <**Linear**>: each Voronoi cell is composed of a set of flat surfaces, with the seed point being the top. The surface slopes from the seed to the cell edges.
    b. <**Quadratic**>: each Voronio cell is defined as a quadratic surface with the seed point being at the top. The surface slopes from the seed to the cell edges.
    c. <**Cushion (Quad)**>: each Voronio cell is a quadratic surface, now symmetrical about the seed, so the surface elements can intersect above the cell edges.
    d. <**Cushion (Sinus)**>: each Voronoi cell has a sinusoidal shape, also symmetrical about the seed, so the surface elements can intersect above the cell edges.
14. <**Pixel Size**> is the dimension of each pixel in the generated image. A value of 1.0 would mean that each pixel would represent 1.0 mm of object. An object 100x50 mm would generate an image 100x50 pixels. A <Pixel Size> of 0.1 would then generate an image 1000x500 pixels. Computation times increase commensurately.
15. <**Cushion Scaling**> is a scaling factor (typically in the range 0.5 to 5.0, default is 1.0) that can be used to accentuate the surface profile when “cushion” surface types are selected. The factor can be used to:
    a. Reduce large dark (deep) areas when points are a relatively long way from the nearest seed point (set value < 1.0).
    b. Make cushion edges appear deeper (set value > 1.0).
16. <**Surface Profile**> options describe whether the created surface is:
    a. <**Embossed**>: i.e. the Veronoi seed points appear above the model surface and the cell edges at Z=0.

    b. <**Impressed**>: i.e. the Veronoi seed points appear below the model surface and the cell edges at Z=0.
17. <**Image File**>: is the name of the file to save the image (PNG, JPG, BMP and TIFF files are supported, as specified in the file name extension). The image is sized to just contain the extents of the bounding shapes.
18. <**…**>: browses to identify the destination folder and file name, and places the full path name in the file name text box.
19. The progress bar shows the progress when creating Voronoi shapes and surface images.
20. <**Cancel**> button can be used to terminate the Voronio generation and (in particular) the image generation process if it is taking too long (hi-res images can take some time to produce).

Some experimentation may be required to get the results you need. Remember that when a random set of seeds is used you will get a different result each time. To get a repeatable result: manually defined seeds can be used; and adjusted as required. You can also save a generated set of seeds, and use it again by selecting the <Manual> option with zero randomness.

The surface models may also require some experimentation as the differences between the various surface styles may not be always clear, and different effects may appear, especially when the Voronoi cell is clipped by the bounding shape.

Surface model bitmaps can be imported directly into CamBam, or converted to an STL model using my Texture Builder plugin.

Operational Notes and Tips:

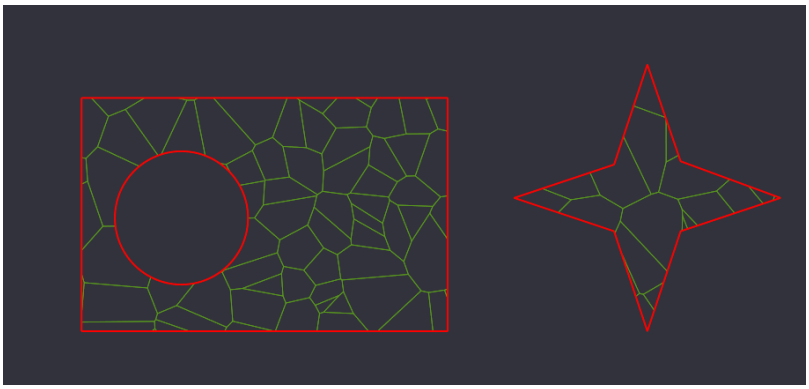
1. The pre-defined grid types offer a quick starting point to define the Voronoi seeds. Any seed pattern can be manually defined and used explicitly by setting the randomness parameter to zero. Some experimentation is required, and useful, to obtain interesting patterns.
2. Variations on the manually defined seeds is allowed by setting the randomness parameter to a value, usefully in the range 0.0 to 1.0.
3. The X-Scale parameters scales the X-axis (in relation to the Y-axis) so a value greater than 1.0 would stretch the seed points apart in the X-direction. Changing the value from 1.0 can produce interesting, and sometimes unpredictable, results. Remember that as the relative X-Y spacing of the seed points change the topology of the Voronoi cells can change radically from what might be expected. Experimentation is suggested.
4. If you want to simply distort a model without changing its topology, then consider applying a CamBam transform to the model *after* generating the Voronoi cells.
5. To get repeatable results the same set of seeds must be used for each run, choose the Manual option with a zero randomness. Generated seeds can also be saved and re-used if required.
6. Hi-res images can take some time to generate and are often not necessary for machining purposes. The time taken depends on both the number of Veronoi cells and the image resolution.
7. If you want a pattern to locate symmetrically inside a bounding shape then it might be useful to:
    a. Generate the pattern, and save the seed points
    b. Move these to the Layer containing the bounding shape.
    c. Translate/scale either the set of seed points, or the boundary shape, to be symmetrical about each other.

d. Regenerate the pattern in Manual mode.

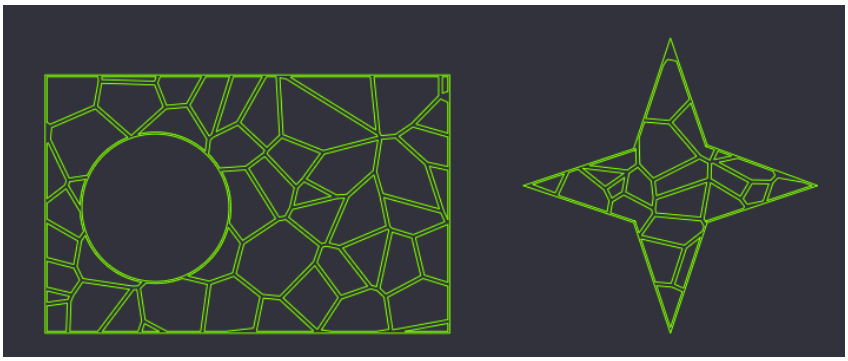
8. It is permissible to have thick cell edges with all types of surfaces but the results may not be useful, unless you want very steep edges around each cell.
9. When using Cushion type surfaces with randomly generated seeds there may be regions of flat surface (either high or low) where seed points are much further apart than the average. To resolve this artifact various options are possible:
    a. Use a smaller randomness setting.
    b. Save the point set and manually add points into the sparse regions.
    c. Adjust the cushion scaling parameter.
    d. Try again!

Examples:

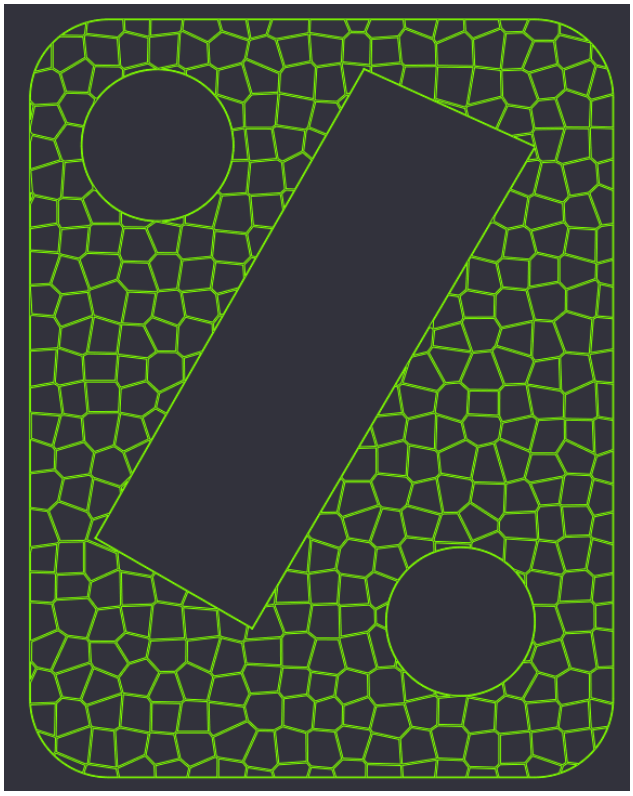
Here is an example of a rectangle with a circular hole (as a Region) and a star shaped polygon:



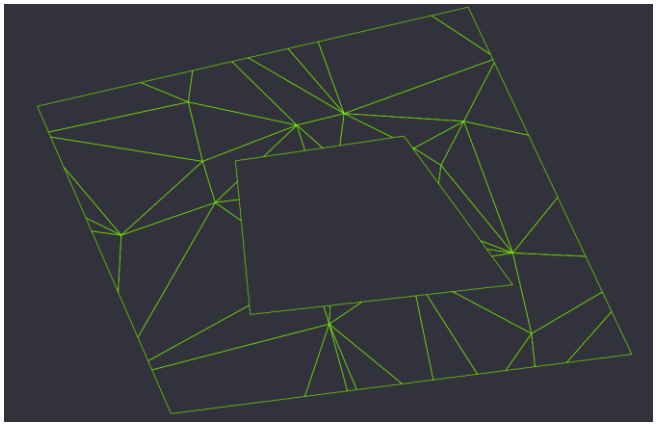
The same example with thick edges and very small shapes removed (edge Thickness = 1.0, and Min Size = 2)



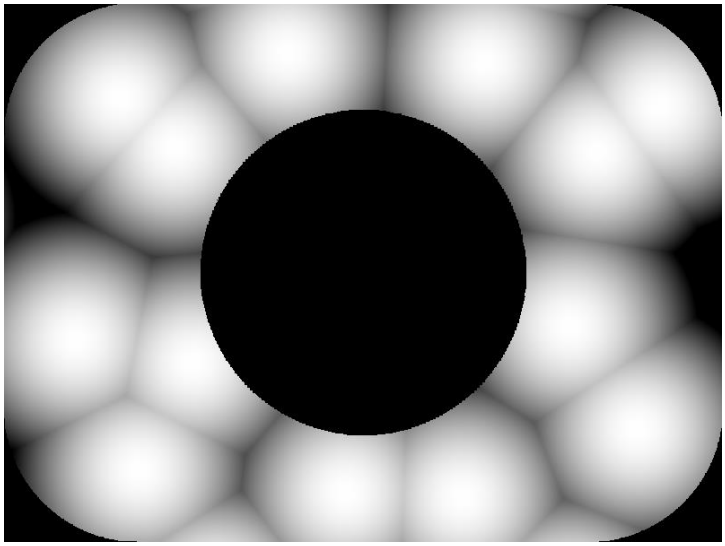
And a more complex example using a randomised square grid of seed points:



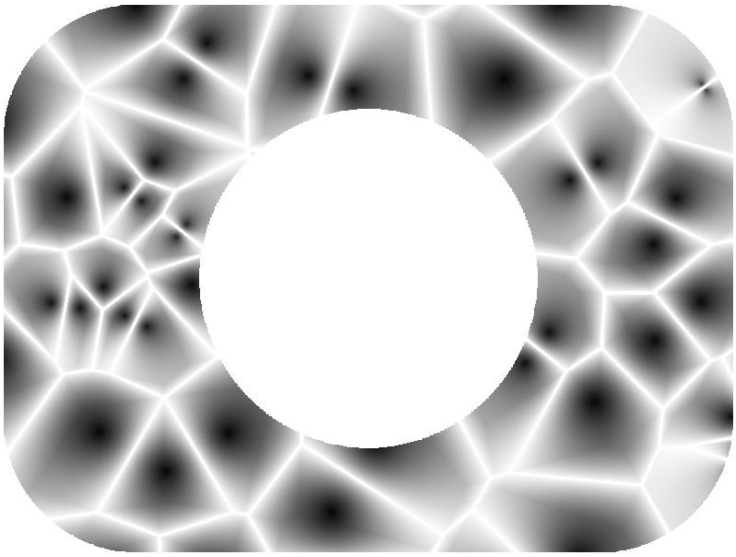
Here is an example Delaunay diagram:



Here is a sample surface texture using a cushion type embossed surface based on a rounded rectangle with a hole.



And here is an impressed surface with a 1mm gap between Voronoi cells.



Images like these can be converted to height maps, and further manipulated in CamBam to create the desired surface element.

Voronoi Builder Plugin Versions

Version	Date	Notes
1.0.1	21/11/2015	First version for feedback and comment.
1.0.2	17/12/2015	• Image generation options added. • Minor improvements to user interface.
1.0.3	21/12/2015	• Can now stop image generation from Cancel button. • Additional grid pattern added for true hexagons (previous hexagon type is now named triangular).

		• Error in translating menu item fixed. • A number of minor bug fixes and fine tuning issues.
1.0.4	22/12/2015	• Problems with image file extensions fixed. • Cancel will now undo any added layer. • Model is refreshed automatically • Attempted translations for combobox items.
1.05	28/12/2015	• Additional grid types added (diamond and octagonal) • Facility to scale grid in an X-direction (relative to Y-direction) added. • Facility added to adjust z-value scaling for cushion type surfaces. • Number of minor bug fixes to surface generation options.
1.06	6/1/2016	• Fixed bug when “Cancel” was clicked to remove the new layer, but the Active layer was not correctly reset causing and exception.
1.07	26/6/2016	• Option added to apply randomness to both X and Y axes, or just the X or Y axis.
1.07a	27/6/2106	• Option added to apply randomness in proportion to the distance from the minimum-X and minimum-Y values of the boundary shapes.