Documentación Proyecto de Traducción

Visión General

Este proyecto aborda la necesidad de gestionar traducciones de textos en múltiples idiomas de manera centralizada y eficiente. Es útil en aplicaciones que requieren soporte multilingüe, como software educativo, videojuegos, y plataformas web.

El objetivo es facilitar la adición, edición y persistencia de traducciones, garantizando que los textos estén disponibles en varios idiomas sin redundancia ni inconsistencias.

El modelo de datos está basado en dos clases principales: Text y TextTranslations. La clase Text representa un texto traducido en varios idiomas, utilizando un mapa (translationsMap) para almacenar las traducciones, donde las claves son los códigos de idioma (ISO-639) y los valores son las traducciones. La clase TextTranslations gestiona múltiples Text identificados por un ID único (textsTranslations). Esta estructura permite organizar y acceder fácilmente a las traducciones por su ID y código de idioma, garantizando una rápida recuperación y almacenamiento eficiente.

El proyecto está dividido en dos partes o núcleos importantes y relacionados, un bloque grande es la parte del manejo de datos y backend, en formato de librería dinámica (DLL) y el otro bloque es la parte del manejo de entrada de datos y apartado visual del uso de esos datos, en formato de proyecto del motor Unity.

Primero veamos la DLL

La DLL de traducción de textos gestiona las operaciones de traducción de texto, como cargar y guardar traducciones desde y en archivos CSV. La DLL está compuesta por varias clases para proporcionar una funcionalidad de traducción y manejo de archivos.

La DLL de traducción de textos permite gestionar traducciones a través de un flujo sencillo:

Primero se crea una instancia de TextTranslations, luego se añaden traducciones utilizando addText. Estas traducciones se pueden guardar en un archivo CSV con saveTextsToCSV y cargar con loadTextsFromCSV. Las traducciones específicas se recuperan con getTextTranslation y todas las traducciones con GetAllTextTranslations.

Finalmente, la instancia se destruye con deleteEditor. Este flujo facilita la gestión centralizada y persistente de traducciones de textos en diferentes idiomas.

Clases Principales

1. TextTranslations
    a. Es la encargada de gestionar las traducciones de textos. Permite almacenar y recuperar traducciones para múltiples textos identificados por un ID único, añadir traducciones a textos, obtener traducciones de textos en idiomas específicos y recuperar todas las traducciones almacenadas.

2. Text
    a. Representa un texto con sus traducciones en diferentes idiomas. Gestiona las traducciones y permite recuperarlas, añadir una nueva traducción para un idioma específico, obtener una traducción en un idioma específico, recuperar todas las traducciones para el texto.

3. CsvSerialization
    a. Se encarga de guardar las traducciones en un archivo CSV. Guarda las traducciones en un archivo CSV, escribiendo primero la cabecera con los códigos de idioma y luego las traducciones correspondientes para cada texto.

4. CsvDeserialization
    a. Se encarga de cargar las traducciones desde un archivo CSV. Carga las traducciones desde un archivo CSV, leyendo primero la cabecera para obtener los códigos de idioma y luego las traducciones correspondientes para cada texto.

5. ExportDLLFunctions
    a. Se encarga de exportar los métodos de las distintas clases de la DLL para poder luego importadas en Unity, estas funciones/métodos son:
    1. createEditor y deleteEditor
        i. Estos métodos permiten crear y destruir instancias de TextTranslations.
        ii. createEditor(): Crea una nueva instancia de TextTranslations.
        iii. deleteEditor(): Destruye una instancia de TextTranslations.

    2. addText
        a. Este método añade una traducción a un texto específico identificado por su ID.
        b. addText(TextTranslations* editor, const char* textId, const char* languageCode, const wchar_t* translation): Añade una nueva traducción a un texto en un idioma específico.

    3. getTextTranslation
        a. Este método obtiene la traducción de un texto específico en un idioma específico.
        b. getTextTranslation(TextTranslations* editor, const char* languageCode, const char* textId): Recupera la traducción de un texto en un idioma específico.

4. GetAllTextTranslations
    a. Este método obtiene todas las traducciones de textos almacenadas en la instancia de TextTranslations.
    b. GetAllTextTranslations(TextTranslations* editor, TranslationEntry* entries, int maxCount): Recupera todas las traducciones de textos hasta un máximo especificado.

5. saveTextsToCSV y loadTextsFromCSV
    a. Estos métodos permiten guardar y cargar traducciones desde y hacia archivos CSV.
    b. saveTextsToCSV(const char* filePath, TextTranslations* editor, const char** languageCodes, int languageCount): Guarda las traducciones en un archivo CSV.
    c. loadTextsFromCSV(const char* filePath, TextTranslations* editor): Carga las traducciones desde un archivo CSV.

Ahora veamos la parte de Unity

La parte de Unity gestiona la interfaz de usuario y las interacciones del usuario con las traducciones de texto. Se conecta con la DLL de traducción de textos para cargar, editar y guardar traducciones. La interfaz de Unity está compuesta por varias clases que proporcionan la funcionalidad necesaria para manejar las traducciones y su visualización.

La integración entre Unity y la DLL de traducción sigue un flujo sencillo: Primero, se inicializa la interfaz de usuario y se establecen las conexiones con los botones y campos de entrada. Luego, se pueden cargar las traducciones desde un archivo CSV utilizando LoadFromCSV, y se muestran en la interfaz. Las traducciones se pueden editar directamente en la interfaz y guardar en un archivo CSV utilizando SaveToCSV. Las traducciones específicas se añaden con AddNewRow. Este flujo facilita la gestión interactiva y persistente de traducciones de textos en diferentes idiomas.

Clases Principales

1. UIManager
    a. Esta clase se encarga de gestionar la interfaz de usuario relacionada con la carga, visualización, edición y guardado de traducciones desde y hacia archivos CSV.

    b. Atributos:
        i. buttonAdd: Botón para añadir una nueva fila de traducción.
        ii. buttonSave: Botón para guardar las traducciones actuales en un archivo CSV.
        iii. buttonLoad: Botón para cargar traducciones desde un archivo CSV.
        iv. buttonAddLanguage: Botón para añadir un nuevo idioma a la tabla de traducción.

v. statusText: Campo de texto para mostrar el estado de las operaciones (e.j., "Archivo cargado correctamente").
vi. tableContent: Contenedor que almacena las filas de traducción.
vii. rowPrefabGO: Prefab que representa una fila de traducción.
viii. inputLanguage: Campo de texto para ingresar el código de idioma.

c. Métodos:
i. LoadFromCSV(): Carga las traducciones desde un archivo CSV.
ii. SaveToCSV(): Guarda las traducciones actuales en un archivo CSV.
iii. AddNewRow(string textId, List<string> translations): Añade una nueva fila de traducción a la interfaz.
iv. UpdateTable(): Actualiza la tabla de traducciones con los datos cargados desde el CSV.

2. TranslationEditor
a. Esta clase actúa como un puente entre Unity y la DLL. Se encarga de llamar a los métodos de la DLL para cargar y guardar traducciones.

b. Atributos:
i. TextTranslations editor: Instancia de la clase TextTranslations de la DLL.

c. Métodos:
i. LoadFromCSV(string filePath): Llama a la función de la DLL para cargar traducciones desde un archivo CSV.
ii. SaveToCSV(string filePath, List<string> languageCodes): Llama a la función de la DLL para guardar traducciones en un archivo CSV.
iii. AddTranslation(string textId, string languageCode, string translation): Añade una nueva traducción utilizando la DLL.
iv. GetTranslation(string textId, string languageCode): Obtiene una traducción específica utilizando la DLL.

Flujo de la Parte Visual / Usuario

Inicialización:

Al iniciar la aplicación, UIManager se encarga de inicializar los botones y campos de texto. Los botones se configuran para llamar a los métodos correspondientes (e.j., buttonLoad.onClick.AddListener(LoadFromCSV)).

Al presionar el botón de cargar (buttonLoad), se llama al método LoadFromCSV de UIManager. LoadFromCSV utiliza TranslationEditor para cargar las traducciones desde un archivo CSV. TranslationEditor llama al método loadTextsFromCSV de la DLL, que procesa el archivo CSV y devuelve las traducciones. UIManager recibe las traducciones y llama a UpdateTable para mostrar los datos en la interfaz.

Al presionar el botón de añadir fila (buttonAdd), se llama al método AddNewRow de UIManager. AddNewRow crea una nueva fila en la tabla utilizando el prefab rowPrefabGO y la añade al contenedor tableContent. Cada nueva fila incluye campos de entrada para el ID del texto y las traducciones en diferentes idiomas.

Al presionar el botón de guardar (buttonSave), se llama al método SaveToCSV de UIManager. SaveToCSV utiliza TranslationEditor para guardar las traducciones en un archivo CSV. TranslationEditor llama al método saveTextsToCSV de la DLL, que guarda las traducciones en el archivo especificado.