Narramancer

A node-based narrative system for Unity

1. Introduction

2. Overview

Narramancer is a node-based scripting system where saving and loading the game state are built-in.

It can be used for branching conversations, for npc behavior trees, or to handle the over-all game loop logic.

The information in this PDF has been abridged for approachability. **For the most current version of the full documentation please visit the official Narramancer website.**

3. Background

The main motivation behind Narramancer is to provide a Unity extension that can chain units of game logic into complex sequences and be able to save and load the running sequence at any given time.

What do we mean by a unit of game logic? For a simple Visual Novel game, the typical units of game logic are:

1. Show or Hide Graphics, such as characters or backgrounds.
2. Play a sound effect or music.
3. Reveal text to the player, either narrative or character dialog. Wait for the player to confirm.
4. Present the player with two or more choices as selectable buttons. Wait for the player to choose one.

A unit of game logic can be something simple that happens instantaneously or something complex that happens over realtime.

Narramancer calls these units Nodes (more specifically RunnableNodes) and has dozens of different Nodes for all kinds of purposes.

4. Why Narramancer?

There are several options when it comes to scripting systems and/or saving solutions. Why should you use Narramancer?

Asset or Package	Disadvantage(s)
Unity	In order to perform a chain of actions in with base Unity it would usually require manually creating Singletons or various inter-dependent scripts.
Unity Visual Scripting	Unity Visual Scripting is a powerful scripting system that uses nodes but does not provide anyway of serializing what node is running.
Fungus	Fungus is another Unity extension that allows scripting using either flow charts or the Lua language but also does not provide anyway of saving and loading.
Naninovel	Naninovel is a powerful visual novel engine that does have saving and loading built-in, but requires the use of Nani scripts (Document and text based files using a proprietary scripting language).

(To be honest, if you are making a visual novel game and would prefer text-based scripting over node-based then we recommend using Naninovel. It is a feature-rich engine with years of experience and support.)

5. Advantages

Building a game using Narramancer has two main advantages:

- Allowing the player to save and load your game at (almost) any given time is baked in from the start, instead of tacked on later.
- The flow and overall logic of your game is easy to develop, edit, and visualize.

Narramancer additionally provides other features:

- Provides a Table of "Things"; a database of people, places, or items that your game can access and manipulate.
- ScriptableObject focused: the main classes in Narramancer are built on top of Unity's ScriptableObject, giving the system ease-of-use and modularity.
- Scene Independent
- General-Purpose variables
- Little or no coding required.
- Rapidly develop and test ideas and logic without having to recompile or even leave play-mode.

6. Getting Started

7. Installation

Import the Narramancer Unity package into your project using the Package Manager

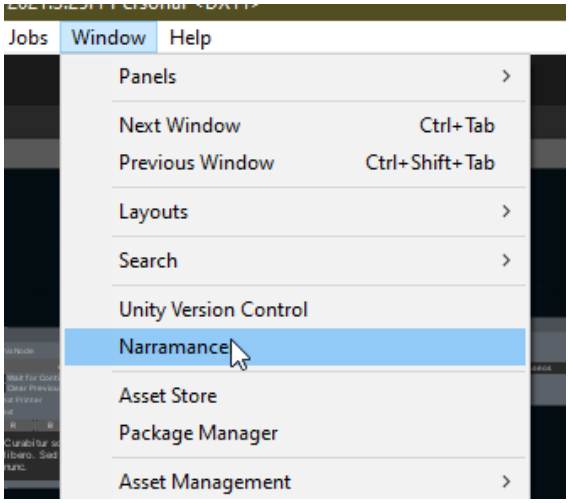
8. Settings

Because Narramancer depends on Odin Serializer, the .NET Api for your project must be set to 4.x.
Go to Project Settings -> Player -> Other Settings and Change Api Compatibility Level to “.NET 4.x”

9. Initialization

Before using Narrmanacer for the first time, you must trigger initialization by opening the Narramancer window.

Open the Narramancer Window by choosing ‘Window -> Narramancer’ from the menu bar.

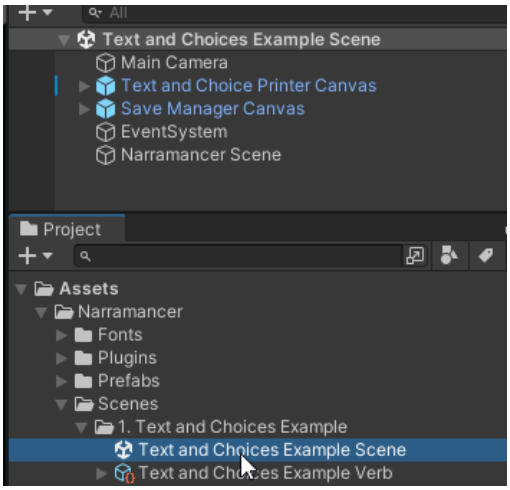


The menu for opening the Narramancer Window.

10. Text and Choices Example Scene

Narramancer includes a handful of example scenes demonstrating some basic implementations of the system. They are located in the folder Narramancer/Scenes.

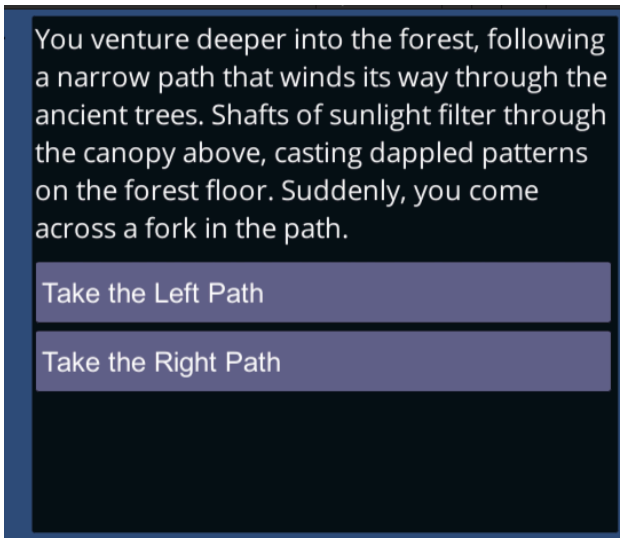
For a simple example of printing text to the screen and offering the player choice buttons open the Text and Choices Example Scene.



A screenshot of where the Text and Choices Example Scene asset is located in the assets folder and the contents of the hierarchy of the scene.

This example scene demonstrates simple text printing and choice selection. Upon entering Play Mode, the user is presented with a text dialog box which will progress when the user clicks on the

screen. Eventually, the user will be presented with two choices, which will result in one of two text paths progressing.



A screenshot of the Text and Choices Example Scene

The flow of this example scene is contained entirely within the ActionVerb asset named 'Text and Choices Example Verb'. If you select the 'Narramancer Scene' object within the scene you will see the ActionVerb included in the list of 'Run on Start Verbs'.

11. Narramancer Window

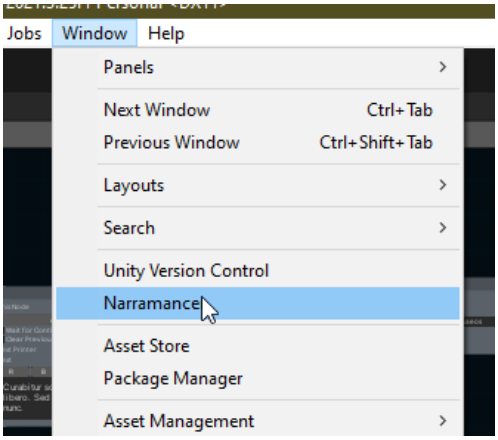
The Narramancer Window acts as a hub for the main elements of your game, including the Verbs, the Nouns, and Global Variables.

Advanced note: the Narramancer Window is the properties panel for Narramancer Singleton, which is a ScriptableObject.

The window has two views: Editor Mode and Runtime Mode.

12. Accessing the Narramancer Window

To open the Narramancer Window, use the menu bar at the top of the Unity Editor, choose Window -> Narramancer.

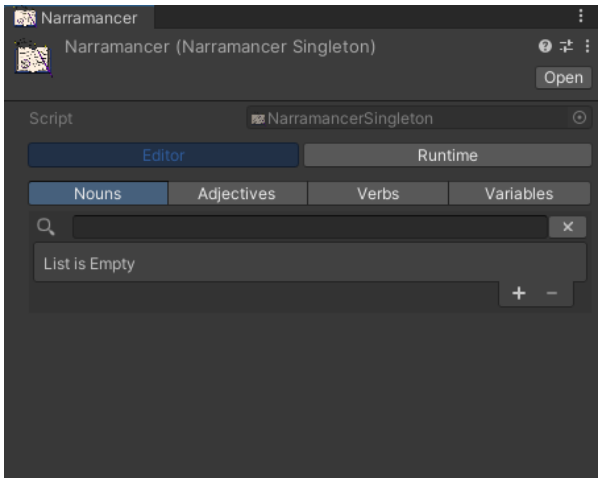


The menu for opening the Narramancer Window.

The Narramancer Window will open up as a Unity Panel, and can be docked in a way like the Inspector panel or the Project panel.

13. Editor Mode

The Editor mode allows you to attach starting nouns and verbs, as well as assign other values.



A screenshot of the Narramancer Window in Editor Mode.

Editor Mode has four subviews: Nouns, Adjectives, Verbs, and Variables.

The first three sub views allow you to assign the starting Nouns, Adjectives, and Verbs, respectively. Nouns applied this way will be instantiated on game start, and Verbs assigned this way will run immediately after.

14. Variables

Narramancer has a built-in 'variables' feature, which allows values to be assigned, accessed, and manipulated from various parts of the system.

The following section outlines what we call Global Variables. There are also Scene Variables and Verb Variables. The three 'scopes' differ in where they exist, but are similar in how they are used.

To use a variable, it must be defined prior to runtime and it must have a type and a name.

15. Creating Variables

Open the Narramancer window, use the Editor mode, switch to the Variables Tab, and add Variables using the variables list.

16. Starting Values

The third field on the variables table is for the 'starting value' (for supported types). The variable will take on this value on game start.

17. GameObject Starting Values

Assigning GameObjects from within a scene to a variable's starting value is not supported. In order to reference in-scene GameObjects using variables, there are three ways:

1. Use a SetGlobalVariablesMonoBehaviour to assign the global variable on scene start.
2. Use a Narramancer Scene component and assign the desired GameObject to a Scene variable.
3. Use a RunActionVerbMonoBehaviour and assign the GameObject to a Verb variable.

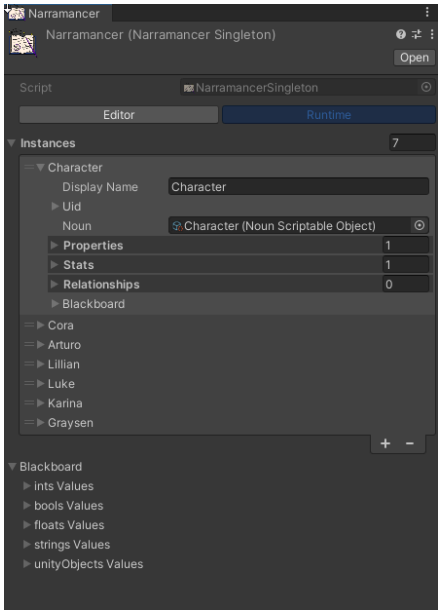
18. Using Variables

Global variables are assigned using SetVariableNodes (with the setting for scope set to Global). Note that the node can only be used in ActionVerbs and will only assign the value when it runs.

Accessing global variables can be done using a GetVariableNode with its scope set to Global.

19. Runtime Mode

During Unity Play Mode and using the Narramancer Runtime Mode, you can see the current table of NounInstances and manipulate various values.



A screenshot of the Narramancer Window in Runtime Mode.

For debugging or testing purposes, you can see all aspects of each NounInstance, as well as add or remove properties and stats.

20. Singleton

The Narramancer system uses a Singleton pattern, meaning there is one static class (called an instance, not to be confused with NounInstances) that handles most of the functionality. The Narramancer Singleton is guaranteed to exist at game start because it is a ScriptableObject asset.

Script and code can call almost any Narramancer method using a static call to NarramancerSingleton.Instance.

This allows you to have access to the table of nouns as well as other functionality.

The Narramancer Window is (more or less) the properties of the Narramancer Singleton ScriptableObject.

21. Narramancer Website

The information in this PDF has been abridged for approachability. **For the most current version of the full documentation please visit the official Narramancer website.**