Simulateur de mouton de Charpy

1. Présentation générale

Le simulateur de mouton de Charpy vous permet de simuler les caractéristiques nécessaires d'un pendule pour permettre de casse une éprouvette. Tout comme un véritable mouton de Charpy, vous pourrez simuler la réaction du pendule lorsqu'il casse un pendule.

1. Cahier des charges
2. Description du problème

L’objectif de l’algorithme est donc de résoudre l’équation d’un pendule simple non linéaire. Notre pendule doit être le plus réaliste possible, c’est pourquoi les frottements ont été pris en compte, tous les paramètres doivent être paramétrable. On modélise un pendule de masse m, de longueur l, une vitesse angulaire initiale v avec un coefficient de frottement f. E On note θ l’angle de notre pendule. En utilisant le principe fondamental de la dynamique on obtient l’équation différentielle suivante :

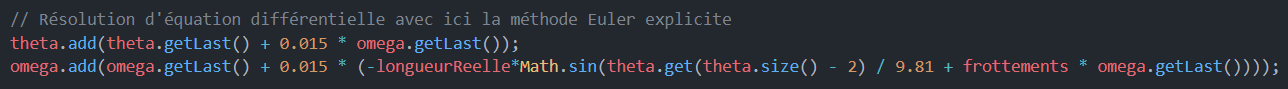
Avec des conditions initiales définies par l’utilisateur

Le programme doit pouvoir écrire un compte-rendu de simulation ainsi que pouvoir lire un fichier

1. Principe de l’algorithme
   1. Fonctionnement général
   2. Résolution de l’équation différentielle

La fonctionnalité principale du programme est donc la résolution d’une équation différentielle non linéaire de second ordre.

Pour cela, nous avons utilisé un schéma Euler explicite. Ce schéma nécessite un pas de temps faible pour fonctionner, nous avons choisi ici un temps de 15 ms, soit une fréquence de 60 Hz. Le code est le suivant :

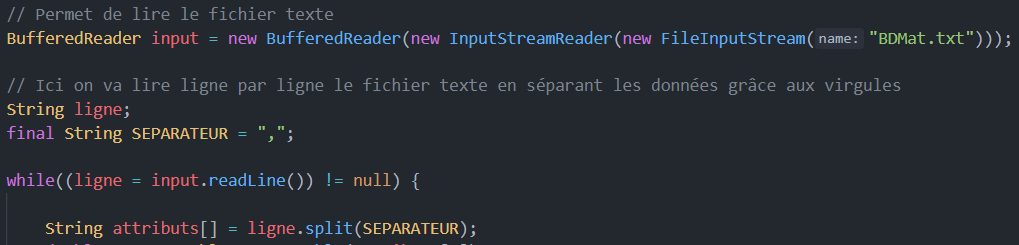


Theta correspond à l’angle du pendule et omega à la vitesse angulaire. La méthode d’Euler est une méthode par incrémentation qui fonctionne sur des approximations, c’est pour cela que le temps entre deux calculs doit être faible.

Theta et omega sont des LinkedList, nous avons fait que choix car il y a beaucoup d’ajout en fin de liste, on ne parcoure qu’une seule fois la liste en entier.

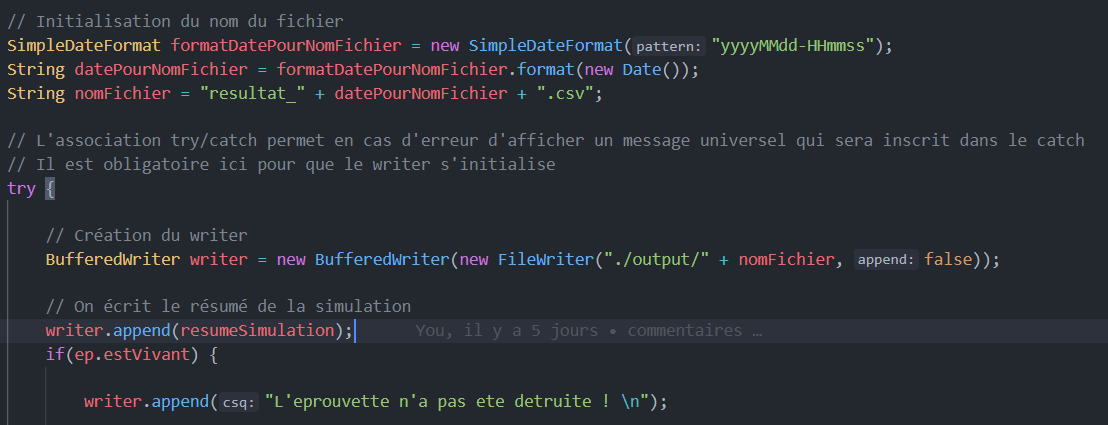
* 1. Lecture du fichier de la base de données de matériaux

Les matériaux créer sont présent dans un fichier texte. Cependant, il faut que le programme puisse lire ces données, pour cela, nous avons utilisé un BufferedReader. Le code est le suivant :



On créer un tunnel de connexion jusqu’au fichier. Le fichier est formaté de sorte à ce que toutes les données soient séparées par des virgules. On récupère donc ligne par ligne le fichier et on utilise la méthode .split() de la classe String pour séparer les données

* 1. Ecriture des résultats de la simulation

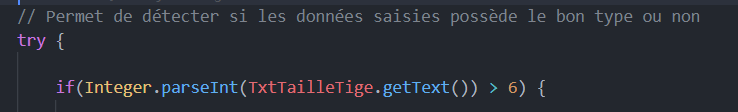


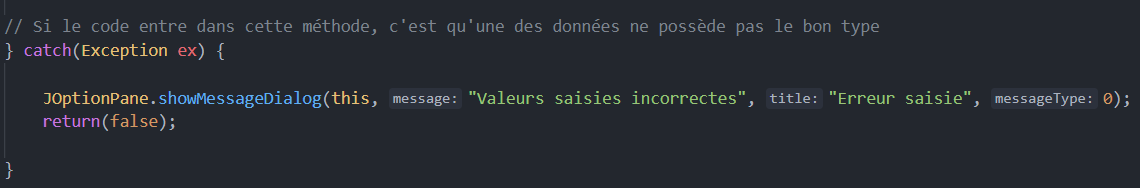
Pour écrire dans un fichier ou pour créer un fichier, il faut utiliser un BufferedWriter qui possède pour argument un FileWriter. Cela permet d’optimiser le programme et éviter certains bugs d’optimisation du FileWriter. Cependant, une des problématiques était de ne pas écraser les anciens fichiers. Le nom du fichier est donc généré automatiquement en fonction de la date et l’heure de la simulation. On définit donc un format de date pour pouvoir formater celle-ci de façon correcte.

Pour écrire une ligne, il suffit d’utiliser la méthode .append() de BufferedWriter.

* 1. Utilisation de try/catch

Dans beaucoup de méthode, nous avons utilisé l’association try/catch qui permet de réagir si des erreurs se produisent. Le fonctionnement est le suivant : on ouvre le try, si une erreur est détectée à l’intérieur de celui-ci, il va jeter l’erreur et lancer le code contenu dans le catch. Cela était obligatoire pour le BufferedReader et le BufferedWriter dans les cas ou les fichiers ne pouvait pas être créer ou être lu.



Nous l’avons utilisé à d’autre endroit comme par exemple lorsque vérifie que les données saisies sont correctes, dans le cas ou on souhaite récupérer un double, on doit utiliser Double.parseDouble(« text ») mais si le texte n’est pas un double, dans ce cas le code de catch va être exécuter.

1. Structuration des données
2. Suggestion d’amélioration et bugs connus

On utilise la méthode Euler explicite pour résoudre l’équation différentielle. Cependant, cette méthode possède quelques limites. La première est que cette méthode est une approximation, elle dépend donc du pas de temps. Or si nous choisissons un coefficient de frottement nul, le pendule a tendance à gagner de l’énergie mécanique ce qui n’est pas très réaliste.

De plus, le simulateur commence à faire des erreurs de calculs lorsque le pendule passe à la verticale.

Une suggestion d’amélioration serait donc d’utiliser un schéma de résolution plus complexe permettant de faire cette simulation. Nous avons décidé de ne pas mettre un autre schéma car cela n’était pas pertinent dans l’utilisation première de notre programme.

Une suggestion d’amélioration serait également de pouvoir revenir en arrière dans la simulation. Utiliser la simulation comme un lecteur vidéo, ce qui peut être très pratique dans certains cas.

Une autre suggestion serait que le programme est créant le compte-rendu de la simulation dessine directement la courbe de l’angle du pendule en fonction du temps.

Un autre bug connu est que l’on peut créer un matériau ayant pour nom uniquement des espaces.

Carnet de route

Répartition du travail dans le groupe

Figure : Une légende caractérisant cette figure ...