

Textes, localisation et mise à jour

Pipeline

```
flowchart LR
  S[Google Sheets] --> I[importUploadTexts.js]
  I --> JSON[shared_texts / gameId_texts]
  JSON --> UNITY[TextsManager]
  JSON --> WEB[Upload serveur web]
  WEB --> API[/getTexts/gameId/lang/timestamp]
  API --> UNITY
  UNITY --> TL[TextLoader / UI]
```

Fichiers générés

- partagés : `Assets/Resources/data/shared_texts_fr.json` et `shared_texts_en.json` ;
- jeu : `Assets/GameSpecific/<gameId>/Resources/data_<gameId>/<gameId>_texts_fr.json` et EN.

Import Node

`exports/importUploadTexts.js` associe chaque gameId à ses feuilles Google et appelle `importer.importTexts()`. Les commandes exposées permettent l'import game-specific, shared, ou l'import suivi de l'upload.

Pour Accounting, le script traite Acc1 et Acc2 ensemble via `accountingTextGameIds = ["acc1", "acc2"]`, afin d'éviter de mettre à jour une seule variante.

Chargement Unity

`TextsManager.Initialize()` charge les textes shared. `Game.SetGameUid()` appelle `LoadGameSpecificTexts()`. `SetLang()` recharge les deux couches puis appelle `TextLoader.ReloadAll()`. Un UID absent retourne `[uid]`.

Mise à jour sans rebuild

`UpdateTextsFromWebServer()` interroge `/getTexts/<gameUid>/<locale>/<timestamp>`. Le client garde timestamp/hash et peut donc recevoir des textes plus récents que ceux embarqués dans le build.

Modifier un texte

1. Trouver l'UID utilisé par `TextLoader` ou `Game.Texts.GetText()`.
2. Modifier la source Google Sheets appropriée.
3. Lancer l'import.
4. Vérifier FR + EN.
5. Publier si le changement doit être disponible sans rebuild.
6. Sinon inclure les JSON générés dans le prochain build Unity.

Procédure exacte pour ajouter ou modifier un texte

1. Choisir un UID stable; ne pas utiliser la phrase FR elle-même comme identifiant.
2. Ajouter/modifier FR et EN dans la Google Sheet de textes appropriée.
3. Depuis `node.js/exports/`, lancer `importTexts.bat` et entrer le GameUid, ou `shared` pour les textes communs.
4. Inspecter les JSON générés.
5. Si le texte doit être disponible sans rebuild, lancer `importThenUploadTextsToUniksimServer.bat`.
6. Tester le changement de langue dans un client déjà ouvert via `TextsManager.SetLang()`.

Commandes directes équivalentes

```
node importUploadTexts.js shared
node importUploadTexts.js game_specific mk1
node importUploadTexts.js uploadTexts mk1
```

Cas Accounting

`getGameUidsToImport()` traite Accounting comme une paire : un import de textes Acc1/Acc2 met à jour les deux jeux concernés. Ne pas maintenir deux traductions divergentes par erreur.

Choisir shared ou game-specific

- texte réutilisé par plusieurs jeux/UI commune → shared;
- vocabulaire/règle propre à un jeu → game-specific;
- ne pas copier un même UID dans plusieurs couches juste pour contourner un problème de chargement.

Debug d'un UID affiché comme [my_uid]

1. Vérifier que l'UID exact existe dans la Sheet.
2. Vérifier que le JSON correspondant a été régénéré.
3. Vérifier shared vs `data_<gameUid>`.
4. Vérifier la locale courante.
5. Si le serveur web doit fournir la mise à jour, vérifier l'upload et le timestamp/hash.
6. Si le texte est dans Resources seulement, rebuild Unity.

Revision #2

Created 2026-09-18 19:10:08 UTC by Mathis

Updated 2026-09-18 20:09:25 UTC by Mathis