**settings.py**

**Provenance :**

Création entièrement originale.

**Utilités :**

* Importer un dossier d’image
* Importer des dialogues depuis un fichier texte
* Importer un carte

**Fonctionnement :**

La première fonction, « ImportFolder », est la plus simple : on donne en paramètre un chemin vers un dossier d’image et la fonction renvoie un dictionnaire d’images associées à leur nom de fichier.

La seconde fonction, « ImportSpeech », est plus complexe. Elle permet d’interpréter des fichiers texte dans lesquels sont écrits des dialogues. Voici un exemple de fichier texte de dialogues :

Une image contenant texte, capture d’écran, Police, nombre

Description générée automatiquement

 Nous avons décidé d’utiliser une syntaxe ressemblant à du CSS pour simplifier l’interprétation du fichier. En effet, la manière la plus simple que nous avons trouvée pour séparer les différents éléments du fichier est l’utilisation des expressions régulières : Regex.

* Grâce aux Regex, on récupère d’abord tous les groupes de dialogues du fichier  : { … }.
* Ensuite pour chaque groupe de dialogues, on récupère les conditions dans lesquelles il faut faire apparaître ce groupe de dialogue (lieu et étape du jeu)





* Puis on récupère tous les dialogues du groupe de dialogue : […].



* Puis pour chaque ligne du dialogue, on récupère le nom de la personne qui la prononce et le texte dit



* Enfin, on vérifie si l’on veut avancer dans l’histoire après le dialogue ou si l’on veut donner un indice au joueur





Et on crée un groupe de dialogues avec toutes les caractéristiques désirées :



Enfin, la méthode « loadMap » permet d’importer des cartes également enregistrées sous la forme de fichiers texte. On commence donc par séparer le fichier en couches de blocs, puis en lignes et puis en blocs : le fichier est segmenté par des caractères spécifiques comme « $ », « | » et « / » (voir l’enregistrement des cartes dans le fichier technique de « map\_editor »).

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

Ensuite, on récupère les caractéristiques de chaque bloc : nom / s’il fait partie d’une image composée de plusieurs blocs / s’il est affiché devant le joueur / s’il a des collisions / s’il a un carte ou des dialogues associés.

Si le bloc fait partie d’une image composée de plusieurs blocs, alors on affiche directement l’image entière pour des gains de performance.

Dans tous les cas, les collisions du blocs sont rajoutées au groupe d’obstacles.

Si le bloc mène à une autre carte, on l’enregistre, et si le bloc a des dialogues associés, on les importe avec la fonction « ImportSpeech ».

Enfin, pour chaque carte, on retourne un dictionnaire comportant le ground (la couche inférieur, sur laquelle on marche), le groupe des éléments visibles (tout élément n’appartenant pas à la 1e couche), le groupe des obstacles, le groupe des blocs interactifs et les dernières coordonnées du joueur sur la carte (utilisées lors d’un changement de carte).

Les classes « Obstacle », « Tile » et « InteractTile » définissent les trois types de blocs précédemment abordés.