Dream Node Editor V1.0
User Manual

Table of Contents

1. Preface
2. Quick Usage Guide
3. Features and Terminology
4. Interface and Viewport Navigation and Node Creation
5. Data Integrity and Mesh Spec File
6. Facilities
7. Shaders
8. Technical Reference Documents
9. Help

1. Preface

Dream Node Editor, or Dream Map Editor, is designed and programmed by Chaojian Zhang, initially for the purpose of organizing dream notes in a graphical way which might help produce more immersive and organized sceneries. Dream map, a dream recall technique, can be used to help formulate a more coherent dream world when conducted regularly.

This application also serves as an openGl programming practice, where common OpenGL techniques like instanced drawing, phone shading, and parallax mapping can be tested. For the above reason, the author doesn't guarantee that the programming techniques being used are by any means completely standard and efficient. The accompanying source code is only for reference purpose only, and any serious study is not recommended. The author prefer any new openGL learner turn for more official OpenGL and Programming Practice books for systematic learning purposes.

The application is designed with open source and portability in mind, thus every functional components, including shaders and application interface graphics, can be customized and edited.

2. Quick Usage Guide

System Requirement: OpenGL 4.2 support

Step1: Unzip the application and put it somewhere, e.g. C:\Applications\DreamEditor

Step2: Create a text file anywhere and give it a name, also change the extension as following: C:\Files\mydocument.txt -> c:\Files\mydocument.dne

Step3: Double click mydocument.dne, choose "select a program from a list of installed programs", navigate to C:\Applications\DreamEditor and choose DreamEditor.exe, also check "Always use the selected program to open this kind of file" so next time you can double click to open it directly.

Step4: Create and Edit, use Ctrl+S to save the document.

3. Features and Terminology

Document

A single .dne or .dream file is called a "document". Each document contains some metadata about the creation time of the document, and most importantly all the nodes in the document. A single document describes a world.

Document Size

There is not theoretical limit to the size of the world, as long as the location of each node doesn't go beyond possible range of a floating point value. The default terrain in the world is stretched to the size 65536*65536 square meters, thus the user will see the end of the terrain one she reaches there. Due to the design of the application, one cannot create nodes in void, although moving nodes outside the terrain into void is possible.

Node

Each document consists of various nodes that form the visual representation of dream worlds, or document sceneries. Each node contain related information data(including texts and images), and also their individual transformations in the world.

Node Types

There are four types of nodes in total in a document, three of which can be created by the user: CameraNode, ImageNode, TextNode and MeshNode. The user look into the scene through a default CameraNode, and navigation inside the document changes the transformation of the CameraNode. ImageNode, TextNode and MeshNode can be created by user using methods explained later.

Mesh Spec File

Mesh specification files, is used to define the properties, including the 3D model being used and textures that describe a mesh node. Mesh spec files are pure text files with suffix .spec as file extension.

Supported Resources and Data Types

Text Data: All printable characters from unicode range 0-65536. Available charactes in current font can be viewed on CharMap.png files. Specifically, English, Chinese, Japanese are supported. Extra characters and fonts can be used, the tool to generate character maps for a specific font, which is used by current application, is BMFont.

Image Data: All common image formats are supported, including .png, .bmp, .jpg and .tga, although not all are tested. Currently only RGB and RGBA formats are supported. No grayscale image is supported for now.

Mesh Data: The application use AssImp as 3D object loading component, and use only the vertex and index and UV information of the first found mesh in the scene file. Only .obj are tested and guaranteed to work, although other file formats like .ma, .blend and .collada should theoretically work.

Recognized Document File Extensions(both lower and upper case can work): .DM .DNE .NOTE .DREAM .DREAMMAP; The distinction is mainly for semantics purpose.

“Data" Folder:

A specific folder that is of special importance is "Data"(Characters must be spelled exactly as shown, not "DATA" or "data") folder. There are two places where this folder can show up: Either in the application folder, or in a folder that contains the document file. The significance of this folder will be mentioned later.

MeshSpec files, 3D models, textures must be in one of the two "Data" folders, or the subdirectories of those "Data" folders. When two files of the same name exist in both the application data folder, or the document data folder, the one that resides in document data folder will be used. This provides a way for each document to override certain document specific aspects of default application data(e.g. default available meshes), while all other documents can still use the default ones.

4. Interface and Viewport Navigation and Node Creation

Document Creation

To create a document file, which is necessary in order to open the application, one can create a plain text file with the extension changed to one of the supported document formats. Then open the file with the application executable. To make later usage easier, it is recommended to manually set the application as default program to open the specific type of file.

By default, a document contains a ground plane, which is defined by terrain.spec inside application directory.

User Interface

There are three components to the user interface: Mainmenu, Property Panel, and Canvas area(Also called Canvas Space due to its 3D nature). All interface components can be hidden except the canvas, by using Alt+F10.

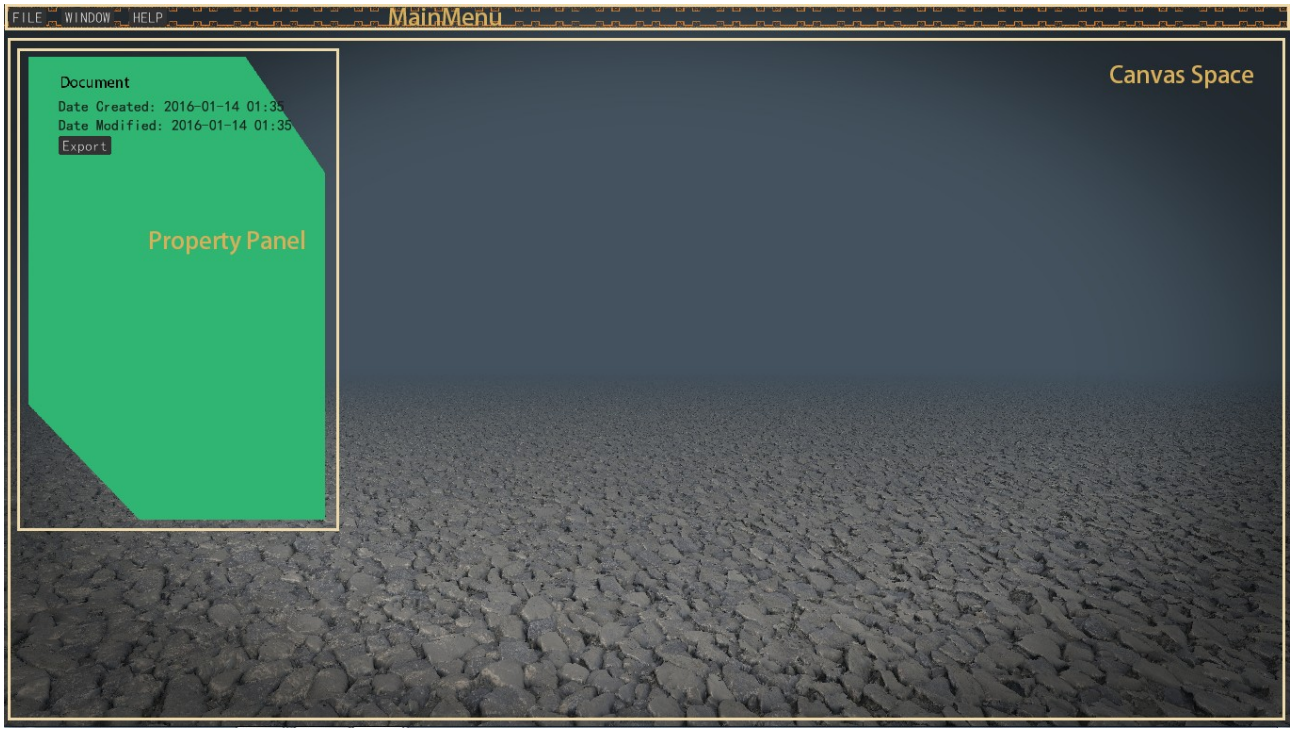
Operations

Navigate the camera: Use WASD to move, use Shift to accelerate, use MMB to rotate, use ZX to lift up and down.

Create a text node: Ctrl+LMB click somewhere on the ground to place a text node. Click on title and content area to add texts.

Create an image node: drag an image onto canvas space to create an image node containing the image. The node will be created above the ground where mouse cursor was hovering on. Click on title and comment areas of image node to add texture descriptions.

Create a mesh node: drag a mesh spec file(.spec) onto canvas space to create a mesh node defined by the spec file. The node will be created above the ground where mouse cursor was hovering on. Click on comment area of mesh node property panel to add texture description.



5. Data Integrity and Mesh Spec File

When working with only text and image nodes, it is safe to move the document file around without losing integrity of the document, however, when working with MeshNodes, it is important to copy all the reference mesh data alongside the document file. Below paragraphs explains how data for each type of three nodes are managed.

TextNodes contain only textes, and the data is saved in the document file.

Once ImageNodes are created, a copy of the referenced image will be kept inside the document file. Thus it is safe to copy a document file around without worrying about needing the original image. This isn't the case for MeshNodes, see below.

MeshNodes store only the name reference to the MeshSpec files. MeshSpec files must reside in one of the two places of the "Data" folder or their subfolders: i.e. either inside application directory, or in the same directory that contains the document file. All paths to 3D model and textures in MeshSpec file are relative to the specific "Data" folder where the MeshSpec file resides, or its subfolder.

6. Facilities

Export:

Warning: When export as files, illegal characters will be stripped out.

Notice: For nodes that doesn't have a title, a default filename will be given.

7. Shaders

Shaders are pure text files written in GLSL. Shaders reside in /Data directory in application folder. Shader specification see Shaders.h.

8. Technical Reference Documents

Shader specification: Shaders.h in visual studio project, download available below

MeshSpec Documentation: Details about how to use MeshSpec files in order to import your own 3D meshes into the application.

9. Help

Tutorial files:

There are two set of tutorials covering the two basic aspects of the application: Navigation and Node creation.

Tutorial files are available for seperate download: download link

Every resource in tutorials files are free for any personal use.

Interface Texture Template and MeshSpec File Template:

In order to simplify content creation process for users, a PSD file of application interface graphics and a template MeshSpec file is provided in /Extras folder

Release Source Files:

For those interested in compiling and tweaking existing application behavior, release source codes are available for download: download link

Original Source Files:

For those interested in the development process of the application, original source files are available for download: download link

The difference between release and original source files are: release version of visual studio project contain only the codes and comments that contribute to the final execution of the application, thus more clear and organized, while original version of visual studio projects contain all sorts of debug comments and tests, which can demonstrate the development process but doesn't contribute much for the final usage. Also original source files contains a log of the application development process and others.

(Notice that original design documents are written in a combination of English and Chinese thus not suitable for English only readers)

Contact:

For any questions, queries, please contact Chaojian Zhang: szinubuntu@gmail.com