MathMesh – Parametric Surface Mesh Generation System for Unity

A Product by SirisGames

Last Edited: October 19th, 2022

MathMesh is a mesh generation system designed for 3D development in Unity.

Send me an email at sirisgamesstudios@gmail.com if any of the following apply to you:

- You still have questions about the system after reading this documentation.
- You have encountered a bug.
- There's a feature or parametric surface you'd like to see added or changed.
- You're desperate for conversation.

Software Requirements

MathMesh was built to be compatible with all pipelines post 2020.3.0f1. This includes, 3D, URP, and HDRP. The materials in the example scenes however use URP shaders and must be manually switched to compatible shaders if opened in 3D or HDRP. VFX Graph is not compatible with the 3D legacy rendering pipeline, so this feature may only be used in URP or HDRP. This system was built in version 2020.3.27f1 and operates optimally in versions post 2020.3.0f1.

Use Cases

Please read to see if this asset applies to your development needs.

Allows you to generate 3d models of parametric surfaces, with custom defined parameters specifying the number of vertices along each axis, as well as additional parameters specific to the surface.

MathMesh also allows for exporting the FBX files for use outside of Unity.

Two important points to note: The visual effects are set up to only work with MathMesh and use experimental features of the VFX graph.

Second, the meshes are not sealed, meaning there are overlapping vertices at the start and end vertices, causing a slight lighting issue. This can be fixed quickly in Blender or other 3d modelling software.

If after reading this documentation you're unsure if it's a fit for your use case, send an email to the address listed above and I'll try to get back to you as soon as possible.

Side note: If you just want to generate parametric surfaces for fun, we may be soul mates.

Getting Started

Before using MathMesh, make sure that you have FBX Exporter installed. Optionally, in order to view the VFX provided, you can set up URP for your project, and install Visual Effect Graph. After importing MathMesh, in order for the VFX assets to work, you must open them individually and save them, otherwise countless errors will be thrown due to a unity bug.

With your project now set up, you can either use the prefabs provided, or attach MeshGenerator.cs to an empty GameObject.

Start by setting a material for your mesh renderer. One is provided if you don't have any materials. Increase your uSlices and vSlices values to something arbitrary like 15 and select a Mesh Type to generate from the list. Click the "Reset to Defaults" button. Click the "Generate Mesh" button. You should now see a 3D Model in your scene.

The total number of vertices = uSlices * vSlices, so a low poly object should have low values for both variables.

MeshGenerator.cs

MeshGenerator is the only component required to get a basic scene up and running. A MeshFilter and MeshRenderer are both required before the MeshGenerator can be added.

In the Inspector there are two settings along the top: Mesh Type and Mesh Topology.

Mesh Type: The parametric surface you wish to generate. To give an example, one of the more commonly known surfaces in the list is the mobius strip.

Mesh Topology: This setting is only used when the MeshRenderer is enabled. Defines the mesh topology. The three modes are pretty self-explanatory: Triangle, Lines, and Points. In layman's terms, the mesh topology tells the renderer what to draw using the vertices: faces, lines or just points.

Editor Properties

Auto-Update: When auto update is enabled, any change in the inspector will cause the mesh to be re-drawn immediately. This is computationally intensive and may cause lag. It is recommended that you disable this setting unless your u and v values are both low. When Auto-Update is disabled, the Generate Mesh button can be used instead.

Double-Sided: In most cases this should always be enabled. Generates the back face of each triangle.

Display Vertices: Using Gizmos, each vertex is displayed on screen in scene view.

Display Normals: Using Gizmos, the normal of each vertex is displayed in scene view. When double-sided is enabled, this displays normal for both sets of geometry, which can be computationally intensive and messy on screen.

Mesh Properties

U-Slices: The number of vertices equally displaced along the first axis

V-Slices: The number of vertices equally displaced along the second axis. The total vertex count for the model is U-Slices * V-Slices.

Size: Changes the size of the model without modifying the scale (transform)

U: The range of values used to generate the first axis

V: The range of values used to generate the second axis

Additional Parameters: Letters A through E are used to represent additional parameters specific to each parametric surface. Some may have all 5 while others have none. These are also occasionally confined to a range or min/max depending on the surface constraints.

Runtime Buttons

Generate Mesh

Generates or regenerates the mesh using the settings defined above.

Reset to Defaults

Each parametric surface has default values for optimally representation of said surface. Clicking reset to defaults after changing the mesh type is always recommended.

Export Mesh

Opens a file selection window, used to specify where the .fbx file will be saved. This only saves the raw mesh, without visual effects, shaders, or anything else.

MeshVFXManager.cs

An optional component script which can be added is the MeshVFXManager. This script must be added if any of the provided VFX wish to be used, as they don't currently work standalone.

Depending on which VisualEffectAsset is set in the Visual Effect Component, different settings will be displayed in the inspector.

SpawnRate : The number of particles spawned per second

Match Vertex Count: Enabling this spawns a single particle per vertex per second

Pull Force: Gravitational force applied to each particle

Main Tex: The Texture2D used for Voxel and SurfaceSwarm

Color: Defines the gradient color for WireFrame and SurfaceSwarm

Finally, for the SurfaceSwarm VFX, a VFX Property Binder must be added with the property MeshVelocity and the Transform as a target.

Available Surfaces

- Astroidal Ellipsoid
- Bent Horns
- BonBon
- Boy-Apery
- Boy-Bryant/Kusner
- Braided Torus
- Breather
- Catalan
- Catenoid
- Clifford Torus
- Cosinus
- Crescent
- Cross Cap
- Cyclide
- Ding Dong
- Dini
- Enneper
- Hyperbolic Helicoid
- Isolator
- Klein Bottle
- Kuen
- Mobius Strip
- Pseudocatenoid
- Pseudosphere
- Roman
- Scherk
- Snail Shell

- Steinbach Screw
- Sterophere
- Torus
- Triaxial Teardrop
- Twisted Klein Bottle
- Twisted Torus
- Wellenkugel

Adding New Surfaces

To add a new surface not included in this package, please see the steps below.

1. Begin by adding the name of your surface in the MeshType enum in *MeshTypes.cs*
2. In *MMData.cs*, add a new entry following the examples already set. The parameters are:
- The name as a string. This name MUST match method name in step 3.
- The number of additional parameters,
- A float array with the default values,
- Any constraints on the values. A constraint object is made up of an array of strings denoting which parameter to clamp and an array of floats denoting the value. These arrays must have the same order.
3. In *MathMeshUtility.cs*, create a static method for your surface using *GetTemplate* as an example. If your surface is complex, put it instead in *ComplexSurfaces.cs*

Issues with MathMesh / Tips and Tricks

Topology and Exporting

- For the Lines and Points mesh topology, there is a sporadic strange lighting issue where the mesh can only be seen from certain angles. Currently no known solution.
- In order to export the mesh, the Mesh Topology must be set to Triangles.

Lag Spikes

- Unity has an unfortunate bug where Application.Tick gets held up if an operation takes too long. In this case, it causes Auto Update to lag massively since each generation is not well optimized and can take up to 15ms, resulting in potentially minutes of frozen application. Nice one Unity.

VFX

- Unity has a bug that crashes the VFX assets. You might see a huge number of errors in the console log when you try to enable the VFX. In this case, simply opening each VFX asset and saving should resolve the issue.

- There’s a soft cap of 40000 vertices, with u and v slices limited to 200, but this can be disabled by commenting out the Mathf.max call in MeshGenerator.cs (start of GenerateMesh()). Depening on how high you go, you may need to increase the capacity variable in the VFX assets as well (currently at 100000).

- If you like the asset and would like to see the same thing but for implicit, explicit, or complex surfaces, please let me know, email at the top.
- If you find anything else, let me know. Depending on how productive I’m feeling, I’ll either add it to this list or fix it right away. Email at the top.