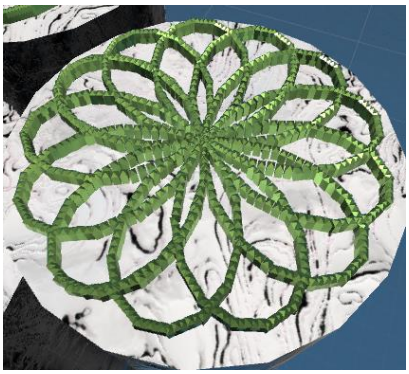
Ornament Generator

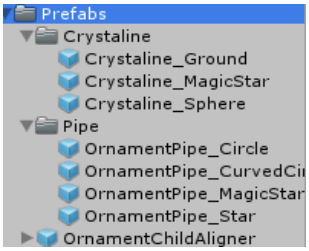
INTRODUCTION:



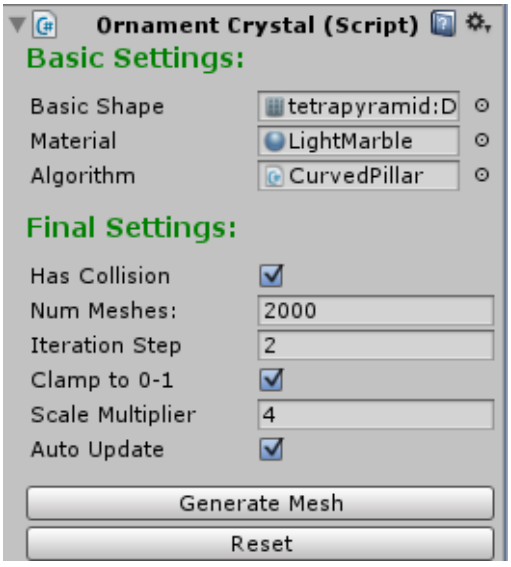
Many objects can be created procedurally and some objects would be very time consumen when created by hand.

The Ornament Generator is capable of creating procedural generated 3D Structures by using those algorithms. Whenever a algorithm is developed, the results are often unknown especially if the formula/algorithm many different values. This means that very simple rules can lead to very complex shapes where small changes could have an huge impact. The main usage is to create decorative structures like ornaments to increase the detail level of your game.

HOW TO USE:



For fast testing, just drag in one of those example prefabs in your scene and hit the “Generate Mesh” button which generates the mesh based on an algorithm. You see a lot of preview spheres which shows, how the result will look like. Currently there are 3 different generators which are the Crystalline, Pipe and Child Aligner. Further details are in their chapters below.



The final settings are the same in all three versions however some generators don’t use every value.

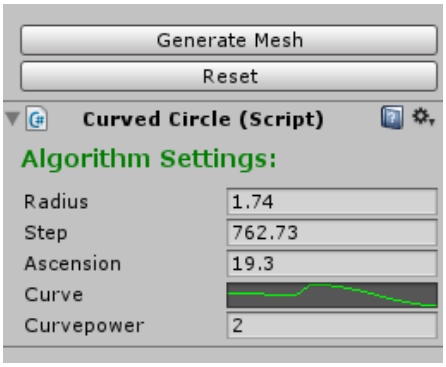
The most important value is the **“Num Meshes”** which describes the amount of mesh-pieces the result will have. The flag “Clamp to 0-1” and “Iteration Step” describes how the index number is set and greatly impacts the appearance of the final result.

Scale Multiplier is used for the individual Crystals or Cross section.

Another option is to let it automatically update the mesh. So you immediately see changes whenever you change parameters. However this is very costly as it will build the mesh every frame so it is capped to 5000 pieces and is deactivated if you exceed this limit to prevent freeze when accidentally adding a zero.

NOTE: If you are going to do unsafe things like setting the mesh count to 10000000, the generator will be locked. You can bypass this lock but you really should know what you are doing.

THE ALGORITHM:

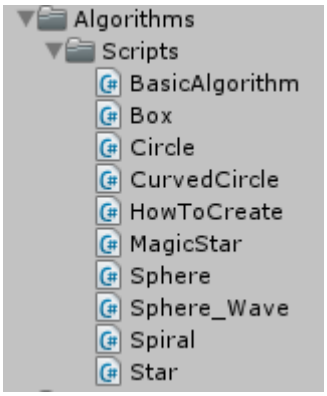


In order to generate ornaments, an algorithm is required which describes how the pieces are positioned and aligned. The algorithm is simply a script attached to the game object as a component and contains settings and parameters.

The algorithm itself is a simple formula which returns the position, rotation and the scale of one object. It easy to create your own algorithms and is described below.

Usually the algorithm can be very volatile so minor changes of some parameters can completely change the result.

How to create patterns (Requires basic C# skills):



This product is meant to be easily extendable by implementing your own algorithms. The package includes sample algorithms to create circles, spheres, ornaments etc. The easiest step is to just copy and paste one of the included algorithms and change their behavior.

As example, copy the “HowToCreate” script, rename it and open it in any text editor like visual studio or monodevelop. When you open the duplicated file, you see 2 functions which are “Initialize” and “ApplyAlgorithm”. The first function can be used to initialize stuff before the mesh is generated. If nothing has to be initialized, this can be ignored.

The important one is the “ApplyAlgorithm”. This section describes the algorithm. And has the current execution number “I” and the Generator itself as parameter. The function returns a package called TRS which contains the position, rotation and scale. These coordinates are in object space which means that the game object is the center.

For example the “HowToCreate” is a bunch of cosinus calls with some parameters. I had no idea what the result looks like when creating it as I usually don’t understand math at all ☺. Attach it to a generator an play with those values.

Important things:

When you create something you can also doing it wrong. Here is a bunch of advices for creating procedural generated stuff:

- Avoid divisions: Most people learn at school that division by zero is impossible. This also counts for such algorithms. Now it is not dangerous when you divide something by a fixed value which does not change. It is dangerous when the divisor changes and could potentially lead to zero which wrecks havoc with everything.

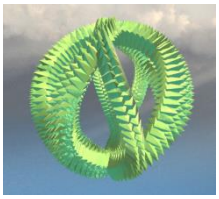
- Altering the Generator: Some Algorithms are initialized by calculating the requiring mesh count which is required and changes it. High values usually get rejected unless you bypass the safety feature. Also do a null check as other generators may just use the function and nothing else

- Using Patented Formulars: Yes it is true! Formulas can be patented (if you have the money). One example is the Superformular which can create many different objects. Whenever you commercially sell your product, you have to make sure that you don't inflict some cancerous patents. If it would not be patented, it would be included to this product but it is so I am not allowed to include it and I don't want to risk any infringement. Patents can be really ridiculous in slowing down science. Imagine someone patenting basic math operations.
- Randomness: The algorithms are supposed to be deterministic. You can include randomness, however the result will change very fast when autoupdating so using a fixed seed as parameter is recommended.

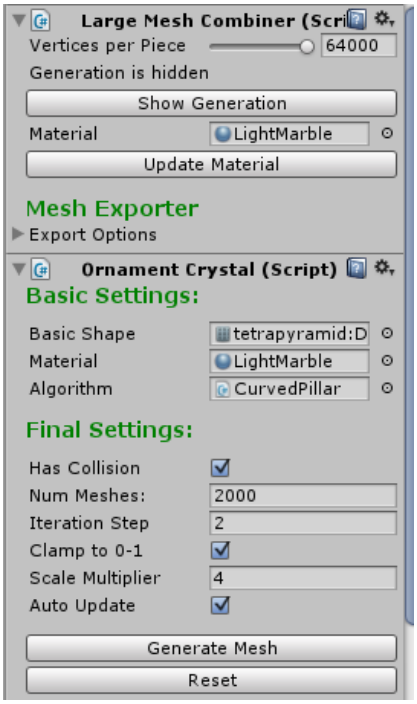
Generator Types:

Currently there are 3 different generators. The most important ones are the crystalline and pipe versions. The third one only aligns children attached to the game object according the pattern.

Crystalline Ornament Generator



This generator is the standard ornament generator. It uses a basic shape which describes each individual crystal. It uses the large mesh combiner and may consist of multiple meshes when the vertex count exceeds the vertex limit.



It is recommended to keep the mesh count smaller than 5000. If it is greater, auto updating is suppressed to prevent slowdowns. The upper limit is set to 100000 pieces without auto updating.

It is also possible to update the generation during play mode however this generator type is the most costly one. It is recommended to use the Pipe Ornament Generator or the Child Aligner because they are way faster that this one. Therefore mesh counts smaller than 500 are advised. The scale multiplier multiplies the size of each individual crystal.

Large Mesh Combiner:

This asset uses the large mesh combiner which breaks the vertices limit of a mesh and allows exporting. It will usually be included in my procedural mesh generation products as it is a core component.

For further details, check out the second included documentation about the Large Mesh Combiner

Pipe Ornament Generator

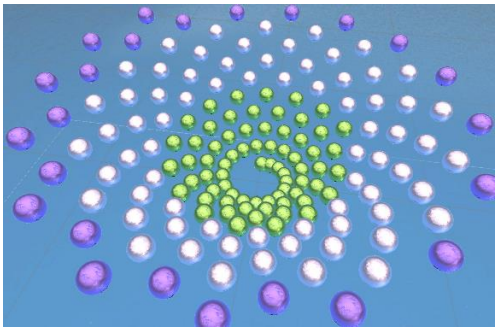
The pipe ornament generator was recently added to this asset and uses direct mesh generation to create ornaments. It uses the same settings as the crystalline generator. However it uses the Pipe Extruder instead of the Large Mesh Combiner and may be found in future assets. The limit is the maximum vertex count because it creates one single mesh. If you exceed the limit, the generation will disappear and no result will be visible.

PIPE EXTRUDER:

The pipe extruder is a new core component similar to the large mesh combiner. It contains important settings regarding this ornament generator. It describes the cross section of the final result and also influences the final vertex count.

For further details, check out the third included documentation about the Pipe Extruder

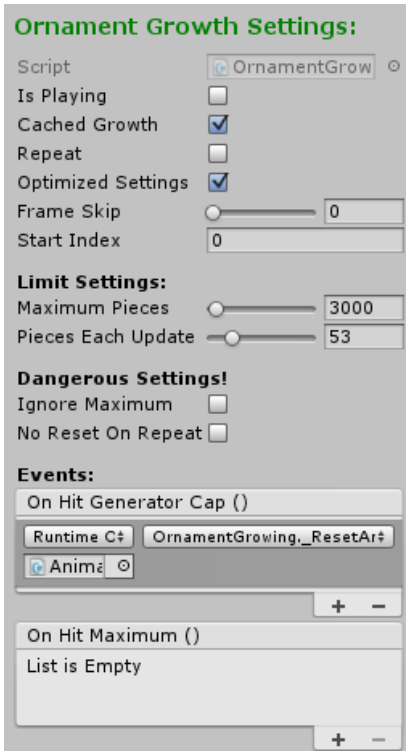
Child Aligner:



The “Child Aligner” can be used to align children of a game object using one of those algorithms where the mesh count is the number of children attached to it. Updating this system is way faster as it doesn’t have to bake meshes into one large structure. It simply changes the transform of those children. This system has no upper limit to the amount of children.

ORNAMENT GROWING:

Objects which are generated using the Large Mesh Combiner can be created gradually over several frames. Gradually generation has some advantages over the standard instant one because more Mesh Counts are possible without killing the frame rate. To use this method, simply attach the Ornament Growing script to the game object containing the Ornament Crystal script. The effect is active during play mode and the result can be different when you exceed the mesh count of the generator.



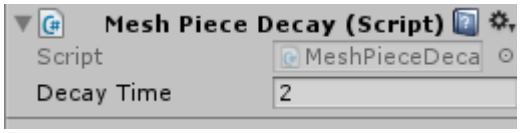
Parameters:

- **Is Playing**: If true, growing is executed each fixed frame
- **Cached Growth:** At the beginning, the position of each piece is pre calculated and stored. While growing, these positions are used instead of calculation using the algorithm. The advantage is the improved performance especially when the algorithm is complex. However you cannot exceed the piece count of the generator(Generator Cap)
- **Repeat:** When the generator cap or maximum pieces are met, the start index will become 0. The mesh will be destroyed unless "No Reset On Repeat" is flagged.
- **Optimized Settings:** Automatically changes the settings to get the best performance.
- **Frame Skip:** Number of fixed update cycles to skip
- **Start Index;** The start index of the mesh generation.

Limit Settings:

- **Maximum Pieces**: The hard cap where the generator will stop. Can be higher than the limit of the generator itself.
- **Pieces Each Update**: How many pieces should be added each growth process. Higher number means faster mesh generaton.

Dangerous Settings (Can freeze Unity if being careless):



IMPORTANT NOTE: These settings will result in potential uncontrolled growing. It is recommended to have at least one removal process attached like MeshPieceDecay which gives generated mesh pieces a lifetime. Amazing effects are possible if you know what you are doing. What you should not do is mark such setting without removal, hit play and go for a coffee and wonder why Unity froze because a mesh with 1000000000 vertices in front of your screen ☺

- **Ignore Maximum:** Hard Limit is removed
- **No Reset on Repeat:** Mesh is not destroyed when it restarts.

Helper Functions:

- **_Play():** starts growing (is playing is set to true)
- **_Stop():** stops growing (is playing is set to false)
- **_ResetAndPlay():** Mesh is destroyed and starts growing from 0
- **_Reset():** Destroys the mesh, index is set to 0.

Ornament Updater:

When you want to update ornaments during play mode, attach the Ornament Updater component to it. It has a simple frame skip option and simply updates the ornament every fixed frame. However keep in mind that the generation process is resource demanding so keep the mesh count low. Also it will refuse updating when the generation process would not be safe. A mesh count under 500 is recommended. Horizontal animation across algorithm settings is significant slower than growing the mesh

Possible Errors/Damgers:

Unity Freezes AHHHHHHHHHHHHHH:

- **YOU COULD HAVE IGNORED THE WARNINGS ☺.**
- Amount of mesh pieces is way to high.
- Amount of game objects is very large:

Z-Fighting:

- This happens occasional when the algorithm creates pieces at the same position with the same orientation. The best way to fix this is by changing the orientation a bit.

God Rays:

A very fancy application is the use in combination with god rays. In the Unity standard assets, there is a Package which imports image effects. This package includes a system called Sun Shafts. However this got removed in Unity 5.6 or 5.5.

Fortunately, there is an Open Source implementation available:
https://forum.unity3d.com/threads/true-volumetric-lights-now-open-source.390818/

Last Notes:

If you have any questions, suggestions, bug reporting don't hesitate to contact me. If you are going to sell a game which uses this asset, inform me because I may buy your game and play it ☺

Contact Information:

E-Mail: mhartl.mmt-b2013@fh-salzburg.ac.at

Homepage: http://lostinpixels.org/