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 :

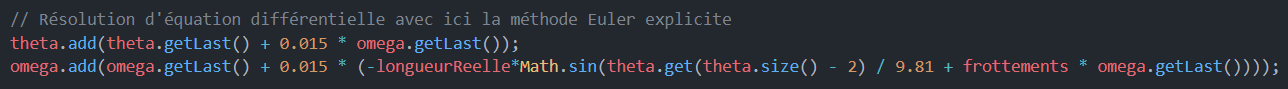
Avec des conditions initiales définies par l’utilisateur

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

1. Principe de l’algorithme
   1. 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 :

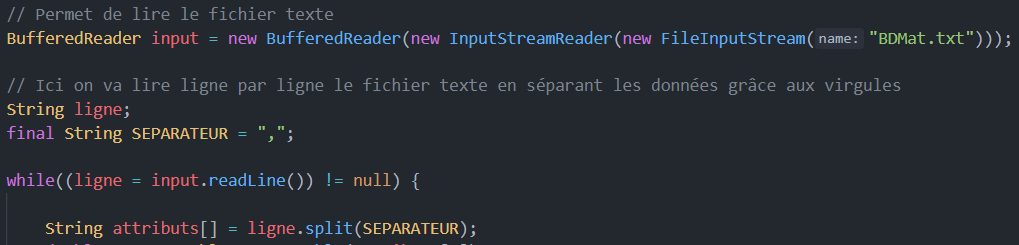


Theta correspond à l’angle du pendule et omega à la vitesse angulaire. La méthode d’Euler est une méthode par incrémentation qui fonctionne sur des approximations, c’est pour cela que le temps entre deux calculs doit être faible.

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

Structuration des données

Suggestion d’amélioration et bugs connus

Carnet de route

Répartition du travail dans le groupe

Figure 1 : Une légende caractérisant cette figure ...