Performant Localization System

Performant Localization System (PLS) lets you manage your game's translations with the flexibility to optimize either CPU usage or memory usage.

Translations are loaded from one or more `.csv` files, depending on your configuration. You can subdivide these files by `Zones` (using a dedicated column) to boost performance.

Features

- Supports **single or multiple** translation files (CSV format).
- **Compatible with Excel** (using CSV format).
- Lets you choose between **CPU optimization** or **memory optimization**.
- Designed to keep your code **clean and readable**.
- Integrated with **TextMesh Pro**.
- Say goodbye to messy key-based translations. Translate directly using text in your native language!

Directory Structure

Performant Localization System should be located in:

```
Assets/Plugins/Performant Localization System
```

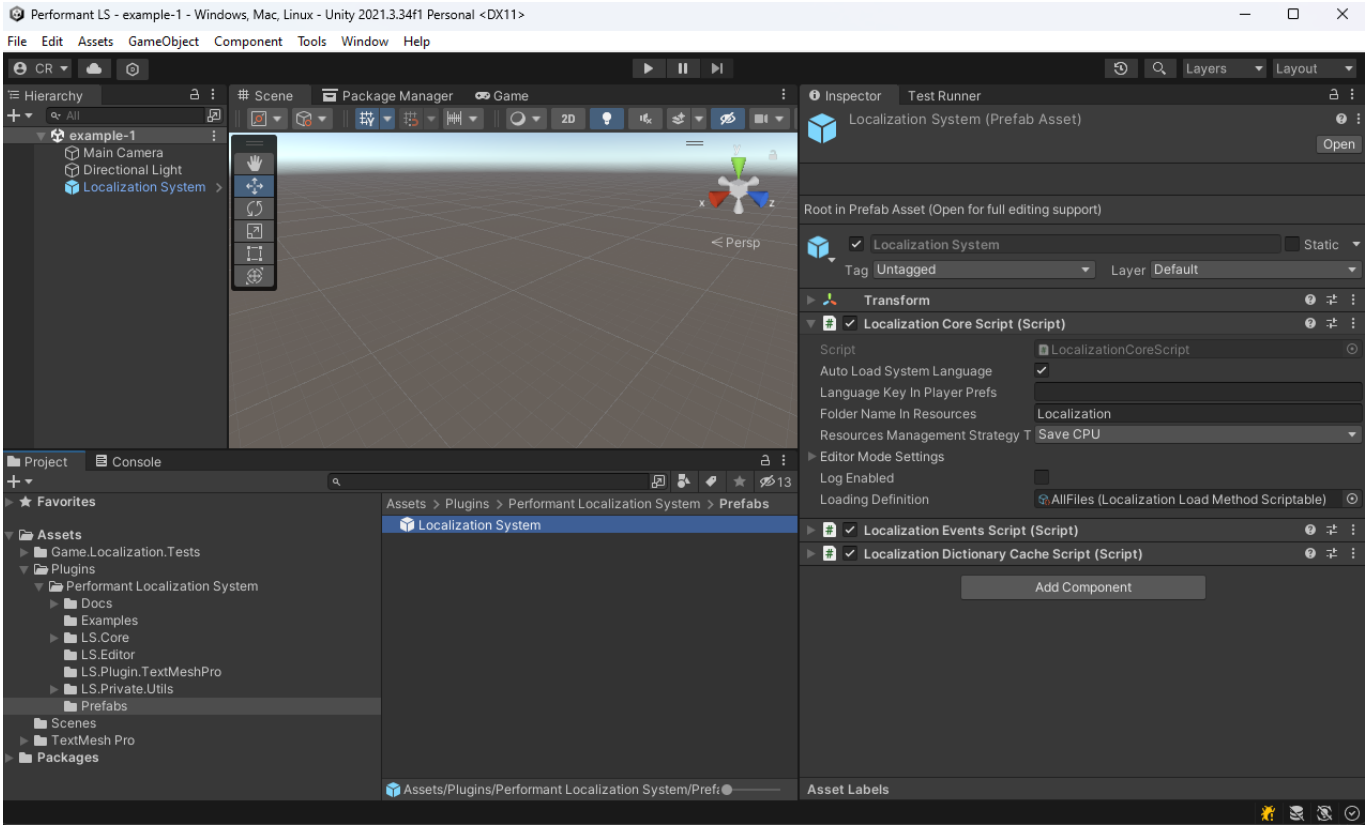
Inside, you'll find the following structure:

- **Docs**: Documentation for the asset.
- **Examples**: Sample scenes.
- LS.Core: Main plugin code. Modifications are not recommended.
- LS.Editor: Code for Unity editor. Modifications are not recommended.
- **LS.Plugin.TextMeshPro**: Additional scripts to simplify translation with TextMesh Pro.
- LS.Private.Utils: Support code for the plugin. Modifications are not recommended.
- **LS.Scripts**: Scripts for you to use in your game.
- **Prefabs**: Pre-configured objects to simplify integration.

Getting Started

1. Add the `Localization System` prefab to the scene.

Drag and drop the `Localization System` prefab into the scene.



This prefab includes three scripts:

- `Localization Core Script`: The main script that handles all translation functions.
- `Localization Events Script`: Provides an event you can subscribe to for detecting language changes.
- `Localization Dictionary Cache Script`: An in-memory cache for storing loaded translations.

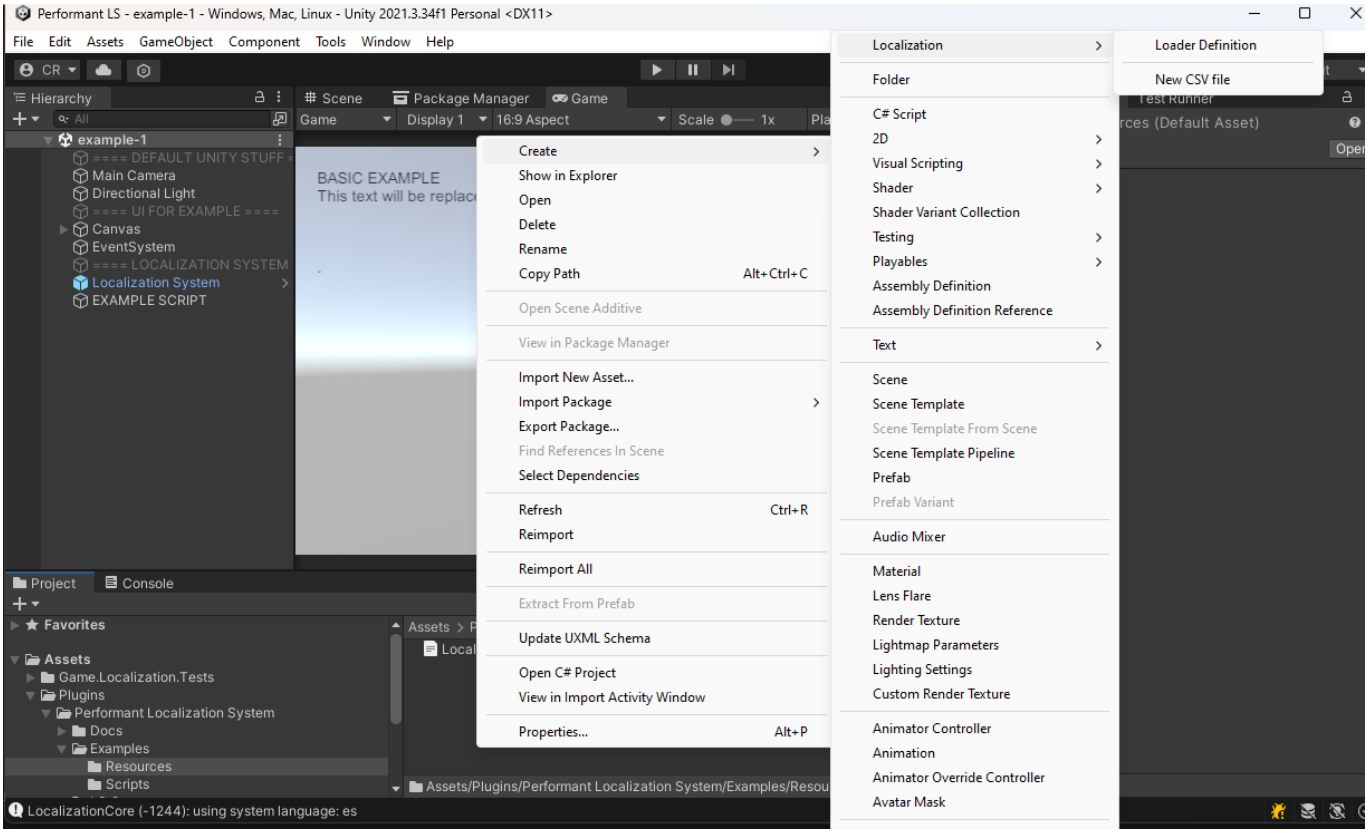
---

2. Create Translation File(s)

Generate the translation file(s) you need. It's recommended to place them in a subfolder within `Resources`.

To create a new file, simply navigate to a folder and do:

```
Right click > Create > Localization > New CSV file
```



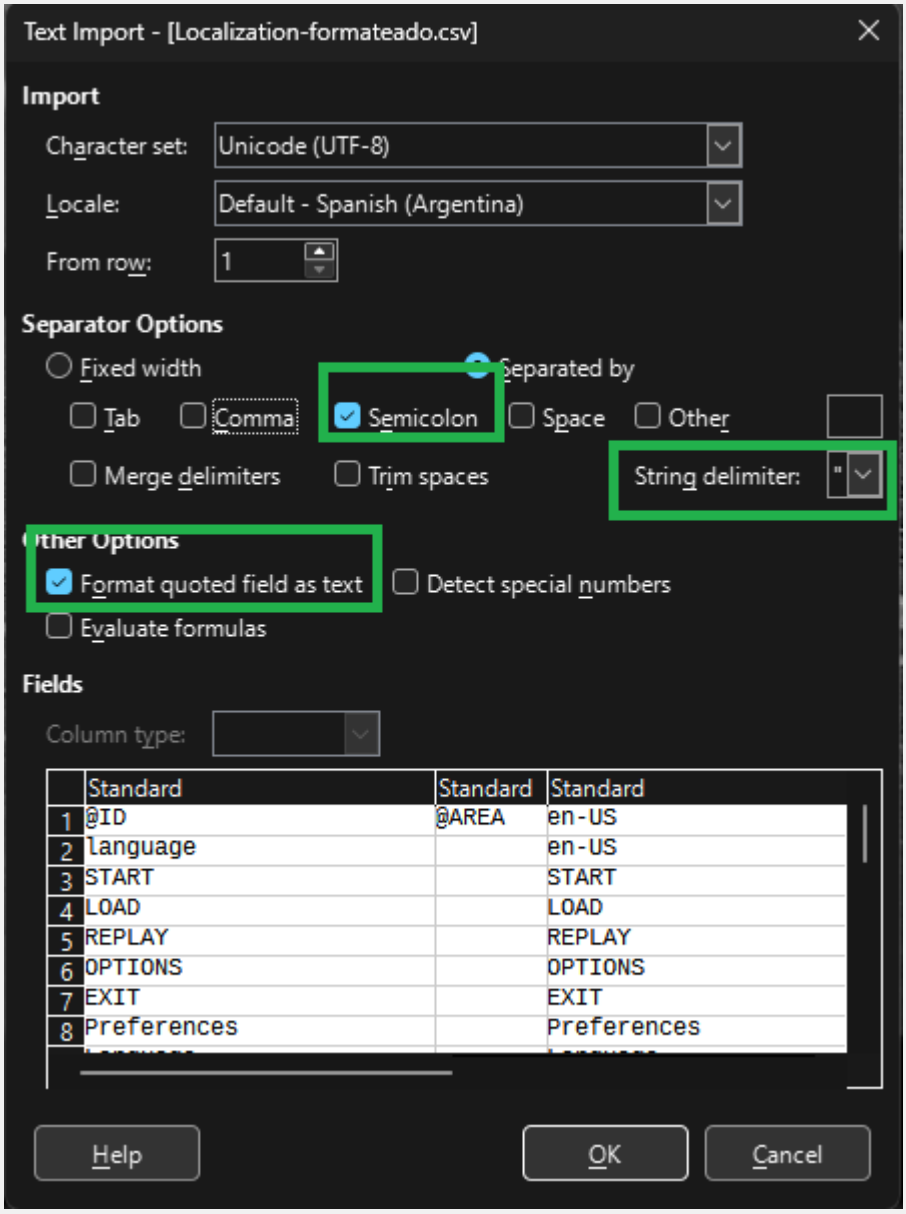
This will automatically generate a `.CSV` file with `UTF-8` encoding in your folder.

You can edit the file with any spreadsheet editor (recommended) or any text editor.

> We strongly recommend using `LibreOffice Calc`. It's *Open Source*, cross-platform, and works perfectly. For more information about `LibreOffice Calc`, please refer to the **Additional Resources** section.

The **default format** of .CSV files in PLS is: cells enclosed in double quotes ("), with cells separated by a semicolon (😉.

Libre Office: Recommended configuration



.

Inside the file, use the first column for keys and the subsequent columns for different languages.

Column headers are defined in the first row. The language should be specified using ISO 639-1 codes.
Examples: `en`, `en-US`, `en-GB`, `es`, `es-ES`, `es-AR`.

CSV File of Example 1

A	B	C	D
@ID	@AREA	en-US	es-AR
language		English	Español
BASIC EXAMPLE			EJEMPLO BÁSICO
In this example, the language will change every five seconds. You can see the code for this in the script 'Example1Script.cs'.			En este ejemplo se cambiará el idioma cada cinco segundos. Pueden ver el código del mismo en el script 'Example1Script.cs'.
Hello world!			¡Hola mundo!
Selected language: {0}			Idioma elegido: {0}
Language code: {0}			Código del idioma: {0}

.

---

3. Load a Language

This step is optional. If you leave the `autoLoadSystemLanguage` option set to `true` in the `LocalizationCoreScript`, it will automatically detect the system language and choose accordingly.

Manual Language Loading

If you prefer to load the language manually, you should disable `autoLoadSystemLanguage` and then call the `LoadLanguage` method.

Example:

```
LocalizationCoreScript.Instance.LoadLanguage("en");
```

> Note: If you have defined the language "en-US" in your CSV file, you can still call it as LoadLanguage("en"). The Performant Localization System will find it without any issues.

---

4. String Localization

In step 4, we proceed to translate the strings you need.

4.1. Extension Methods

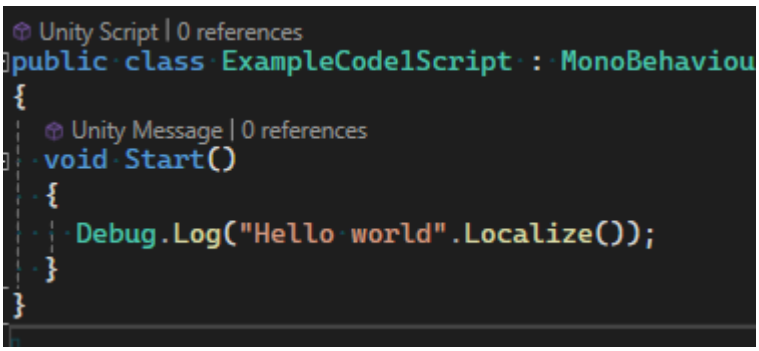
`Extension Methods` are the best way to localize strings while keeping your code clean. These methods extend the `string` class to easily access its translation.

To use these extension methods, make sure to include the following `using` directive at the beginning of your script:

```
using PLS.Extensions;
```

This will allow you to use the localization methods seamlessly in your code.

To localize strings, in most cases, you simply need to call the `Localize()` extension method.



We recommend using the texts to be translated as keys instead of hardcoded keys like "hello-world". This way, your code will be more natural and easier to read.

List of Extension Methods

- `Localize()`
  - Attempts to translate the text; if the translation is not found, it returns the original text.
- `LocalizeOrDefault(defaultValue)`
  - Attempts to translate the text; if the translation is not found, it returns the value specified in the `defaultValue` parameter.
- `LocalizeOnlyIfExists()`
  - Attempts to translate the text; if the translation is not found, it returns `null`.
- `LocalizeIfKeyLengthGT(lengthLimit)`
  - Performs the translation only if the text exceeds the length specified in the `lengthLimit` parameter. This is useful for not translating controller buttons like `A`, `B`, `X`, `Y`, etc.
- `LocalizedReplace(key)`
  - Attempts to translate all occurrences of `key` in the input text. This uses the `String.Replace` method, so it may generate some garbage.
- `SmartLocalize()`
  - This is the most resource-intensive translation method. It is recommended to use it sparingly. `SmartLocalize` tries to translate a paragraph "intelligently" (using an algorithm) with everything available in the dictionary. As such, it comes with a cost.

4.2. Direct Calls to LocalizationCoreScript

While we recommend using the methods described in `Section 4.1`, you can also call the functions directly from the `LocalizationCoreScript`.

This script also includes additional properties, which allow you to, for example, check the currently loaded language.

```
7 references
public interface ITranslator
{
    6 references
    bool IsLoaded { get; }

    4 references
    string Language { get; }

    1 reference
    string NormalizedLanguage { get; }

    3 references
    List<string> Languages { get; }

    11 references
    void LoadLanguage(string language);

    1 reference
    void LoadLanguageByZone(string language, params string[] zones);

    9 references
    string Localize(string key);

    3 references
    string LocalizeOnlyIfExists(string key);

    4 references
    string LocalizeOrDefault(string key, string defaultValue);

    5 references
    string Replace(string textToLocalize, string id);

    6 references
    string SmartLocalize(string text);
}
```

4.3. Scripts

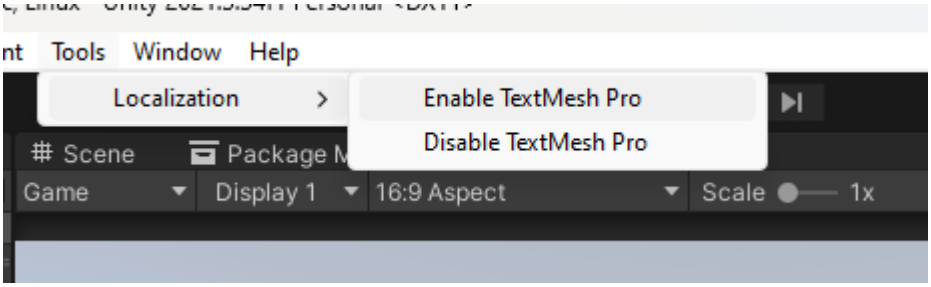
- AudioTranslationScript
- EnableObjectByLanguageScript
- LanguageChangerScript
- TextTranslatorScript

4.4. Plugins / Auto-Localization Scripts

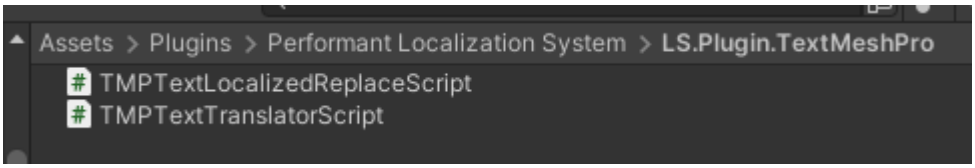
In the `LS.Plugin.TextMeshPro` folder, you'll find scripts for automatically translating interfaces using TextMeshPro.

To use these scripts, you need to have `TextMesh Pro` installed in your project.

After installing it, go to `Tools > Localization > Enable TextMesh Pro`.



This will allow you to use the following scripts:



Scripts

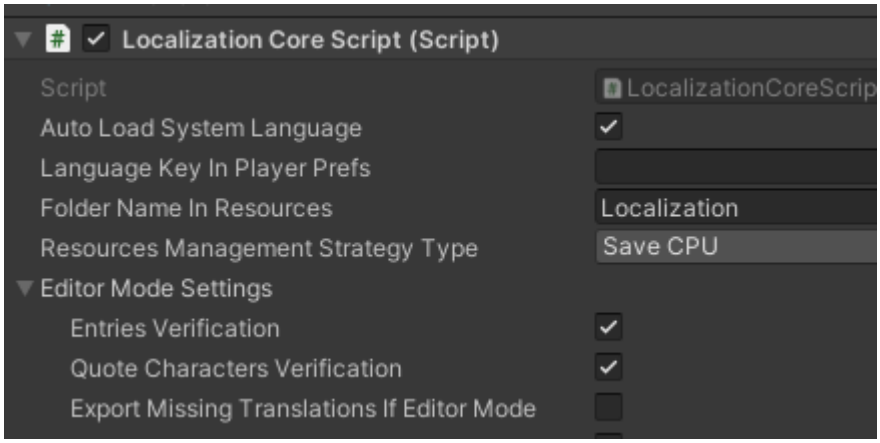
- `TMPTextLocalizedReplaceScript`
  - Automatically translates using the `LocalizeReplace` method.
- `TMPTextTranslatorScript`
  - Automatically translates using the `Localize` method.

Migration

Migrating to the Performant Localization System (PLS) is designed to be simple and non-intrusive. Our main goal is to ensure that integrating the translation system into your existing project is straightforward and does not compromise code readability. Here are some steps and tips to assist you with the migration:

1. **Enable Missing Translations Export**

- You can enable the option `Export Missing Translations If Editor Mode`. This will generate a `.CSV` file in the `Assets` folder each time a `Localize` method call does not find a translation.



- The best part is that this does not affect the final performance of your game in production, as it is only enabled while you are in the Unity editor.

2. **Identify Translation Points**

- Start by identifying the parts of your code where text localization is needed. This can include user interface elements, in-game dialogues, and any other textual content.

3. **Prepare Translation Files**

- Create and organize your translation files. Ensure that each language and its respective translations are correctly formatted and saved in a subfolder within the `Resources` folder of your project.

4. **Update Your Code**

- Replace hardcoded strings with calls to the localization methods provided by PLS.

- Example before migration:

```
myTextComponent.text = "Hello, World!";
```

Example after migration:

```
myTextComponent.text = "Hello, World!".Localize();
```

5. **Test Your Localization**

- After updating your code, thoroughly test the project to ensure that all text is correctly translated. Remember to check different languages.

6. **Optimize and Refine**

- Review your implementation to look for optimization opportunities.

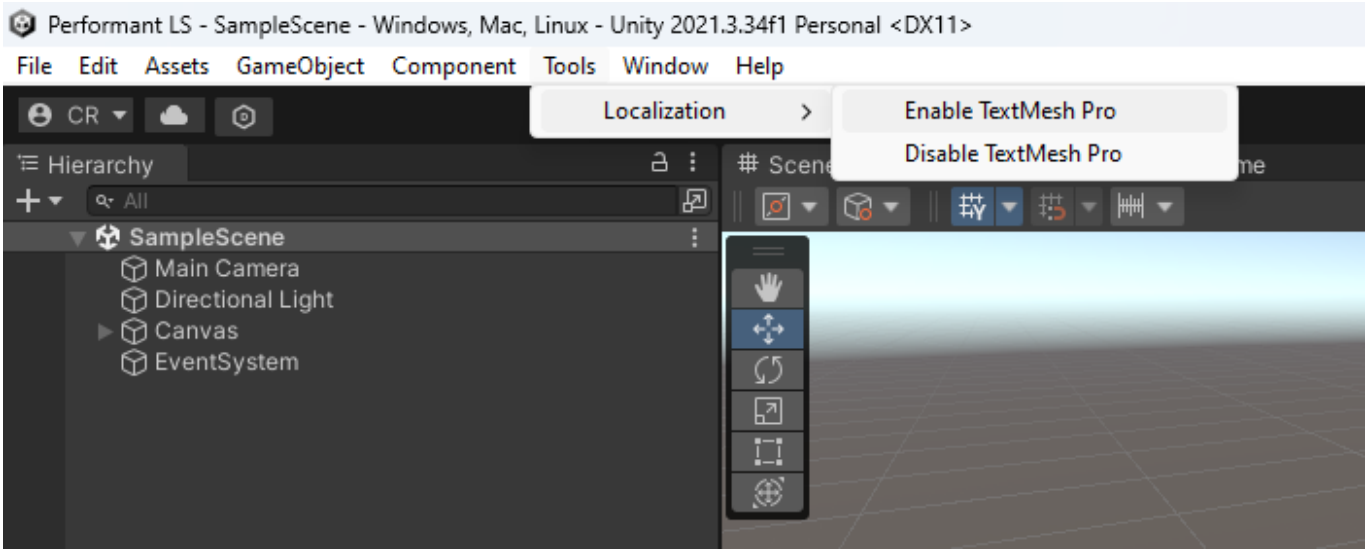
Additional Tips

- **Backup Your Project**: Before starting the migration, make sure to back up your project to avoid losing any existing work.
- **Incremental Migration**: Consider migrating in stages to handle changes more effectively and catch issues early.
- **Seek Support**: If you encounter challenges, don't hesitate to reach out to the support community or consult the documentation for help.

Plugins

TextMesh Pro

To enable the scripts that interact with TextMesh Pro, go to the `Tools > Localization > Enable TextMesh Pro` menu.



FAQ

Is this system a new version of "Localization System"?

This system is completely distinct from our previous "Localization System".

The "Localization System" asset has been evolving for many years. It is a robust tool that has been used by various studios, including in AAA games.

Performant Localization System, on the other hand:

- Is a completely **new and different** localization solution.
- Is designed to minimize **garbage** generation.
- Aims to ensure **clean code-based translations**.
- **Only supports CSV format**.

References

- Asset store page: https://assetstore.unity.com/publishers/1881
- Official website: https://forjagames.com
- Youtube channel: https://youtube.com/@ForjaGames
- Support (discord): https://discord.com/invite/dqGdSpVcAc

Additional Resources

- LibreOffice official website: https://www.libreoffice.org

Support

You can get real-time support through our Discord:

https://discord.com/invite/dqGdSpVcAc

If you like this Asset, please remember to rate it. Your feedback is very important to us and helps us grow.

Thank you very much!