Tool Serializer

Introduction

The core idea of this method is that we surround types of text around [Tags]. There are 4 types of 'static' tags which include: [Narrator], [Dialogue], [Choice] and [Consequence]. Tags should surround the body of text that are of that type so that it can be effectively processed by the tool.

For Example:

[Narrator]
The lobby is two-stories high. A massive WODA Corp logo takes up the wall behind the front desk. You glance at the receptionist and she smiles – a strange smile, like an involuntary reaction, almost artificial.
[\Narrator]

The first tag opens the [Narrator] "body type" while [\Narrator] closes the body type. Everything in between will be processed as a type of Narrator dialogue.
All tags should be closed with the backslash symbol such as: [\Narrator]

Document Encoding

All text should be encoded in UTF8, this is important to standardize the text across all systems and avoid small text errors due to special characters.

The Tags

[Narrator]

All text that does not have characters should be surrounded by the Narrator Tag.

Example:

[Narrator]
What can you do?
The heat is unbearable today. It's too early in the morning to be this warm. It's been like this for a while and there is no air conditioning in your house. You share it with eco-conscious people, living sustainably, refusing to throw more carbon dioxide into the atmosphere just for personal comfort. It's hard to live by their standards sometimes, but you know they are right.
But it is so hot today. What can you do?
[\Narrator]

[Dialogue]

Dialogue tag is like the Narrator Tag but should be reserved for situations where there is both narration (from the narrator) and characters. Furthermore, Narration that includes text modifications due to consequences given a previous choice (more on this later).

Characters Dialogue should follow the same nomenclature as before:

C_NAME: “Dialogue”

This will be picked up by the tool as a character dialogue. Without this, the tool will assume it is the Narrator speaking.

Example:

[Dialogue]
Your request is submitted. You hope you’ve made the right choice.
In the break room, you get some WODA water and an apple. You break the plastic wrap to find the most lustrous red apple you've ever laid your eyes on.
C_Cory: “Are you the new assistant manager?”
You turn around to see a jovial woman stroking her pregnant belly. You nod and introduce yourself.
C_Cory: “Cory, from accounting.”
She smiles - the first genuine smile you’ve seen today. She asks how the work is going, offers to help you if you need anything.
At the end of the day you make a point to stop by her desk to wish her a good evening. You walk down the hall, where Cory said her desk was…
A massive safety vault catches your attention immediately. It’s so out of place, as if you had suddenly turned the corner into a bank. It takes a while to notice Cory, by her desk, right next to the huge safe door.
You ask her what that is. She smiles and shrugs.
[\Dialogue]

[Choice]

Choice as the name suggests consists of the gameplay element, where the player must pick an option to advance the plot. The Choice syntax is more specific and goes as follows:

[Choice]
(Flavour Text for the Choice)
[C2_1] -> Unique Identifier for the Choice
1: “nervous.”_0
2: “confident.”_0 } Consists of OptionNum: “TextChoice”_ScoreValue
3: “eager to start.”_0
[\C2_1]
[\Choice]

All choice options should have a score value, even if no modification is available. If the later just use the score of 0.

Example:
[Choice]
You check your email: WODA Corp has been using mostly bleached white, silky quality paper. There were a few purchases of recycled paper a few months ago, but not recently. There is a small price difference, but the order will be big.
[C2_2]
1: "Bleached white paper - $2 per 1000 sheets"_-2
2: "Recycled paper - $2,5 per 1000 sheets."_2
[\C2_2]
[\Choice]

[Consequence]

Consequences are tags that need to refer to the choice ID and must be nested inside a Dialogue tag.

Such as:

[Dialogue]
….
[Consequence]
[C2_1.1]
….
[\C2_1.1]
[\Consequence]
[\Dialogue]

Furthermore, consequences need to be identified in the dialogue sections, which will consequently point to the exact text that needs to be presented depending on the previous choice given.

Example:

[Dialogue]
Here comes the big man. Impeccable in a tailored suit, Mr. Avery strides right at you, ready for a handshake. You rise.
C_Mr. Avery: “Hey, you look [C2_1.1]”
[\Dialogue]

Thus, a correct consequence tag should look something like this:

[Dialogue]
Here comes the big man. Impeccable in a tailored suit, Mr. Avery strides right at you, ready for a handshake. You rise.
C_Mr. Avery: “Hey, you look [C2_1.1]”
You shake his hand sturdily. He exhibits an impossibly white smile… And sustains it for an unnaturally long time.
He shows you around the office floor. It’s not typical for the boss to tour new employees – he ensures you know that – but he wants you to feel welcome.
You step into the break room. Between comfy chairs and the kitchen counter are refrigerators filled to the brim with WODA Water bottles.
C_Mr. Avery: “We take our employees' well-being seriously!”
He points to the vending machines. Instead of the usual chip bags and candy, you see a selection of individually wrapped fruit: apples, bananas, berries – each in their own little plastic wrap.
He offers that artificial smile again.
C_Mr. Avery: “Let me show you to your desk.”

[Consequence]
[C2_1.1] # The Choice and the Consequence ID that influences what will be displayed
1: nervous_0
2: confident_0 } Consists of OptionNum: *DisplayedText*_ScoreVal
3: eager to start_0
[\C2_1.1]
[\Consequence]
[\Dialogue]

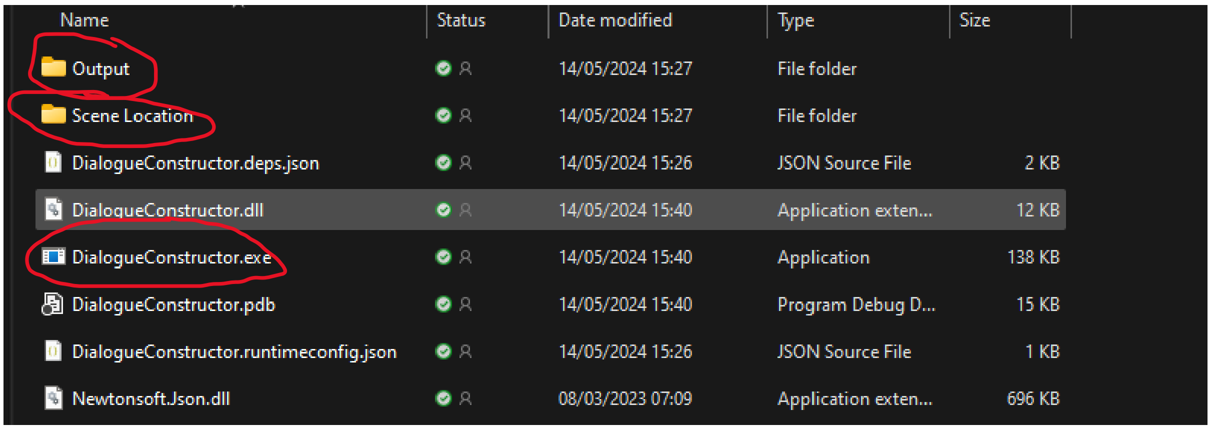
In this case the OptionNum refers to what should be displayed based on the previous option chosen in choice [C2_1].

The Consequence ID serves as an identifier in situations where we wish to use the same choice for different consequences over the course of the game. This allows the game to clearly separate repetitive same choice-consequences.

How does the tool work?

All Scenes should be in a .txt format with a UTF8 encoding and placed in the “Scene Location” Folder.

Running the system consists simply of double clicking the *DialogueConstructor.exe* which will cycle through all the Scenes in the folder and output all the JSON files into the output folder with the same file name.



Name	Status	Date modified	Type	Size
Output	✓	14/05/2024 15:27	File folder	
Scene Location	✓	14/05/2024 15:27	File folder	
DialogueConstructor.deps.json	✓	14/05/2024 15:26	JSON Source File	2 KB
DialogueConstructor.dll	✓	14/05/2024 15:40	Application exten...	12 KB
DialogueConstructor.exe	✓	14/05/2024 15:40	Application	138 KB
DialogueConstructor.pdb	✓	14/05/2024 15:40	Program Debug D...	15 KB
DialogueConstructor.runtimeconfig.json	✓	14/05/2024 15:26	JSON Source File	1 KB
Newtonsoft.Json.dll	✓	08/03/2023 07:09	Application exten...	696 KB

Examples

I have included 2 examples that can be tested to get a sense of how the tool works and the format expected.