Localization

Components overview

GameManager

The GameManager script is a crucial component that manages the game's state and the currently selected language. The GameManager ensures that there is only one instance of itself throughout the game, providing easy access to game-related functions and variables.

languageCode: This variable stores the currently selected language code (e.g., "en" for English). It is initialized with a default value of "en."

SetLanguageCode(string languageName): This method allows the game to switch between languages. It takes a languageName as input, converts it to a language code using the LanguageManager, and updates the languageCode variable. It also triggers a LanguageChanged event using the EventManager class to inform other components about the language change.

LanguageDropdownHandler

The LanguageDropdownHandler script is attached to a TextMesh Pro (TMP) dropdown, and is responsible for handling language selection by the player.

languageDropdown: This variable is a reference to the TMP dropdown component that displays a list of available languages.

Start(): This function ensures that the dropdown exists and is not null. If the dropdown is present, it clears any existing options and fetches the available language names from the LanguageManager. It then populates the dropdown options with these language names, providing players with an up-to-date list of language choices.

OnLanguageDropdownValueChanged(int index): This method is called when the player selects a different language from the dropdown. It takes the selected index as an argument and retrieves the selected language option from the dropdown. Then, it calls the SetLanguageCode method of the GameManager to change the game's language based on the player's selection.

LanguageManager

The `LanguageManager` script manages supported languages and their codes. It provides methods for translating between language codes and names, initializing the language dictionary, and retrieving lists of available languages.

languageDictionary: A dictionary that maps language codes (e.g., "en") to language names (e.g., "English").

InitializeLanguageDictionary(string firstLine): This method initializes the language dictionary with language code and name pairs. It parses the first line from the CSV file to add languages to the language dictionary.

GetLanguageName(string languageCode): Retrieves the language name for a given language code. If the language code is not found, it returns "Unknown Language."

GetLanguageCode(string languageName): Retrieves the language code for a given language name. If the language name is not found, it returns "Unknown Language."

GetAvailableLanguagesNames(): Retrieves a list of available language names.

GetAvailableLanguageCodes(): Retrieves a list of available language codes.

LocalizationManager

The LocalizationManager script manages the localization of the game. It loads translations from a CSV file and provides the ability to retrieve translated text based on the selected language.

translations: This dictionary stores translation data, where the keys are unique identifiers for text elements, and the values are dictionaries containing translations for different languages

csvFilePath: The path to the CSV file containing translation data.

Awake(): In the Awake method, the script ensures that there is only one instance of LocalizationManager, making it persist across scenes. It also loads translations from the CSV file using the LoadTranslations method.

LoadTranslations(): This method reads the translation data from the CSV file, parses it, and populates the translations dictionary with the appropriate translations for each language.

ParseCSVLine(string line): A private method for parsing a line of CSV data, handling quoted strings and commas within quotes.

GetTranslation(string key): Retrieves a translated text for a specific key based on the selected language code from the GameManager.

LocalizedText

The LocalizedText script is attached to TextMeshPro components and allows dynamic localization of text elements in the game.

translationKey: This variable specifies the key used to retrieve the localized text for the attached TextMeshPro component.

textMeshPro: A reference to the TextMeshPro component on the same GameObject.

Start(): In the Start method, the script retrieves the TextMeshPro component and calls the UpdateText method to set the initial text.

OnEnable() and OnDisable(): These methods subscribe and unsubscribe from the OnLanguageChanged event, allowing the text to be automatically updated when the selected language changes.

UpdateText(): This method updates the text content of the TextMeshPro component based on the selected language and translation key by calling LocalizationManager.Instance.GetTranslation(translationKey).

In summary, these scripts work together to enable language localization. The GameManager handles the game's state and language selection, while the LanguageDropdownHandler lets the player choose their preferred language. The LocalizationManager manages translation data and provides the translation for the specified language, and the LanguageManager is responsible for managing supported languages and their codes. Finally, the LocalizedText script allows dynamic localization of text elements.

Expanding language support

How to add more languages

If you want to add more languages, you should update the csv file by following these steps:

1. Add a new column:

You should add a new column for each language you want to add. The column header should be in the format of “languageName(languageCode)”, without the quotes.
For example *French(fr)*.
Add translations for the newly added language(s).

2. Add a new row:

Create a new row in the CSV file with a unique identifier. This identifier should be in the format of "lang_{languageCode}" where {languageCode} corresponds to the language code used in the column header.

In this new row, provide translations for the language names in all available languages. For example, if you added Dutch, English, and French, the row might look like this:
lang_nl;Nederlands (nl);Dutch (nl);Néerlandais (nl)

How to add more translations

1. Create a new translation key:

For each new text element that needs translation, create a unique translation key. These keys should be descriptive and reflect the context of the text. Avoid using spaces or special characters in your keys; instead, use underscores or dashes if needed.

2. Update the CSV File:

1. Open the CSV file containing your existing translations.
2. Add a new row to the CSV file for each new translation key you've created.
3. In the first column of the new row, enter the unique translation key you created in step 1.
4. In the CSV file, go to the columns for each language where you want to provide translations.
5. Enter the translations for the new translation keys in the corresponding language columns. Make sure the translations match the context and are accurate.

3. Update localizedText components:

1. In your game, find the UI elements that should display the new translations.
2. Attach the LocalizedText script to these UI elements.
3. Set the "translationKey" variable in the LocalizedText component to the unique translation key you created in step 2.
4. When the game runs, the LocalizedText component will automatically retrieve and display the translated text based on the selected language, using the new translation keys.