Dream Node Editor V1.0
MeshSpec Documentation

Table of Contents

0. Prerequisites and Intended Readers
1. Introduction
2. Mesh Import Guide(.obj)
3. Mesh Spec File Syntax
4. Mesh Specifications and Shader Types
5. Summary
6. Trouble Shooting

0. Prerequisites and Intended Readers

Please read UserManual.pdf first before reading this document to have a general view of the application first.

This document assume the reader has knowledge about creating 3D models and textures and how to export them into .obj and common texture formats. If not familiar with those concepts, please study external softrwares like Blender and Photoshop first.

For the most part, this document assume the reader has some programming knowledge and know GLSL concepts. If not, never mind.

1. Introduction

For introduction about MeshSpec files, refer to UserManual.pdf. Here are the main ideas: MeshSpec files must reside in one of the "Data" folders, of subfolders of the "Data" folder in order for the application to find it; Drag and drop a MeshSpec file onto application canvas to load it into the document.

The user use MeshSpec file to specify which assets to load, and instruct the program how the mesh should be rendered. Internally, a MeshSpec file directly specify the data fed into GLSL shaders.

Three kinds of data are needed in order to import a mesh properly into the application: the 3D model itself, textures the model uses, and texture descriptions about the shape and materials of the 3D model(e.g. how rough and bright the surface is, how much space the surface occupies).

2. Mesh Import Guide(.obj)

There are several considerations when exporting a model from 3D other 3D applications:

- The mesh needs to be triangulated
- The mesh must contain a UV coordinate
- DreamEditor uses right hand coordinates, with Y pointing up and -Z pointing forward
- The whole mesh should use only 1 material and all meshes should be combines into one, i.e. a single .obj should only contain one mesh

Tutorials on specific examples how to export and import an .obj file might be requested if needed, consult author for requests.

3. Mesh Spec File Syntax

MeshSpec files uses a similar syntax to C/C++: each line represents a single property of mesh, optionally separated by ";" symbol, e.g. ShapeMesh = "Temp\Plane.obj"

Both // and /**/ comments can be used in a MeshSpec file.

There are 4 basic data types: boolean, integer, float, string. Arrays of the four basic types are also available, indicated by using a pair of "{}", e.g. PointLightColor = {1.0, 1.0, 1.0}

Boolean values must be types as true or false with no quotation marks, and numbers like 0 and 1 cannot be used otherwise it would be treated as integers, e.g. PointShadow = true;

Floating point values must contain a decimal mark, e.g. PointLightStrength = 1.0 or PointLightStrength = 1. are legit, and they both represent 1.0.

There are a set of predefined property names and types, e.g. PointLightColor, ShapeMesh. Each MeshSpec file uses those properties to fully describe a mesh. For detailed explanation of what those properties are and when to use them, see below.

Strings are used to specify paths to texture files. Those paths are relative to "Data" folders, not relative to the MeshSpec file itself, e.g. "SimpleShapes\ParallaxTest.png" refers to a texture names ParallaxTest.png that resides either in C:\ApplicationPath\Data\SimpleShapes, or in C:\DocumentPath\Data\SimpleShapes.

4. Mesh Specifications and Shader Types

There are 3 available shaders for use with MeshNodes, each with a specific set of properties that can be specified in the MeshSpec file: Texture(2) or Lambert(3) or SEM(4).

Mesh Specification File Format: All fields except Basic Information are neglectable; Order doesn't matter

Basic Information

Property	Type	Property Example	Usage	Default Behavior if not specified
Shader	integer	Shader = 2	Normally Meshes Use Simple Texture Shader(2) or Phong Shader (3) or SEM Shader(4); Certain Shaders Doesn't Need a Texture Coordinate; Once This Has Been Specified, Only Related Properties to That Shader Need to Be Specified, Unrelated Properties Won't Be Loaded.	Shape(1) with default shape parameters
ShapeMesh	string	ShapeMesh="File name"	The Name of The Actual Mesh File	MeshNode won't load properly
CollisionMesh	string	CollisionMesh=" Filename"	The name of The Collision File, Can Be The Same As ShapeMesh; Assign This Property Only If Needed(If you don't know when you need this, then always assign this property)	A spherical collision will be generated for the mesh
Radius	float	Radius = 5.0	Unit in Object space(i.e. meters); Notice floats must have a floating point otherwise it won't get recognized, I know this sucks	Default value 1 unit

			but bear with it	

Shape Shader Properties

Property	Type	Property Example	Usage	Default Behavior if not specified
Color	float*	Color = {0.1, 0.1, 0.1}		Default Red

SimpleTexture Shader Properties

Property	Type	Property Example	Usage	Default Behavior if not specified
ImageTexture	string	ImageTex = "Filename"		Default will use system Texture

SEM Shader Properties

Property	Type	Property Example	Usage	Default Behavior if not specified
EnvBaseTexture	string	EnvBaseTexture = "Filename"		Default Glossy Black
EnvTopTexture	string	EnvTopTexture = "Filename"		Default White
GlossTexture	string	GlossTexture = "Filename"		Default White

Lambert Shader Properties

Field	Type	Attribute Format	Comment	Default Behavior if not specified
GroundColor	float[3]	GroundColor={ 0.277, 0.624, 0.227}	Ground reflection onto the object	Default Black
PointLightColor	float*	PointLightColor = {0.1, 0.1, 0.1}	A light that facilitates lighting the mesh	Default White
PointLightLocation	float*	PointLightLocation = {1,2,3}	The location of the light relative to the object	Default {1,1,1}
DiffuseColor	float*	DiffuseColor = {0.1, 0.1, 0.1}	Only use if DiffuseTex is not specified; Describes the amount of diffuse light accepted by the object, the actual color depends on the lighting condition	Default Red

DiffuseTexture	string	DiffuseTex = "Filename"	If not specified, use diffuse color; If specified, diffuse color is ignored	No Default
SpecularColor	float*	SpecularColor = {0.1, 0.1, 0.1}	Specifies the faked highlight reflection.	White is physically correct
SpecularPower	float	SpecularPower = 1	Specifies how specular the surface is. A bigger value will cause light reflections on the surface to concetrate on a smaller area; Range in [0, 1]	Default 0
SpecularTexture <Should be GreyScale> <Complex Usage>	string	SpecularTex = "Filename"	If not specified, use SpecularPower; If Specified, SpecularPower is ignored. When used it specifies the SpecularPower at each surface point. Only the Red channel of the texture is used to specify SpecularPower; Green channel is used for mesh transparancy; Blue channel for height(Remember to invert black and white) map in Parallax mapping	No Default
EmissiveColor	float*	EmissiveColor = {0.1, 0.1, 0.1}	Only use if EmissiveTexture is not specified	Default Red
EmissiveTexture	string	EmissiveTexture = "Filename"	If not specified, use EmissiveColor; If specified, EmissiveColor is ignored	No Default
NormalTexture	string	NormalTexture	If not specified,	Default Flat

		= "Filename"	not used; Used in conjuction with Height map in SpecularTexture for parallax mapping effects	

5. Summary

To get a 3D mesh into the application: First Create A 3D Shape with UV and Triangulate It, Export as .Obj. Then Create an Accompanying Texture. Finally Create a MeshSpec file to Describe the Mesh. Then Copy All Those Files(One .obj file, Several .png/.jpg/.bmp files, One .spec file) Into a Data Folder. Drag The .spec File in The "Data" Folder onto the Application to Load It.

6. Trouble Shooting

- If after a mesh is loaded you see nothing but a shadow, or the surface appears black, check your texture path spelling in MeshSpec file: there might be some type with texture path, such that the application isn't able to load texture correctly.