# Dialogue System for Unity
# Menu Framework



This package contains a menu framework that leverages the Dialogue System for Unity.

## Requirements

- Unity 2019.4+
- Dialogue System for Unity 2.0+

## Features

- Title Screen Main Menu:
  - Start/Continue/Load game
  - Options menu, Credits scene
- Gameplay Pause Menu:
  - Quests (open quest log window)
  - Save & Load games
  - Options menu
  - Quit to title screen or quit program
- Options Menu:
  - Set fullscreen/windowed, resolution, and graphics quality settings
  - Set music and sound effects volumes
  - Toggle subtitles
- General features:
  - Gracefully switches between joystick and keyboard/mouse modes, auto-focusing buttons and hiding the mouse cursor in joystick mode to allow proper navigation
  - Supports Unity UI, TextMesh Pro, or SuperTextMesh.

# Example Scene

To see the menu framework in action:

1. Open the Build Settings window.
2. Add these scenes:
   - Scene 0: Examples/Start
   - Scene 1: Examples/Loading
   - Scene 2: Examples/Credits
   - Scene 3: Examples/Gameplay
3. Play the Start scene.

# Setup

To use this menu framework in your project:

1. Import the Dialogue System for Unity.

2. Import this package.

3. Add a **Dialogue Manager** to your title menu scene. Assign your dialogue database and dialogue UI to it. This will be your main Dialogue Manager. It will persist when you change levels.

   (a) If you will be using a quest log window, add another Canvas to the Dialogue Manager. Set its Sort Order to 2. Remove the quest log window from the Dialogue Manager's current *Instantiate Prefabs* component. Then add an additiona *Instantiate Prefabs* component, assign the new Canvas as the Parent, and assign the quest log window prefab to its Prefabs list.

   (b) Add a *Save System* component to the Dialogue Manager. Optionally add a *Disk Saved Game Data Storer* component if you want to save to local disk files instead of PlayerPrefs. The Dialogue System Extras page has a "SaveSystemPrefabs" package containing a preconfigured **Save System** prefab with a scene transition manager (see (c) below) and loading screen if you prefer to use that instead of adding a *Save System* component to the Dialogue Manager.

   (c) Optionally add a *Standard Scene Transition Manager* component to the Dialogue Manager, unless you're using the Save System prefab mentioned above. This component can run scene transition effects such as fading to black (as in the example) or using a loading scene.

(d) Inspect the Dialogue Manager's *InputDeviceManager* component. This component handles switching between joystick, keyboard, mouse, and touch input modes.
   - Click the **Add Input Definitions** button at the bottom of the component's inspector to add any missing input definitions to Unity's Input Manager.
   - Optionally customize the list of buttons and axes to check when determining which input device to use.
   - Note: If you use a different input system such as Rewired or InControl, you can assign a delegate method to InputDeviceManager.GetButtonDown. See the corresponding info in the online manual's Third Party Integration sections.
   - If your game primarily uses mouse+keyboard instead of joystick, change the **Input Device** to *Mouse*.

4. Add the prefab Dialogue System Menu Framework/Prefabs/**Menu System** to your scene.

5. If you're using a splash scene before your title menu scene (e.g., Examples/Start), add the splash scene in build settings as scene 0 and the title menu scene as scene 1. Otherwise leave the title menu scene as scene 0.

6. If you're using a credits scene, add the credits scene as scene 3. Otherwise leave it as scene 2.

7. Inspect the **Menu System**. Set the correct scene indices and names in the *SaveHelper* and *TitleMenu* components.

8. Customize the placeholder images and text. Tip: Save this step for the end once you've confirmed that everything works the way you want.
   - Activate each child panel of the Menu System so you can see it while editing. After editing, remember to deactivate it.
   - If the panel has an Animator with trigger parameters named "Show" and "Hide", it will set those triggers when showing or hiding the panel. By default, the PausePanel has an *AnimatorController* that expands the window on Show and shrinks it on Hide.

9. Every panel has On Open and On Close events to which you can assign handlers. By default, when the PausePanel is open it sets Time.timeScale to zero and hides the player's *Selector/Proximity Selector* component (if it's in the scene).

10. The Menu System starts with three saved game slots. If you want to adjust the number of slots:
   - Activate the LoadGamePanel.
   - Duplicate or delete slot buttons.
   - Inspect each slot button and assign a unique slot number (e.g., 0, 1, 2, etc.) to On Click.
   - Repeat the process for the SaveGamePanel.

11. When loading games, the LoadInProgressPanel is shown. If you don't want to show this, unassign it from the Menu System's SaveHelper component. If you want to show a loading screen instead, inspect *SaveHelper* and tick Use Loading Scene. Then specify the build indices of the loading scene and the first gameplay scene. If you want to do something more, assign an event handler to LoadGamePanel's On Load Game event.

12. During gameplay, you can set a Dialogue System text variable named "CurrentStage". The contents of this variable will added to the saved game summary information, which is shown in the details section of the load game panel when the player selects a saved game. For example, you can use a *DialogueSystemTrigger* set to *OnStart* that sets Variable[“CurrentStage”] to the name of the scene, or use a dialogue entry’s Script field to set “CurrentStage” after major story events.

    If you want to add more information to the saved game summary, assign your own delegate methods to SaveHelper.RecordExtraSlotDetailsHandler.

    To show saved game details differently in the load game panel, you can assign an event handler to LoadGamePanel's On Set Details event.

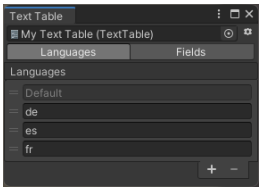
13. If you want to save the player’s position in saved games, add a *PositionSaver* component to the player GameObject.

14. To facilitate playtesting, you can add Dialogue Manager and Menu System instances your gameplay scenes. In your gameplay scene, deactivate the Menu System's TitleMenuPanel. This way you can playtest them without having to come in from the title menu scene. Just keep in mind that, when you come in from the title menu scene, the title menu scene's Dialogue Manager and Menu System instances will be the ones present in the scene.

## Localization

To set up localization and language selection:

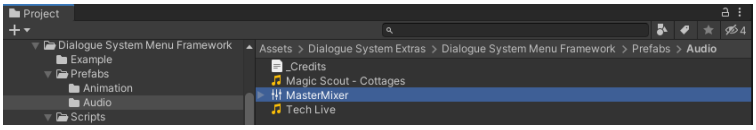
1. Create a Text Table asset, and assign it to the Dialogue Manager GameObject's Display Settings > Localization Settings. Or use the Text Table asset provided in the Menu Framework’s Prefabs/Localization folder. It’s already set up with some starter entries for the title menu.
2. Add your languages to this Text Table asset:




3. Those languages will appear in the Options panel's Languages dropdown.
4. Use the same languages in your dialogue database: Localization.
5. Add LocalizeUI component to your Menu System prefab’s Text elements. The title menu already has LocalizeUI components.
6. In the Menu System’s OptionsPanel, activate LanguageLabel and LanguageDropdown.

## Audio

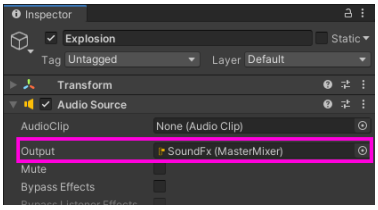
For audio volumes, the Menu Framework comes with an Audio Mixer asset in the Prefabs/Audio folder:




It's assigned to the Options component on the Menu System GameObject.

When you change the audio sliders in the Options menu, it adjusts the volumes of the audio mixer groups in this Audio Mixer.

Assign those audio mixer groups to your Audio Sources' Output fields:




## Loading Scenes

To load a new scene, use the LoadLevel() sequencer command:

> LoadLevel(*sceneName*, [*spawnpoint*])

Parameters:

- *sceneName*: The name of the scene to change to.
- *spawnpoint*: If specified, tells the player's *Position Saver* to move the player to the position of the GameObject named *spawnpoint*. Make sure to use a *Position Saver* and not a *Persistent Position Data* component.

Example:

> LoadLevel(Tavern, Entryway)

Or use a *ScenePortal* component, or call PixelCrushers.SaveSystem.LoadScene() in C#. The "SaveSystemPrefabs" package on the Dialogue System Extras page has a preconfigured **Scene Portal** prefab.

### Music

To set up music, inspect the Menu System's *Music Manager* component. Assign the title scene music to Title Music, and assign other clips to the Gameplay Music list. If Title Music is assigned, it will automatically play when the title scene is open.

To change the music for a scene, add an *Event On Start* component to the scene and set its Music Index to the corresponding index in the *Music Manager*'s Gameplay Music list.

## Scene Streamer Integration

To use Scene Streamer with the Dialogue System Menu Framework addon:

- Define the scripting symbols `USE_SAVESYSTEM` and `USE_SCENESTREAMER`.
- Add a SceneStreamerMenuSystemBridge component to the Menu System GameObject.
- Add a Scene Streamer to the title menu scene, and add a SceneStreamerSaver to it.

## FAQ

*Why am I not seeing dialogue text during conversations?*

Check the Options menu's **Subtitles** checkbox. If it's unticked, the Menu Framework will turn off subtitle text.

*Why is my game's text stuck in a certain language?*

A language selection may be stored in PlayerPrefs. Inspect the Dialogue Manager's Display Settings > Localization Settings, and tick **Reset Language PlayerPrefs**.