Dialogue Editor - 1.0.0
User Documentation

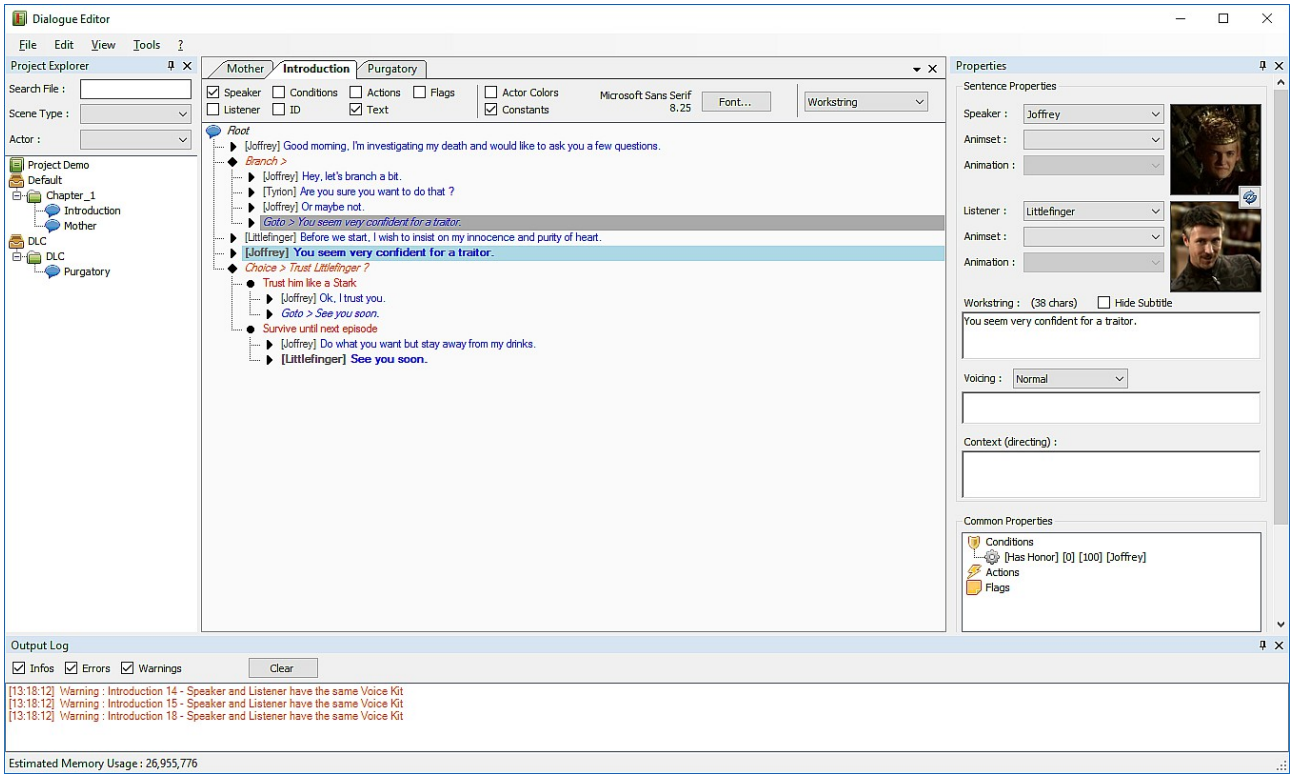
Summary

You can find a list of all the available shortcuts at the end of this document, or by pressing F1 in the Editor.

1 - Overview

The goal of this editor is to allow quick iterations on dialogues writing, and to provide the necessary tools for gameplay flow and data exports.
It's inspired by the tools used on Cyanide's Game of Thrones RPG, and the Aurora tool used on Neverwinter Nights.

The main functions of this editor pipeline are :
- edition of the dialogues trees in a standalone editor.
- exports for the targeted game engines and localization, voicing, and any external tools.
- samples for easy integration and iteration in game engines, with both editors running in parallel.



1.1 - Project files

The files manipulated by the editor consist of a .project file and several .dlg files.

The .project will hold the general information, such as actors, constants, languages and packages descriptions.
It will also act as a root for the files hierarchy. When you open a .project with the editor, all the folders next to it will be parsed recursively to find all .dlg files and fill the editor view.

Each .dlg file will hold a single dialogue. They are independent from each other, and only need to be situated inside a folder next to a .project for them to be manipulated through the editor.
Dialogues are accessed by their names, not their path. This means that every dialogue in a project must ave different names, even across different folders.

Several exporters have been implemented to provide the necessary data for external tools.

1.2 - Dialogues structure

Dialogues consist of a list of sentences, associated with a speaker and a listener.
Special nodes can be used to fork the dialogue's flow and create multiple paths, either by using interactive choices, conditional branches or "jumps" (goto).

1.3 - Nodes IDs

Every dialogue node will automatically receive a numeric ID on their creation. This ID will be unique and constant for each node, and should never be modified by hand.
IDs are used for cross-reference when exporting dialogues and moving nodes.
Rearranging nodes in the tree will not affect this ID.

This implies that after some time editing a dialogue, nodes will appear as if they are disordered (if you display the ID in the interface). This is perfectly normal and intended.
IDs are not indexes, and nodes will be ordered when exporting.

2 - Getting started

Launch the editor with the DialogueEditorBuild.exe executable, or the project specific build executable.

To create a new project, select "File > New Project" then select a folder to host your dialogue files, and enter the name of the project file you wish to create (XXX.project).

To open an existing project, select "File > Open Project" then select the project file you wish to open (XXX.project).

You can edit the project configuration by double-clicking on it on the left panel (Project Explorer). You will need to create actors in the Actors tab of the project configuration, as well as declare the languages you wish to export.
(See part 5 for more details about the project configuration)

You can open an existing dialogue by double-clicking on it on the left panel (Project Explorer), or create a new dialogue by using the menu "File > New Dialogue", or right-clicking on a folder and select "New Dialogue".

3 - Dialogue nodes

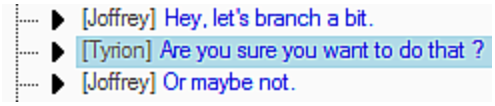
Dialogues are built with a succession of nodes. When playing a dialogue, the game will parse the nodes in sequence, top to bottom from the "Root" node, until a Choice, Goto or Branch node creates a fork. The dialogue will stop automatically after playing a node without child.
You can use five types of nodes to edit a dialogue, defined below.

3.1 - Root

The root node allows the edition of the node attributes, as well as global dialogue's attributes. Additional Actors can be defined on this node. This allows you to include actors in a dialogue that are not used as speaker or listener in any sentence, but should be managed as participants anyway (spectators).

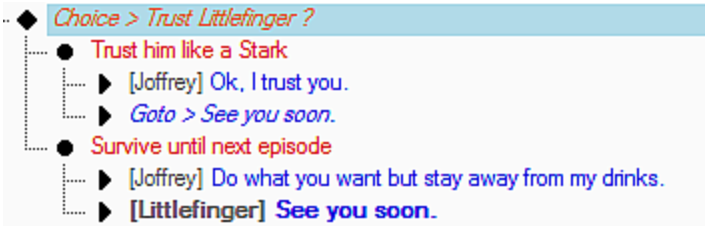
3.2 - Sentence

Represents a single line of dialogue.
It is essentially composed of a workstring, which is the text that will be translated, voiced and displayed in the game.
A speaker must be picked from the Actors list defined in the project file. It will be used for the voicing.
A listener can be defined, to be used by the game's realization.
Additional information can be filled, such as animations, directing, and voicing intensity.



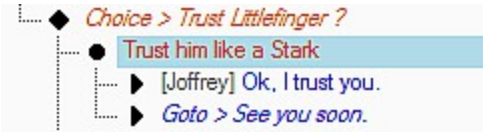
3.3 - Choice

Represents an interactive choice.
It is composed of a list of replies, that must be added as children of the choice.
When parsing a choice node, the game will display the list of all the replies and let the player choose the option he wants to read.
A text can be registered on the choice node for debugging purposes, it is not intended to appear in the game.



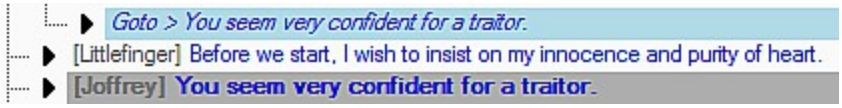
3.4 - Reply

Represents a choice's option.
A workstring must be filled, and will be translated to appear in the game interface, but will not be voiced.
It acts as the new root of a fork.



3.5 - Goto

Represents a jump in the dialogue's flow.
To edit a goto, you have to use "right click > copy reference" on the node you wish to jump to, then use "right click > paste reference" on the goto. Several goto nodes can target the same node. Every goto nodes will display the text of their target as a reminder, and every targeted nodes will be displayed in bold.
You can also use special paste (ctrl+shift+v) to set a goto reference.
Goto nodes are useful to merge different forks back to a common flow, or to organize the nodes hierarchy.



3.6 - Branch

Represents a conditional branch.
It can be used to fork the dialogue's flow.
A branch node needs conditions defined to be effective.
To edit a branch node, a dedicated option will appear on the context menu to add children : "Branch > Add node".
You can also use special paste (ctrl+shift+v) to paste a sequence as branch children.

4 - Dialogue edition

4.1 - Context menu

When you right-click on a node, a context menu will provide several edition options, depending on the selected node.

- Open Directory : opens the directory containing the dialogue.
- Copy Name : copies the name of the dialogue to the Windows clipboard.
- Add Sentence/Choice/Goto/Branch : inserts a new node of the selected type just after the selected node.
- Add Reply : inserts a new reply child node under the selected choice or reply.
- Branch > Add xxxx : inserts a new node of the selected type as a child of the selected branch node (effectively creating a branch).
- Copy Reference : stores the reference of the selected node.
- Paste Reference : assigns the stored node reference on the selected goto node.
- Copy ID : copies the ID of the selected node to the Windows clipboard.
- Move Up/Down : moves the selected node upwards/downwards in the tree.
- Delete : deletes the selected node.

4.2 - Node properties

When a node is selected, different properties will be available on the right side, depending on the type of node.

Root properties allow the edition of :
- the dialogue type.
- the specific voice bank.
- a comment used to describe what the dialogue will be about, or the context.

Sentence properties allow the edition of :
- the workstring, which will be translated and voiced in game. The number of characters is displayed next to it, and will change color when hitting the defined max size.
- the speaker actor, who will be used to actually speak the sentence.
- the optional listener actor, who will be used to define the speaker look-at.
- the optional animset and animation used by those actors.
- the Hide Subtitle option can be used to indicate that a sentence should be voiced but not displayed (for cases like "Aaarrg !").
- the Tone used for the voicing.

Reply properties allow the edition of :
- the workstring, which will be displayed but not voiced. The number of characters is displayed next to it, and will change color when hitting the defined max size.

Choice properties allow the edition of :
- the workstring, which will not appear in game and can be used as a comment for edition.

Common properties allow the edition of attributes (conditions, actions, flags) on each nodes.

4.3 - Node attributes (Conditions, Actions and Flags)

It is possible to declare conditions on a node, like testing if the player possesses an item, or if he has information about a specific topic.

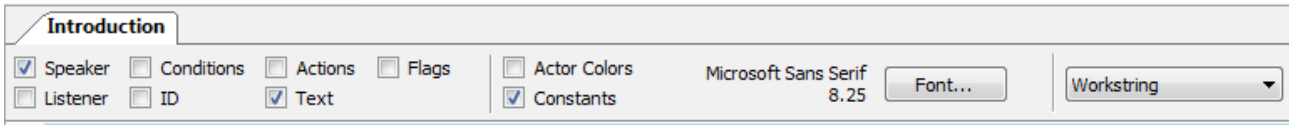
When the game hits a node with a condition, it will execute it only if the condition is valid, and skip it otherwise.
When a node is skipped, the game will try to execute the next child of the current hierarchy level.
Skipping a branch will skip its children.
Skipping a goto will skip the teleport.
A skipped reply will not appear in the interface.

Actions will be used to trigger in-game events, like acquiring items or starting an animation in the level.

Flags will be used to store additional information that will be used by the dialogue system, like indicating that a Reply node will use a skill test when selected.

4.4 - Filters

Filters are available on the document view to show/hide additional information on each nodes.
The first group of filters will control the visibility of different bits of information.
The second group allows you to use a user-defined colors for sentences, to use localized Constants, and to change the font size and style.
The third group allows you to select the language used for display.



4.5 - Constants

Constants are a list of IDs associated to a localized workstring. They are useful to use abstract names for characters or places in your sentences, centralize the localization of their names, and easily modify them during production.
For instance, you can use a constant "KingName", and localize it into "King Robert".

You define these constants in the project configuration.
To use them, you have to wrap their names between { }, like this : {KingName}
You can press Tab when editing a sentence to display the list of available constants.
You can toggle the "Constants" filter on the dialogue view to see the localized value of the constants used in your text.

5 - Project edition

On the left panel, you can double-click on the project's name to open the dedicated edition panel. This panel is subdivided in five tabs : "General", "Actors", "Voicing", "Constants" and "Debug".

The General panel allows the configuration of :
- the languages used by the exporters.
- the packages used to regroup dialogues exports.
- the maximum size of the workstrings in sentences and replies.

The Actors panel allows the edition of Actors used by the sentences :
- the ID is the reference of the actor, used by the game.
- the display name is only used by the editor.
- the picture will be displayed in sentences and in the viewer.
- the gender, species, build, age, height, and personality will be used by exporters (voicing).
- the description will only appear here and not be exported.
- the color can be used when working on a dialogue to highlight the actor's sentences.
- the voice kit will be used on voicing export.

The Voicing panel allows the edition of voice kits :
- Voice Actors represent your physical actors, such as Male1 or Female1.
- Localized Actors allows you to specify the actual names of your voice actors.
- Voice Kits represent the game voices, such as Guard1, King, or Bartender.
- Voice Actors can be affected and shared between several voice kits, allowing the same voice actor to play different characters with different acting.

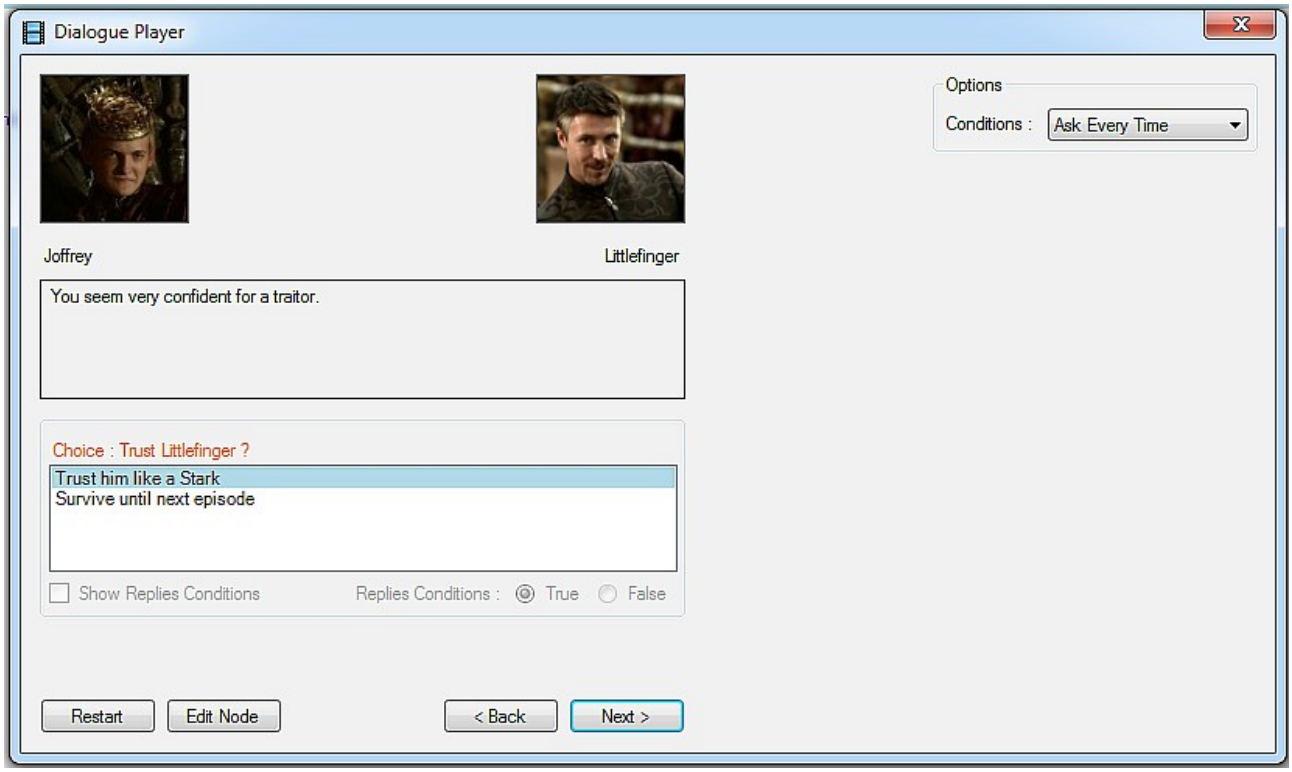
The Constants panel allows the edition of the localized constants :
- the ID is the reference of the constant, used in the sentences in the form {ID}
- the workstring is the base text used for localization of this constant.
- the comments section will be exported in the localization spreadsheet.

The Debug panel displays a view of the custom lists used by the editor.

6 - Tools

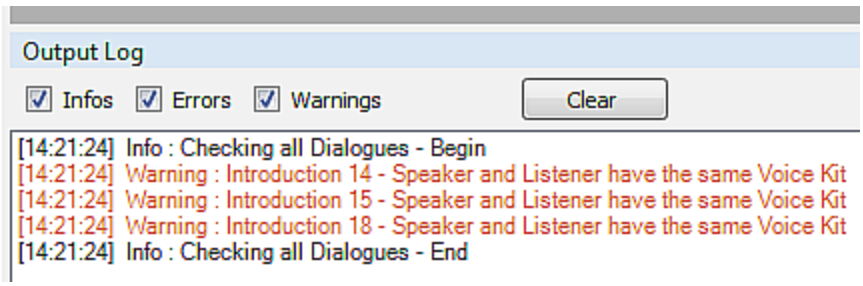
6.1 - Viewer

You can use "Tools > Play Dialogue" to open a little viewer and test your dialogue flow.



6.2 - Errors check

You can use "Tools > Check Dialogue" to let the editor execute a quick sanity check on your dialogue. Error will be displayed in the bottom panel, and you can double-click on those to jump to the concerned node.



6.3 - Exporters

Several exporters are built-in in the editor under the "Tools >" menu.

- Import/Export Dialogues can be used to edit dialogues in a spreadsheet and import the modifications back. This exporter is mainly targeted towards localization.

- Export Stats is used to get word-counts on dialogues.

- Export Localization is used to export your localized strings towards a targeted game engine. Currently only .po files targeted for Unreal4 are provided.

- Export Voicing is used to export spreadsheets dedicated to audio studios, containing informations for the voice actors and the localized texts to voice. A Wwise text helper file is also generated, and can be imported in Wwise to pre-generate all the events with the generated VoicingID.

- Export Game Lipsync is used to export lipsync informations towards generation tools. Currently only text helper files targeted for FaceFx are provided.

6.4 - Localization

Each dialogue holds his own translation table. Each dialogue node will have his own section, containing the translated strings of its workstring.
Those table are currently not editable through the editor, but can be viewed on the dialogue tree by selecting the appropriate language in the filters.

To edit the translations, you need to use the "Tools > Export Dialogues" to export a spreadsheet of the selected dialogues with their current localization information.
You can send this spreadsheet to your localization team, and import it back by using "Tools > Import Dialogues".

Once your translation tables are filled, you can either read them directly from the .dlg files when importing them in your game, or use the provided Unreal .po exporter, or any exporter your custom build may provide.
The voicing exporter will also use the translation tables.

6.5 - Voicing

The voicing exporter allows you to export spreadsheets intended for the voicing sessions. This includes a "Loca_" file to handle all the localized sentences, an "Actors_" file to handle all the actors information, and a "Dialogues_" file to handle general dialogues information.

Each sentence will have a unique VoicingID generated, that is guaranteed to never change whatever happens to the associated sentence. This VoicingID should be used as a basis to name the voicing files, allowing the game to dynamically retrieve them during play.

By default, voice banks should be named after the dialogues names. If you want custom bank names or to regroup several dialogues inside a single bank, you can edit the property "Voice Bank" on the Root node to indicate the intended bank name.

A Wwise text file is generated alongside every export. This file can be used inside Wwise to generate all the sound events with the right VoicingID, scene type, actor and text.

Voicing intentions and scene context can be edited on each dialogue and sentence to provide the voicing sessions with as much information as possible. Those will be exported in the spreadsheets.

6.6 - Copy / Undo

You can copy and paste nodes inside a dialogue, as well as into other dialogues.
Copying nodes attributes is also supported.
Special paste (ctrl+shift+v) can be used to set goto references and branch child sequences.

An undo/redo feature is available for the whole dialogue structure by using ctrl+z and ctrl+y, but only when the focus is on the dialogue tree.
You can also use ctrl+z when editing a single text-field.

7 - Annex

You can display the Help dialog by pressing F1 in the editor.

7.1 - Shortcuts

- Ctrl+N : Create new dialogue file.
- Ctrl+S : Save current file.
- Ctrl+Shift+S : Save all opened files.
- Ctrl+R : Reload current file.
- Ctrl+Shift+R : Reload all project files.
- Ctrl+Shift+T : Open last closed file.

- Enter : Edit current node Text / edit next node Text.
- F2 : Edit current node text / return on node tree.
- Ctrl+Enter : Insert sentence after selected node with same actors.
- Ctrl+Shift+Enter : Insert sentence after selected node with inversed actors.
- Tab : Open the auto-complete box for Constants (Sentences and replies only).

- F4 : Show current dialogue's stats using the current language.
- Shift+F4 : Show the whole project's stats using the current language.
- F5 : Play current dialogue in a viewer.
- Shift+F5 : Play current dialogue starting at current sentence in a viewer.
- F7 : Check current dialogue.
- Shift+F7 : Check all dialogues.

- Ctrl+C : Copy selected node or attribute.
- Ctrl+X : Cut selected node or attribute.
- Ctrl+V : Paste node or attribute.
- Ctrl+Shift+V (on goto) : Set goto reference towards the copied node.
- Ctrl+Shift+V (on branch) : Paste as branching sequence.
- Ctrl+Z : Undo.
- Ctrl+Y : Redo.
- Del : Delete selected node.

7.2 - Tips

- Use copy on the root node to copy the whole dialogue.
- Use copy on a root attribute to copy all its children.
- Use special paste (ctrl+shift+v) to quickly edit goto and branch nodes.

Credits

This editor was developed as part of the Cthulhu project at Cyanide.

Author :
Adrien Cambon "Legulysse"

Additional programmers :
François Bogaert
Antoine Lemaire

Technical director :
Vianney Lançon