Dialogue System

Two types of dialogue are implemented in the game: dialogue boxes and dialogue bubbles:



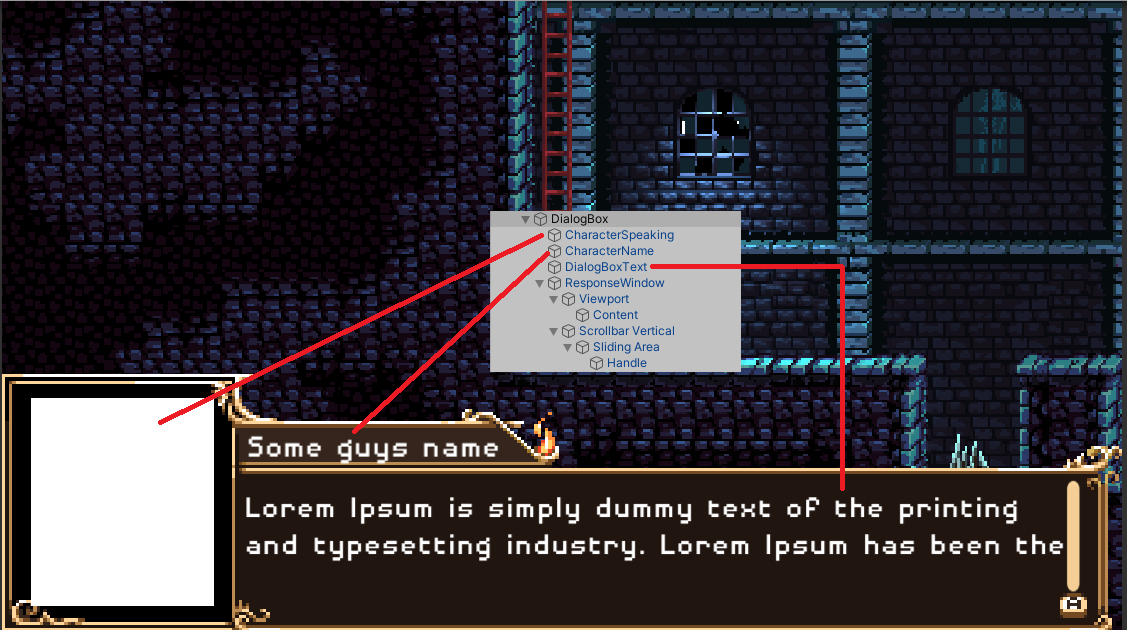
Above you can see an example of choices for a dialogue. Below is an example of dialogue with bubbles:



In both cases the same **dialogue manager** controls them. Beside the manager, a **dialogue trigger** is needed in order to start dialogues. It can be placed somewhere in the scene if we wish to make our character comment something, or it can be placed on an NPC. Dialogues can be triggered automatically or through player interaction. When using boxes, dialogue branching is allowed through multiple choices. In order to know which text to show, a **dialogue importer** is also required. These three elements will be explored in the following sections.

# DialogBox

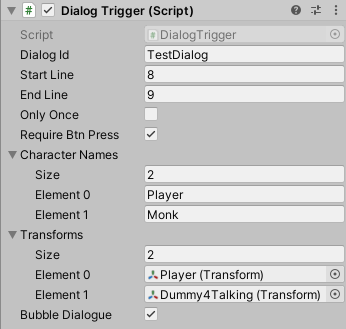
On the following picture you can see the hierarchy of the DialogBox and what it looks like in the scene:



Whether you plan on using dialogue boxes or bubbles, the DialogBox GO should be present and disabled among the UI elements. It is important to have it in the scene as it contains the DialogManager script and it is hooked up to manage all dialogue within the UI controller (UI elements).

# DialogTrigger

The second key element to dialogues is the dialogue trigger:



It has to be put on a game object with a trigger collider. The exposed variables are as follows:

* Dialog Id – It has to match the filename in the dialogue importer, the file we wish to read from
* Start Line & End Line – A file can contain multiple dialogues thus we have to specify on which lines the dialogue for this trigger starts and ends. The line numbering starts with 1, whereas the end line number is inclusive (in the above examples the dialogue will consists of lines 8 and 9 from the file)
* Only Once – Indicates whether the dialogue should fire only once or multiple times
* Require Btn Press – Indicates whether the dialogue should start automatically when the player enters the trigger or if it requires the player to press a button.
* Bubble Dialogue – Indicates if the dialogue will be displayed in bubbles or in a box
* Character Names & Transforms – If we want to use dialogue bubbles, these two lists should be populated with the characters’ names and the appropriate transforms. Based on the transforms, the script will create a dialogue bubble next to the speaking character. The character names should match the names from the dialogue file.

# DialogImporter

The dialogue importer is a critical part of the dialogue system. It should be planned with the rest of the team as it is unlikely that the programmer will be writing all dialogues. You should first make sure to communicate with the dialogue writer and agree on the file format. Without an agreement, it is impossible to go on. After a thorough research on the writing tool features, Unity integration, documentation and support, the idea was to maybe use Yarn Spinner. It is free, a lot of tutorials can be found online, there is even a sample project which I downloaded and tried modifying to match our needs and it was all fairly easy. In my opinion, a huge pro is the simple localization (even though we may not need it in the start, it is better to plan ahead). Another candidate was Ink which is much stronger than Yarn but does not offer localization support. Since Ominous is not controlled by the narrative, Yarn should be more than enough for the game’s needs. To this point the team has not reached an agreement (nor a disagreement, just a lack of feedback whatsoever) therefore the import system was not implemented. Should the need for it arise, I could share with you the test project and documentation I used to try Yarn with Unity.

In order to test the rest of the dialogue system, a simple importer had to be made. It expects a .txt file with the following format:

*ImageName#CharacterName#Text*

The *ImageName* is the filename located in a Resources folder and it will be used as the character’s portrait while talking. When bubbles are used, the *ImageName* will be ignored (but it should still be present in the line otherwise the parsing will fail). Moreover, with bubble dialogue enabled, the characters’ names from the DialogTrigger element, have to match these *CharacterNames* within the file.

In the following picture you can see the temporary simple DialogImporter. It can accept multiple text files. The important thing is that the DialogId in the DialogTrigger, matches the filename in the DialogImporter.

