Easy Voxels: Marching Cubes

Tehnical Documentation

1. Introduction

1.1. Project Description

EasyVoxels: Marching Cubes provides a fast, multi-threaded and reliable way to generate Voxel Geometry using **Dual Marching Cubes** algorithm.

1.2. Algorithm

The marching cubes (MC) algorithm is a widely used technique for computing triangular mesh Iso Surfaces from discretely sampled volume data over rectilinear lattices. **Dual Marching Cubes** tends to eliminate the poorly shaped triangles often present in MC surfaces.

https://www.researchgate.net/publication/4112407_Dual_Marching_Cubes

2. Structures

2.1. FVoxelSettings

Holds the generic settings for Marching Cubes algorithm.

Units `FIntVector`	Grid dimensions sizes in each direction.
Resolution `float`	Voxel Size. Large values are generating lower mesh quality.
ISOLevel `float`	The minimum ISOLevel. All voxel coordinates with values >= ISOLevel will contribute to the mesh.
bInverted `bool`	Whether to invert the generated mesh triangles.
ChunkRadius `FIntVector`	Amount of chunks displayed at once in each direction.
bUseSharedPoints `bool`	If set to true, enables use of shared points to reduce vertices amount.
bForceManifold `bool`	Force Manifold mesh generation. Enabling this removes dualism. **!!!Experimental!!!**

2.2. FDensityPoint

Holds the density data for specific {X, Y, Z} coordinates in the voxel grid.

Value `float`	Point density
Color `FLinearColor`	Point color

2.3. FMeshData

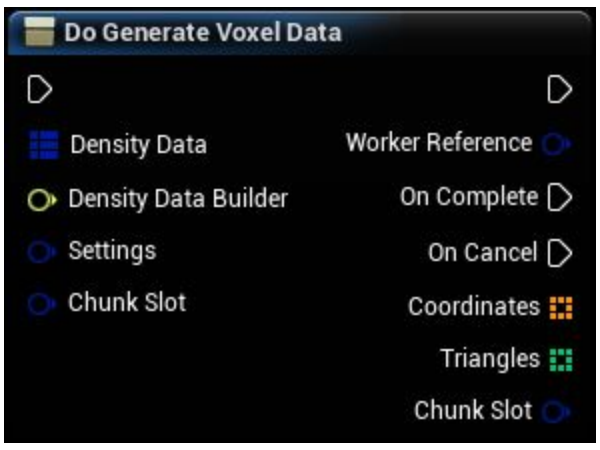
Holds the final mesh data.

Vertices `TArray<FVector>`	Vertex positions
Triangles `TArray<int32>`	Triangles
Normals `TArray<FVector>`	Vertex normals
UV0 `TArray<FVector2D>`	Vertex texture coordinates
Colors `TArray<FLinearColor>`	Vertex colors
Tangents `TArray<FProcMeshTangent>`	Vertex tangents

3. Blueprint Library Functions

3.1. DoGenerateVoxelData

Generate Voxel Data using multi-threaded marching cubes algorithm.



Inputs:

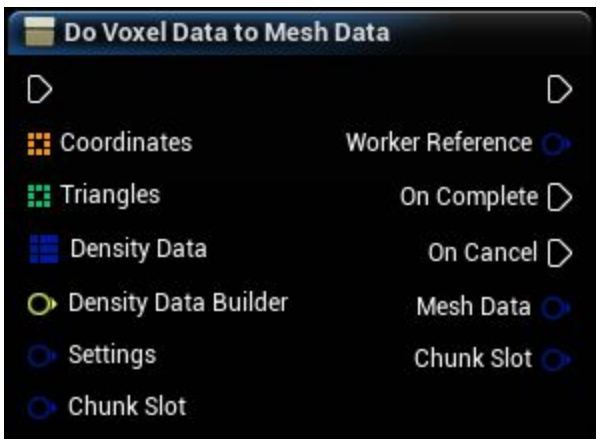
In Exec	
Density Data Set of Density Point Structures	TMap containing pre-generated density data. Voxel coordinates contained in this TMap will not get passed to the DensityDataBuilder. Useful when you want to pass voxel edits to the MC mesh generator. This parameter is optional.
Density Data Builder Density Data Builder Interface (by ref)	Density Data Builder interface used to generate dynamic density data.
Settings Voxel Settings Structure	Generic settings for Marching Cubes algorithm.
Chunk Slot Int Vector Structure	{X, Y, Z} coordinates of the geometry in the chunk grid. If not using a chunk system, could use {0, 0, 0}. We pass this to Density Data Builder Interface.

Outputs:

Out Exec	
Worker Reference Easy Voxels Worker Ref Structure	Generator worker reference. Can be used to cancel or query the generator status.
On Complete Exec	Executes on game thread when marching cubes worker completed voxel data generation.
On Cancel Exec	Executes on game thread when marching cubes worker canceled voxel data generation.
Coordinates Array of Vectors	Voxel coordinates generated by the Marching cubes algorithm that contribute to the mesh.
Triangles Array of Integers	Triangle data for the voxel coordinates.
Chunk Slot Int Vector Structure	Keep track of the work we just completed.

3.2. DoVoxelDataToMeshData

Converts voxel data to mesh data.



Inputs:

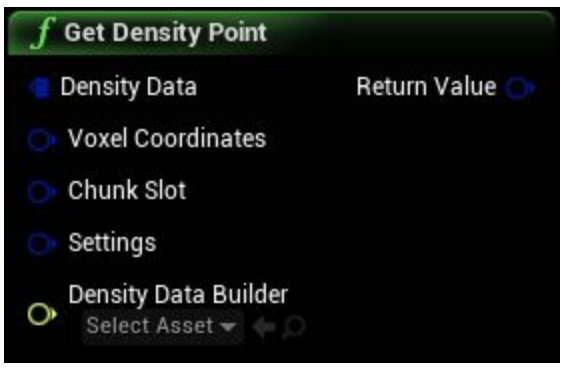
In Exec	
Coordinates Array of Vectors	Voxel coordinates generated by the Marching cubes algorithm that contribute to the mesh.
Triangles Array of Integers	Triangle data for the voxel coordinates.
Density Data Set of Density Point Structures	TMap containing pre-generated density data. Voxel coordinates contained in this TMap will not get passed to the DensityDataBuilder. Useful when you want to pass voxel edits to the MC mesh generator. This parameter is optional.
Density Data Builder Density Data Builder Interface (by ref)	Density Data Builder interface used to generate dynamic density data.
Settings Voxel Settings Structure	Generic settings for Marching Cubes algorithm.
Chunk Slot Int Vector Structure	{X, Y, Z} coordinates of the geometry in the chunk grid. If not using a chunk system, could use {0, 0, 0}. We pass this to Density Data Builder Interface.

Outputs:

Out Exec	
Worker Reference Easy Voxels Worker Ref Structure	Generator worker reference. Can be used to cancel or query the generator status.
On Complete Exec	Executes on game thread when conversion is completed.
On Cancel Exec	Executes on game thread when conversion is canceled.
Mesh Data Mesh Data Structure	Holds the vertex data generated by the conversion from Voxel Data.
Chunk Slot Int Vector Structure	Keep track of the work we just completed.

3.3. GetDensityPoint

Get the Density Point at the given voxel coordinates.



Inputs:

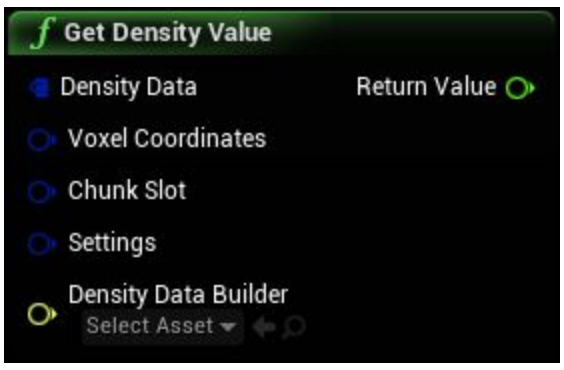
Density Data `Map of Int Vector Structures to Density Point Structures`	TMap containing generated density data. If the Density Point is not in the TMap, Density Data Builder is used to generate it.
Voxel Coordinates `Int Vector Structure (by ref)`	{X, Y, Z} coordinates in the voxel grid.
Chunk Slot `Int Vector Structure (by ref)`	{X, Y, Z} coordinates of the geometry in the chunk grid. If not using a chunk system, could use {0, 0, 0}.
Settings `Voxel Settings Structure (by ref)`	Generic settings for Marching Cubes algorithm.
Density Data Builder `Density Data Builder Interface (by ref)`	Density Data Builder interface used to dynamically generate dynamic density data.

Outputs:

Return Value `Density Point Structure`	Density point at voxel coordinates.

3.4. GetDensityValue

Get the Density value at the given voxel coordinates.



Inputs:

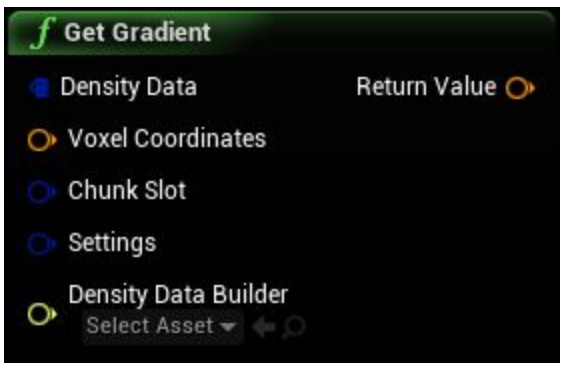
Density Data Map of Int Vector Structures to Density Point Structures	TMap containing generated density data. If the Density Point is not in the TMap, Density Data Builder is used to generate it.
Voxel Coordinates Int Vector Structure (by ref)	{X, Y, Z} coordinates in the voxel grid.
Chunk Slot Int Vector Structure (by ref)	{X, Y, Z} coordinates of the geometry in the chunk grid. If not using a chunk system, could use {0, 0, 0}.
Settings Voxel Settings Structure (by ref)	Generic settings for Marching Cubes algorithm.
Density Data Builder Density Data Builder Interface (by ref)	Density Data Builder interface used to dynamically generate dynamic density data.

Outputs:

Return Value Float	Density value at voxel coordinates.

3.5. GetGradient

Get the Normal Gradient at the given voxel coordinates.



Inputs:

Density Data Map of Int Vector Structures to Density Point Structures	TMap containing generated density data. If the Density Point is not in the TMap, Density Data Builder is used to generate it.
Voxel Coordinates Int Vector Structure (by ref)	{X, Y, Z} coordinates in the voxel grid.
Chunk Slot Int Vector Structure (by ref)	{X, Y, Z} coordinates of the geometry in the chunk grid. If not using a chunk system, could use {0, 0, 0}.
Settings Voxel Settings Structure (by ref)	Generic settings for Marching Cubes algorithm.
Density Data Builder Density Data Builder Interface (by ref)	Density Data Builder interface used to dynamically generate dynamic density data.

Outputs:

Return Value Vector	Normal Gradient.

3.6. GetChunkSlotFromLocation

Convert location to chunk slot.

Inputs:

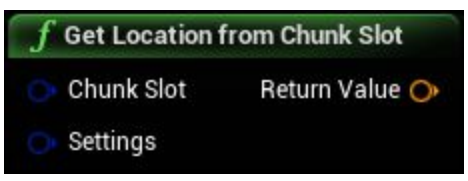
Location `Vector (by ref)`	World Location.
Offset `Vector (by ref)`	Offset Location. This is optional.
Settings `Voxel Settings Structure (by ref)`	Generic settings for Marching Cubes algorithm.

Outputs:

Return Value `Int Vector Structure`	{X, Y, Z} coordinates of the geometry in the chunk grid.

3.7. GetLocationFromChunkSlot

Convert chunk slot to location.



Inputs:

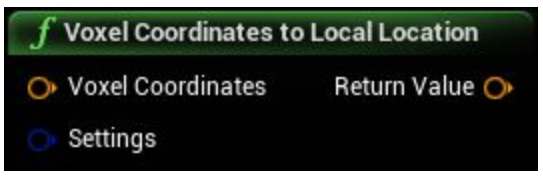
Chunk Slot `Int Vector Structure (by ref)`	{X, Y, Z} coordinates of the geometry in the chunk grid. If not using a chunk system, could use {0, 0, 0}.
Settings `Voxel Settings Structure (by ref)`	Generic settings for Marching Cubes algorithm.

Outputs:

Return Value `Vector`	World Location.

3.8. VoxelCoordinatesToLocalLocation

Convert voxel coordinates to Local Space location.



Inputs:

Voxel Coordinates `Vector (by ref)`	{X, Y, Z} coordinates in the voxel grid.
Settings `Voxel Settings Structure (by ref)`	Generic settings for Marching Cubes algorithm.

Outputs:

Return Value `Vector`	Local Space location.

4. Interfaces

4.1. Introduction

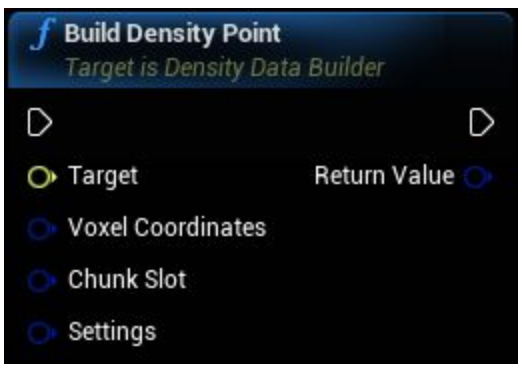
Interfaces are a very important part of Easy Voxels plugin.
They allow us to provide custom data to the C++ Marching Cubes algorithm, at runtime, using blueprints graph. This makes it easy for advanced as well as beginners to use Easy Voxels.

Currently Easy Voxels has one interface implementation available, **Density Data Builder**.

Once implemented and with functionality provided, you just need to pass the object that holds the interface implementation to marching cubes generator and everything will work out fast and smoothly.

4.2. Density Data Builder

Dynamically build density data using blueprints. This Interface implementation is used by Marching Cubes algorithm to generate density data for voxel coordinates that are not contained in the initial provided Density Data TMap.



Inputs:

In Exec	
Target Density Data Builder Interface	
Voxel Coordinates Int Vector Structure (by ref)	{X, Y, Z} coordinates in the voxel grid.
Chunk Slot Int Vector Structure (by ref)	{X, Y, Z} coordinates of the geometry in the chunk grid. If not using a chunk system, could use {0, 0, 0}.
Settings Voxel Settings Structure (by ref)	Generic settings for Marching Cubes algorithm.

Outputs:

Out Exec	
Return Value Density Point Structure	Computed density point.

5. Conclusions

Overall, Easy Voxels provides a super fast solution for runtime geometry generation.
You can generate terrains, asteroids or any procedural geometry.
Literally **everything**.
Easy Voxels is the result of over 1 and half years of research and has gone through lot of changes in order to reach its current state.

The speed, robustness and customization abilities are what makes it a very powerful solution.

Check the provided demo project for more detailed usage of Easy Voxels.

For more informations:
support@yakistudios.com
https://discord.gg/v4D42Vp

THANK YOU