MODULES

- Map creator (GameScene)
- Dialog creator (SimpleDialogCreatorScene)
- World map (MapScene)
- Other scenes
- Extended Dialog Manager
- Input Manager

Map Creator

Consists of the GameScene and GameManagerController and PauseMenuController scripts.

Has UI for debugging which is not used and should be kept inactive in scene hierarchy as well as turn off the checkbox in GameManagerController script (Debug Section).

The GameManager and the scripts attached are already configured but some values can be modified to customize the grids.

- Game View Section
    - Side view min scale controls the size multiplier of the furthest object in the grid.
    - Playing field count controls how many major grids that will be created.
- Grid Section
    - Grid Cell Size - how big each cell can be
    - Playing field sub grid size - how many cells per sub grid
    - Playing field grid count - how many sub grids in a to create in either direction
    - Boundary color - Color of the bounds around the grid
    - Subgrid color - Color of the bounds around each subgrid
    - World boundary size mult - thickness of the grid boundary
    - Subgrid boundary size mult - thickness of the subgrid boundary
- Transition Section
    - Transition controls which preset to use. Currently works when going to the main menu scene only.

The PauseMenuController script only controls the UI while most of the mechanics (eg. Load/Save) are implemented in the GameManagerController script. The code in the GameManagerController script also has relevant comments.

This scene does not utilise the Input Manager script for handling both mouse and touch inputs. It only implements touch input so any testing for this scene should be done through Unity Remote.

Dialog Creator

Consists of the SimpleDialogCreatorScene, the dialog asset prefab (bubble variant) and dialogCustomiserManager script.

Note that the bubble variant has an ExtendedDialogManager script that handles speech bubble type dialogs while the original version is preferred for RPG type of dialogs. The extended script deals with almost all the features provided in the original apart from emotions and images.

This scene also has another extension script (dialogCustomiserManager) which handles the creation of dialogs without code. To ensure consistency, speech bubbles added to the list must remain in the same order if this script is added to any other scenes. Apart from this the characters must also exist together as children of a GameObject (named Characters in this scene). A custom Canvas is also present which controls the UI for dialog creation purposes.

If this script is used in scenes not needing dialog creation then the 'Is Dialog Creator Mode' toggle should be disabled else it is enabled for this scene.

To test dialog, first create the dialog in the scene (with dialog creator mode enabled and the scene is played in the editor), then save the dialog. Type in the filename in the inspector for the dialogCustomiserManager script and click the button. This should simulate how the dialogs would appear in game. You can also load already created dialogs to alter and make changes.

There exists another scene (DialogCreatorScene) but this is an incomplete prototype that is currently in a buggy state and not recommended for actual usage.

dialogCustomiserManager

Apart from the dialog creation mode, this script should be used if you want to display the dialogs from a file created by the creator. You only need to call the function:

```
Show([filename])
```

If the dialogs are already present or preloaded (using the Load() function) then you can just call show without the filename.

For more information, the source code has comments for the relevant sections.

World Map

Consists of the MapScene and MapManagerController and InputManager scripts.

Has a simple interactable map with each region being a separate sprite contained in the ‘Map’ GameObject. Each region also has a polygon collider which allows the script to detect which region is being clicked on.

In addition, there are predefined map markers which can be interacted with but currently do not have any actions associated with them. Each of the map markers are currently grouped together as children of ‘Map Markers’ GameObject.

The MapManagerController script utilizes the InputManager to handle user interactions and has a few parameters that can be customized:

- Input Section
    - Zoom speed - how fast the camera is zoomed
    - Min zoom, max zoom - limits how much the zoom can be applied
- Map Section
    - Handles generated markers - this section is currently used to show that map markers can also be placed dynamically with code
- Outline Section
    - Handles the outlines of regions when they are selected.
    - Outline Pulse Speed - how fast the outline will be animated
    - Outline Min/Max Thickness - the thickness of outline at the max and min points.

Other Scenes

SpeechTestScene is another prototype of dialog creation. It is incomplete and does not have much to offer. Included in this scene is a speechTestScript that allows a simple test conversation to play when the scene is running. The source code for this conversation should provide an understanding of the features that the dialog managers offer as well as demonstrate how to set up dialogs programmatically.

MainMenuScene only consists of a very basic main menu and a transition which opens the GameScene when the ‘Play’ button is clicked.

DialogCreatorScene is incomplete and not recommended for use.

Other scripts and/or scenes not mentioned are either external libraries which already have their own documentation or have very minor roles or not fully implemented within the project and are not mentioned in this documentation.

Extended Dialog Manager

The setup for this asset is explained in another documentation and a video is also provided to help with the setup. This section goes through the workings of this manager as well as the changes made to the original asset to accomplish the features provided with this. The source code also contains comments for relevant sections.

Changes to DialogManager only include the addition of the events and a function GetCurrentInGameCharacter().

Events:

- OnPrintStartEvent - calls at the start of every single conversation
- OnPrintEvent - called when printing each individual character
- OnPrintFinishedEvent - called after printing has ended
- OnDialogEndedEvent - called when the manager has finished showing all the dialogs, alternately a callback can be added instead as the DialogManager has a callback implemented which is called when dialogs are done.

GetCurrentInGameCharacter() is also included as the characters are handled by this manager. This function makes use of the BindedCharacter prefab as that includes the binder script.

Apart from emotions and images, Extended Dialog Manager can handle almost all the features provided by the original.

This also implements a custom editor to facilitate the setup for characters (setup also explained).

The Bubble dialog asset variant has changes made to the printer object and the text object. This allows for the printer to utilize the speech bubble sprites and adjust to the content of the text.

The BindedCharacter only has the binder script attached (to allow the actual character to be recognised by the manager) and the sprite fully transparent.

Input Manager

This manager makes use of the delegates in Unity to enable the script to be added to any scene and implemented in code very easily.

Currently it supports both mouse and touch inputs, and the various actions that are handled are Drag, Pinch/Scroll and Single/Double tap. Besides these it also informs whether a UI element is being interacted (requires EventSystem) with or the scene.

A few parameters can also be customized to adjust how the system differentiate between different actions:

- Tap vs. Drag action

  Time, velocity and distance thresholds allow the system to determine if the action is a tap or drag. If the action occurs at a time faster than the threshold or the distance of the drag is more or the velocity of the action is fast enough then it is considered to be a drag action else it is a tap.

- Single tap vs Double tap

  The distinction is handled by a timeout threshold that determines the action to be a double tap if the time taken between two tap actions is less than the threshold otherwise the system would consider it to be 2 separate single tap events.

Refer to the source code for more details about the implementation.