Projet de Simulation robotique

Instructions pour le rendu

- Un compte rendu en pdf est obligatoire, le code tout seul n'est pas recevable. Il doit illustrer et décrire le travail réalisé, expliquer les choix techniques (quelle classe/méthodes, pourquoi – éviter quand même la répétition avec des choses vues en cours), mettre en valeur votre implémentation (surtout si différente de ce que j'avais suggéré), enfin décrire et illustrer les résultats. Inclure des schémas, courbes et captures d'écran. Un bon rapport se lit sans avoir besoin d'aller voir le code.
- Les codes pour les modules et les scripts demandés (***Run_XXX.py***). Inclure des sections de test et de validation dans les modules. Commenter *raisonnablement* le code. Si le script est interactif, ajouter un affichage à l'exécution les instructions (*print* quelles touches pour faire quoi)
- La qualité du rendu a un vrai impact sur la note.
- L'utilisation du *LateX* et du *git* est conseillée.
- Sur la page de titre, bien mettre votre nom, numéro d'étudiant et **une photo d'identité.**

---

1. Infrastructure de simulation mécanique

Créer des classes élémentaires avec les méthodes externes suivantes. Toutes les grandeurs sont traités en MKS. *help(class)* devrait donner une documentation intelligible.

Classe Particule

- __init_(*masse, pos0, vit0, name, color, fix*)
  # avec des attributs position et vitesse qui gardent l'historique (listes)
  # On peut avoir un attribut pour la somme des forces extérieur ou la prendre comme argument pour la méthode PFD
  attribut *fix*=True pour ne pas respecter la PFD.
- getPosition(), getSpeed()
  # retourne la position et vitesse actuelles
- simule(*step*)
  # pour un pas de temps *step*, résolution de PFD avec somme des forces → accélération → vitesse → position
- setForce(*V3D*)
- setSpeed(*V3D*), setPosition(*V3D*),
  # ignore la PFD et impose la vitesse / position (au choix : impose *fix=True* ou marche seulement si *fixe=True*)
- plot2D(*plot*), plot3D(*plot*), gameDraw(*fenetre*)
  Fonctions d'affichage de trajectoire *matplotlib* et de représentation dans une fenêtre *pyGame*
- Toutes les méthodes internes nécessaires (__str__, __repr__, PFD etc)

Classe Univers

- __init__(*name, step, (dimensions en pixel W,H), scale*)
  avec l'échelle définie comme [m]**scale* = pixels (=> 1 pixel = (1/*scale*) mètres)
  step est le pas de temps de simulation.
  ajouter des attributs nécessaires pour gérer la population, des générateurs, le temps et la fenêtre pyGame
- addAgent(**particule*)
  on peut ajouter une ou plusieurs particules (voir *args dans la doc python / fonctions)
- addSource(*gene*rateur*)
  **generateur* peut etre une source ou plusieurs (voir *args)
- simule()
  effectuer un pas de simulation pour toute la population
- simuleAll(*time*)
  Faire une simulation de durée *time* (plusieurs pas de simule())
- gameInit ()
  Démarrer une simulation temps réel avec affichage dans une fenêtre pyGame. Fixer la fréquence d'affichage à 60 *fps*.
- gameUpdate()
  Gérer les entrées clavier/souris et simuler pour un pas d'affichage (pas de temps d'affichage ≠ pas de temps simulation!!)
- plot2D, plot3D

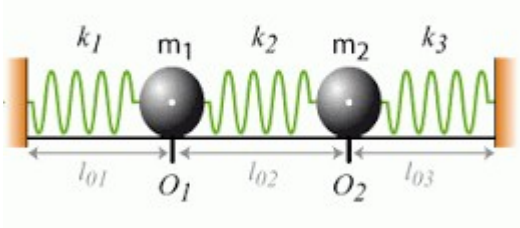
Des classes 'générateur' (pensez à ajouter un attribut 'active' qui permettrait une commande au clavier)

- ForceConst(*Value V3D, *particules*)
  **particule* peut être une particule ou plusieurs particules (voir **args*). Si vide →toutes les particules
- ForceHarmonic (*Value V3D, pulsation, *particules*)
  crée une force *Value**cos(*pulsation***t*)
- Viscosity(coef, *particules)
  amortissement visqueux
- ForceField(pos0,amplitude,*particules)
- SpringDumper(stiffness, dumping, lenght, particule1, particule2)
  Ressort-amortisseur entre p1 et p2
- Rod (particule1, particule2)
  tige rigide entre p1 et p2: cas particulier de ressort-amortisseur, avec k, c et l0 automatiques
- PrismJoint(axe V3D, particule)
  Laision prismatique: le mouvement de particule est contraint à l'*axe* qq soit les forces appliquées. (on peut ajouter un ressort-amortisseur perpendiculaire à l'*axe* ou dans le PFD utiliser uniquement la projection des forces sur l'*axe*)

2. Validation

Réaliser les simulations suivantes, les illustrer dans votre rapport avec des captures d'écran et des courbes commentées. Inclure un script py séparé pour chacun. Assurez-vous que les scripts se terminent proprement (*pygame.quit(), sys.exit()*). *Je ne déboguerai pas les scripts qui ne fonctionnent pas.*

1. Dans une scène de 10mx7m,10 particules de masse et de position (x,y) aléatoires, en chute libre selon -z, avec un champ de force attractive placé au centre, à z=-5m, avec des paramètres choisies pour pouvoir observer son effet sur les particules.
2. Dans une scène de 100mx70m, en 2D (x,y) avec g selon y et une viscosité : appui sur ‘espace’ crée une particule à position et vitesse aléatoires (dans des limites bien choisies).
3. Un système masse + (ressort+amortisseur) avec application au clavier d’une force constante ou une force harmonique. Vérifier l’accord avec le comportement théorique.
4. Trois pendules de l=10, 20 et 30 cm. Vérifier l’accord avec le comportement théorique. (bonus : https://www.youtube.com/watch?v=yVkdfJ9PkRQ)
5. Une pendule double, avec l= 15 et l2=10
6. Un système 2 ddl couplé 2 masses + 3 ressorts, avec m1 = m2 et k1=k3, l1 = l2 = l3 = 10 cm. Montrez qu’on retrouve bien les 2 modes propres en phase et opposition de phase aux bonnes fréquences théoriques. (bonus: pour ddl =3, 4 ou plus)



7. Reproduction du TP pendule inverse sur base mobile : créer en simulation un système équivalent, effectuer l’identification pour retrouver les paramètres choisis.
    1. Commande au clavier : l’utilisateur garde l’équilibre avec des touches gauche et droite, en déplaçant la base mobile.
    2. Commande automatique : proposer un contrôleur qui assure l’équilibre. Les touches gauche et droite perturbent l’équilibre par une petite force appliqué sur la masse en haut.

3. Conclusion et perspectives

Format libre.

4. Auto-Évaluation et commentaires

Attribuez-vous une note sur 5, qui prend en compte votre assiduité, votre participation, les efforts consacrés, travail personnel, et les progrès que vous avez fait. Justifier avec un texte de longueur raisonnable.