# Dialogue and Questing TDD

## V\_00.01

## Author: Finn Varvell

# Introduction

## Rationale

This TDD is for making a dialogue and questing system. The dialogue part will include NPCs with multiple lines of dialogue the player can read through and choices that change the dialogue and also the npcs impression of you, which can change what they say when talking to them again. The questing part will allow NPCs to give the player quests that require the player to complete a certain task and receive a reward when returning to the NPC.

## Background

Almost every game has some form of a dialogue system, allowing the player to talk to NPCs in the world or just to narrate the story in some way. Having dialogue in a game greatly expands the game’s world and can help the player understand the story and how to play the game. Certain games will have more dialogue than others, for example a more story-focused game may be mostly dialogue whereas an arcade game may just use it for a tutorial. Questing systems are slightly less important and are more commonly found in RPGs. Having quests to do gives the players’ actions more purpose, and optional quests give the player more freedom and can build the world of the game even more, while the rewards make it feel like a worthy spend of time and that the player is making progress

## Terminology

NPC – Non-Player Character

RPG – Role Playing Game

TDD – Technical Design Document

UI – User Interface

GUI – Graphical User Interface

## Non-Goals

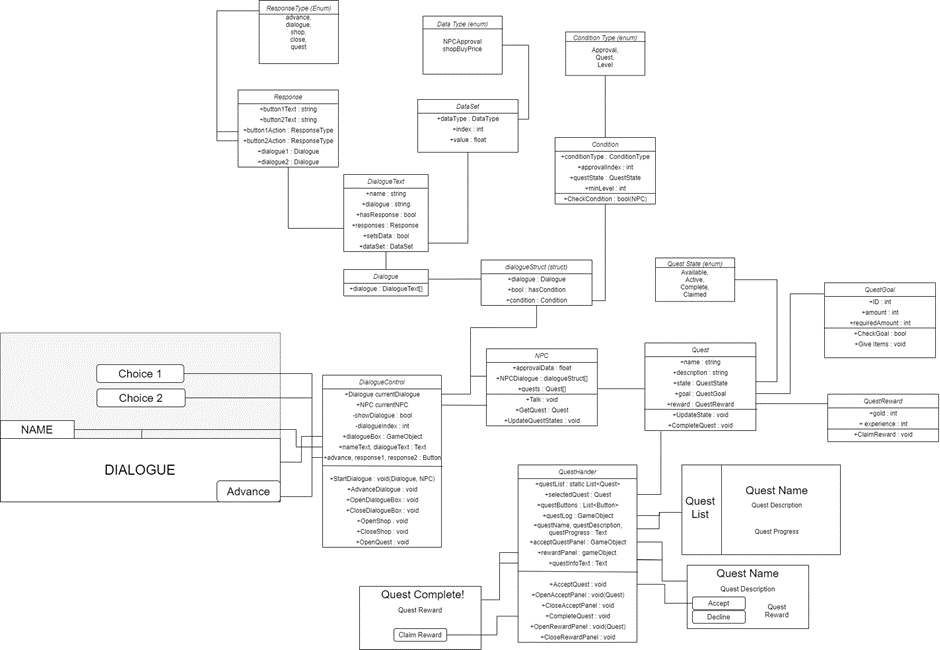
* Dialogue typing out over time rather than all appearing at once
* Player being able to accept multiple quests at once and viewing them in a quest log
* Multiple quests from the same NPC, with one becoming available after the other is completed

## Proposed Design

* Dialogue System
  + Interacting with NPC shows dialogue on screen and pauses game
  + Player can advance through dialogue, and game resumes when it ends
  + Box to show name of whoever is speaking above the dialogue box, can be name player chose for themself
  + Multiple choices can appear, and the player can choose one
  + Different choices can have different effects, including:
    - Advancing the dialogue like normal
    - Moving to a different set of dialogue
    - Instantly ending the dialogue
    - Opening a shop
    - Bringing up the NPCs quest
  + Certain dialogue appears based on different conditions
    - Players making different choices can affect NPC data
    - NPC data changes what they say when talking to them
    - Certain dialogue can only appear based on player’s level, progress on certain quests or how much gold they have
* Questing System
  + Choosing certain dialogue options can bring up the quest of an NPC
  + Quest can be accepted or declined
  + Once accepted, the player can view the quest progress
  + NPC will have different dialogue when quest is active and complete
  + Once quest is complete, talking to the NPC will claim the reward
  + Items required for quest will be removed from inventory upon completion
  + Certain quests will only show at certain levels

# System Architecture

If the design consists of a collaboration between multiple large-scale components, list those components here — or better, include a diagram [UML].

str

## Data types

### Dialogue

* Array of DialogueText, which is a class that holds the information of each piece of dialogue
  + An array is a fixed collection of indexes and elements for each index. So in this case, a collection of DialogueTexts, another class containing the information for each piece of dialogue.

### DialogueText

* Strings for name and dialogue
  + Dialogue will be displayed on the screen in text components, so the data that will be displayed are strings, which are arrays of characters
* Bools for hasResponse and setsData
  + Bools can be true or false, so they are best suited to choosing whether the specific piece of dialogue has a choice from the player or whether it sets NPC data such as their approval
* Response for responses and DataSet for dataSet
  + Response and DataSet are separate classes stored within DialogueText, each with their own variables
  + Marking these classes as serializable means they can be edited in the inspector from DialogueText, essentially making them a struct or collection of variables that can be opened and closed

### Quest

* Strings for name and description
  + Like with the name and dialogue, the quest name and description are just text that needs to be displayed, so strings will be used
* QuestState enum for state
  + Enums are custom identifiers that hold a user-defined list of values
  + The questState enum will have four options: Available, Active, Complete and Claimed, so the state variable can be one of these states. This is easier than using an int or string because these four are the only options and they show clearly what they do
* QuestGoal for goal and QuestReward for reward
  + These are more serializable classes that each quest has with their own variables with goal containing variables for what needs to be done to complete the quest and rewards containing ints for the gold and experience rewards of the quest
  + These classes also have their own functions that can be called, allowing the goal to be checked and the reward to be claimed from inside functions in the quest

## Data Model

* Dialogues will be scriptable objects and all classes used by it will be serializable so that everything can be edited from the scriptable object and dragged into an NPC
* The NPC script will allow for multiple dialogues, with a condition on each, and when talking to the NPC it will go through them all until it reaches a condition returns true, and then that dialogue will be selected.
* Dialogue choices that have the ‘Shop’ response will check if the NPC has a storage script of type shop on it, and then open it and close the dialogue if it does, otherwise it will just advance the dialogue instead. Similarly, choices with the ‘Quest’ response will only open the quest window if the NPC has a quest with the available state.

## Interface/API Definitions

Describe how the various components talk to each other. URL and the format of the data and parameters used.

Libraries used:

UnityEngine.UI – Allows references to the Unity UI such as the text, buttons and dropdowns/sliders/toggles. All the functions of them can then be used, and the values can be changed. Accessing the dialogue box and buttons, as well as the quest windows will require being able to interact with UI.

UnityEditor – using UnityEditor allows parts of the interface to be changed, which may help when making the scriptable objects for the dialogue.

## Impact

* Performance – doing things to reduce the performance cost of running the game
  + Running events that only need to be run once a single time rather than every frame, for example updating the dialogue text only once rather than every frame
  + Coroutines to wait for responses over multiple frames – The DisplayText coroutine will show the text letter by letter so that it types out using a typewriter effect. Coroutines can go over multiple frames, so it will wait between each letter a certain amount of time depending on a typingSpeed variable. This is much better for performance that using a timer in update and checking every frame for if the timer is complete.
* Security
  + Public variables vs private variables – Public can be accessed outside of the class and changed from the editor and private can only be accessed from within the class.

## Risks

If there are any risks or unknowns, list them here. Also, if there is additional research to be done, mention that as well.

* I want to use scriptable objects to hold the data for dialogue but I’m not entirely sure how they work

## Alternatives

If there are other potential solutions which were considered and rejected, list them here, as well as the reason why they were not chosen.

* Rather than in scriptable objects, every piece of dialogue could be in the NPC or a separate script for holding it all, but I think this could get very messy and hard for me to understand, especially when NPCs have multiple sets of dialogue with choices jumping to each other. A better solution would be to try research scriptable objects and figure out how to use them.

# System Testing

## Testing

Show progress, Error reports and explain fixes you used.

**Completing quests –** The quests that involve collecting a certain amount of items should require you to have those items on you when you talk to the NPC to claim the reward, however there was a problem where if you had picked up the item and then dropped it again it still counted as the quest being completed. To fix this, I made it so that every time you open your quest log or talk to the NPC it checks your inventory and updates the quests state to either ‘active’ or ‘completed’ depending on whether you have the items on you at the time or not.

**Giving items** – This was more of a problem to do with inventory, but I needed the items the quest required to be given to the NPC upon claiming your reward. The problem was with the item stacking, as certain quests wanted different amounts of items, so for example if the player had a stack of 10 apples and another stack of 3 and the quest required 5 apples, it should take 3 from the second stack and then another 2 from the first stack, but with the way I had coded it, it would count as if the player did not have enough apples or just take from the first stack depending on the stack it found first. In the end I solved it by using a similar method to stacking the items in the first place, and went through the inventory backwards taking one item at a time until the goal had been reached.

**Multiple quests on one NPC –** There was a problem where quest was just a script added to the NPC, and if you added a second one it would cause errors and crash, so there was no way for a single NPC to have two quests. I tried making it so it would find the first one that was available and choose that one, but then there was still no way to change the NPCs dialogue based on which quest you were accepting. The solution to this was removing the quest giver script entirely and instead making an array of quests on the NPC script where the states of each of them could be checked and different quests could be given in different dialogues.

**Saving Quests –** I wanted the quests in your log and their states to save when you exited the game, but this would require that all the NPC’s approval and quest data also had to save, and I wasn’t sure how to go about doing that, so this wasn’t included.