Table des matières

I – Définition

LanguagesFx est un module conçut pour la gestion des langues d'un jeu. Elle assure le chargement des différents langages du jeu. Ces langues sont décidées par le développeur. En d'autres termes, c'est le développeur lui-même qui intègre les différentes langues dont-il pourra charger grâce à ce module. Cela lui permettra ainsi de prendre en charge autant de langues que possible.
NB : Notez que ce module est de nature indestructible, est compatible à un jeu 2D, 3D et est sauvegardable.

II – Les propriétés disponibles

+ int TargetPath = 0 : Contient les différents chemins que prend en charge ce module. Ces chemins représentent les endroits possibles où l'on peut accédé au fichier (.csv). Les valeurs possibles sont celles définient au sein de la méthode *get_os_dir ()* de la classe **MegaAssets**. Le champ d'activité de ce champ est uniquement sur le moteur Godot.

+ String Input : Contient le chemin pointant vers le fichier de langue à prendre charge. L'extention de ce fichier doit être en (.csv). Le champ d'activité de ce champ est uniquement sur le moteur Godot.

+ String Separator = 0 : Contient les différents séparateurs que supporte ce module. Le champ d'activité de ce champ est uniquement sur le moteur Godot. Les valeurs possibles sont :
-> **LanguagesFx.Divider.COMMA ou 0** : Le fichier sera traité avec les **virgules**.
-> **LanguagesFx.Divider.SEMICOLON ou 1** : Le fichier sera traité avec les **points virgule**.
-> **LanguagesFx.Divider.TAB ou 2** : Le fichier sera traité avec les **tabulations**.

+ String Output : Contient le chemin pointant vers le(s) fichier(s) de langue au format (.translation).

+ bool Security = true : Voulez-vous mettre une sécurité sur le(s) fichier(s) de langue ?

+ int Level = 1 : Contient les différents niveaux de sécurité possibles pour la création de(s) fichier(s) au format (.translation). Les valeurs possibles sont celles définient au sein de la méthode *serialize ()* de la classe **MegaAssets**.

+ String Pass : Contient le mot de passe à utilisé pour sécuriser les données de langue. Vous avez la possibilité de générer automatiquement un mot de passe via le booléen *GeneratePassword*.

+ int | String ActiveLanguage : Contient la langue à chargée. Notez que les valeurs que vous enterez ici, devront prendre en charge les éléments qui seront générés dans le *Output*. Vous avez la possibilité de cibler une langue en donnant le nom correspondant à son fichier.

+ **bool AutoGeneration = false :** Souhaitez-vous générer de façon automatique les correspondances (.translation) du fichier (.csv) ? Notez que cette option, une fois activée, met en écoute l'édition du fichier source en vue de générer en temps réel, les équivalents (.translation). Vous ne serez donc plus tenu d'appuyer sur le bouton *GenerateLangFile(s)* à chaque fois que vous progresserez dans l'édition de votre source. Toute fois, désactivé cette fonctionnalité avant chaque exécution du jeu pour éviter des écoutes inutiles. Le champ d'activité de cette propriété est uniquement sur le moteur Godot.

+ **float Interval = 3.0 :** Quel est le temps mort avant chaque génération ? Notez que la valeur minimum de ce champ est de 0.1 secondes et ne s'active que lorsque l'option *AutoGeneration* est activé. Le champ d'activité de cette propriété est uniquement sur le moteur Godot.

+ **bool GeneratePassword = false :** Souhaitez-vous générer automatiquement un mot de passe ? A ce niveau, la valeur du champ *Passs* sera mise à jour à chaque génération. Le champ d'activité de cette propriété est uniquement sur le moteur Godot.

+ **bool GenerateLangFile(s) = false :** Bouton générant les équivalents (.translation) du fichier source (.csv). Le champ d'activité de cette propriété est uniquement sur le moteur Godot.

III – Les méthodes disponibles

+ **void generate_passwords (delay = 0.0) :** Génère au hazard, un mot de passe en fonction des configurations présentes à son niveau. Le(s) élément(s) contenant la clé *password* seront affecté(s) par cette génération. Attention ! Cette méthode ne s'exécute qu'en mode édition.
  » **float delay** : Quel est le temps mort avant la génération ?

+ **void generate_lang_files (delay = 0.0) :** Génère les équivalents (.translation) du fichier source (.csv) ayant été précisé. Attention ! Cette méthode ne s'exécute qu'en mode édition.
  » **float delay** : Quel est le temps mort avant la génération ?

+ **String get_value_at (key, default = "Null") :** Renvoie la valeur associée à un identifiant donné en fonction du langage actif.
  » **String key** : Contient l'identificateur de la valeur à retournée.
  » **String default** : Quelle valeur doit-on retournée en cas de problème ?

+ **Dictionary | String get_language_data (json = true) :** Renvoie toutes les données associées au langage actif.
  » **bool json** : Voulez-vous renvoyer le résultat sous le format json ?

+ **bool has_keys (keys) :** Détermine si un ou plusieurs identifiant(s) sont bel et bien définient dans le gestionnaire de langue.
  » **String | PoolStringArray keys** : Contient tous le(s) identifiant(s) à cherché(s).

+ **PoolStringArray get_language_list () :** Renvoie les noms de toutes les langues prises en charge dans le module.

+ **void reload_language (delay = 0.0) :** Recharge dans le gestionnaire des langues du jeu, les données du language actif.
  » **float delay** : Quel est le temps mort avant le rechargement ?

IV – Les événements disponibles

+ **language_changed (data) :** Signal déclenché lorsque les données de langue ont changées.
  » **Dictionary data** : Contient les données du langage actuellement chargé en mémoire.

+ **lang_file_cant_open (data) :** Signal déclenché lorsqu'on a du mal à ouvrir un fichier de langue ou que son accès à été refusé. Cet événement renvoie un dictionaire contenant les clés suivantes :
  » **String path** : Contient le chemin pointant vers le fichier de langue.
  » **int type** : Contient le type de l'erreur déclenché.
  » **String message** : Contient le méssage renvoyé par l'erreur déclenché.

+ **lang_file_not_found (data) :** Signal déclenché lorsqu'un fichier de langue n'est pas définit. Cet événement renvoie un dictionaire contenant les clés suivantes :
  » **String path** : Contient le chemin pointant vers le fichier de langue.
  » **int type** : Contient le type de l'erreur déclenché.
  » **String message** : Contient le méssage renvoyé par l'erreur déclenché.

+ **lang_file_corrupted (data) :** Signal déclenché lorsqu'un fichier de langue a été corromput de l'extérieur. Cet événement renvoie un dictionaire contenant les clés suivantes :
  » **String path** : Contient le chemin pointant vers le fichier de langue.
  » **int type** : Contient le type de l'erreur déclenché.
  » **String message** : Contient le méssage renvoyé par l'erreur déclenché.

+ **lang_file_loading (data) :** Signal déclenché pendant qu'un fichier de langue est en cours de chargement. Cet événement renvoie un dictionaire contenant les clés suivantes :
  » **String path** : Contient le chemin pointant vers le fichier de langue.
  » **bool is_over** : Est-ce que toutes les données du fichier de langue ont-elles été complètement chargées ?
  » **int progress** : Contient le nombre total de donnée(s) déjà chargée(s) en mémoire.