Simulateur de vol spatial

Un vaisseau spatial se déplaçant dans le système solaire est soumis à la force d'attraction des différentes planètes et autres corps célestes. Cette force peut modifier la vitesse du vaisseau et la trajectoire du vaisseau (l'accélérer, le ralentir, le faire dévier). En approchant un corps avec la bonne vitesse et la bonne direction, il est possible de se mettre en orbite autour de ce corps. En utilisant des propulseurs on peut également y atterrir.

Dans ce projet il vous est demandé d'implémenter un *simulateur interactif* qui permet de guider un vaisseau spatial à travers un système planétaire en actionnant ses propulseurs. Votre programme devra afficher l'avancement du vaisseau au cours du temps, ainsi que le mouvement des planètes et autres corps du système. Vous pouvez aussi voir ce projet comme un jeu où l'objectif est de partir depuis la Terre et être la première à se mettre en orbite autour de Mars, en utilisant au mieux ses propulseurs et son carburant en quantité limitée. Quel que soit votre point de vue, le déplacement du vaisseau doit respecter les lois de la mécanique.

Dans ce qui suit nous décrivons les différentes fonctionnalités que doit proposer votre programme, l'organisation du module projet et les critères d'évaluation.

1 Fonctionnalités demandées

1.1 Définition d'un système planétaire

Un système planétaire est constitué d'une étoile et de plusieurs planètes ou autres corps célestes (lunes, comètes, ...). Nous y ajouterons un vaisseau, celui que l'utilisatrice dirige. Tous les objets composant le système se déplacent sur le même plan (plan de l'écliptique), et seront donc affichés en deux dimensions. Ce plan est muni d'un repère orthonormé. Votre programme doit pouvoir **charger un système planétaire depuis sa description donnée dans un fichier texte** qui respecte le format décrit ci-dessous. Un exemple de tel fichier est présenté sur la Figure 1; d'autres exemples sont disponibles sur Moodle.

La première ligne commence par PARMS et définit les paramètres du système: *G* la valeur de la constante gravitationnelle, *dt* le pas de temps pour la simulation, *fa* le facteur d'accélération pour la simulation, et le *rayon* du système qui définit son étendue (donc ce qu'il faut afficher).

Les autres lignes définissent chacune un objet du système. Chaque ligne commence par le nom de l'objet (doit être unique) suivi de deux points (:). Ensuite on précise le type de l'objet parmi *Fixe*, *Simulé*, *Ellipse*, *Cercle* et *Vaisseau* qui détermine comment est définie la trajectoire de l'objet. Tous les objets ont une *masse* (qui peut être 0) ainsi que d'autres propriétés qui dépendent de leur type:

Fixe est un objet fixe dans le système de coordonnées. Ses propriétés sont *posx* et *posy* les coordonnées de sa position.

Simulé est un objet dont on va simuler la trajectoire par intégration numérique. Ses propriétés sont *posx* et *posx* les coordonnées de sa position initiale, et son vecteur de vitesse initiale composé de *vitx* et *vity*.

Ellipse est un objet qui suit une trajectoire elliptique autour de deux foyers fixes. Ses propriétés sont *f1* et *f2* deux noms d'objets fixes définis précédemment utilisés comme foyers de l'ellipse, *posx* et *posy* la position initiale de l'objet, et la *période* de rotation de l'objet.

Cercle est un objet qui décrit une trajectoire circulaire autour d'un centre qui peut être mobile. Ses propriétés sont *centre* le nom de l'objet au centre de la trajectoire, la position initiale *posx* et *posy*, et la *période*.

Vaisseau est un vaisseau dont la trajectoire est simulée par intégration numérique. Hormis les propriétés d'un objet *Simulé*, il a les deux propriétés *pprincipal* et *pretro* qui définissent la poussée (càd la force exercée sur le vaisseau) de son propulseur principal et de ces rétro fusées, respectivement. Un système peut contenir au plus un vaisseau.

D'autres propriétés optionnelles des objets seront données plus bas.

Les différentes propriétés sur une ligne sont séparées par un ou plusieurs blancs (whitespace). Les noms des objets stellaires sont des chaines de caractères qui ne contiennent pas d'espace ni le caractère deux points (:). Toutes les valeurs numériques sont de type double. Toutes les longueurs et distances (*rayon*, *posx*, *posy*, *vitx*, *vity*) sont exprimées en mètres. Les masses sont exprimées en kilogrammes. Le pas de temps *dt* est donné en secondes. Les périodes sont données en secondes. La valeur absolue indique le temps pour faire un tour complet, et le signe indique le sens de rotation: positif dans les sens des aiguilles d'une montre, négatif dans le sens inverse. Finalement, la constante gravitationnelle *G* est donnée en $m^3kg^{-1}s^{-1}$; la poussée des propulseurs en Newton. Ainsi, vous n'aurez pas de conversions d'unités de mesure à faire.

```
# Deux planètes tournent autour du soleil
# Un vaisseau se déplace dans le système
PARAMS G=0.01 dt=0.025 fa=1 rayon=500
Soleil: Fixe masse=30 posx=0 posy=0
Planète1: Simulé masse=1 posx=-100 posy=0 vitx=0 vity=0.06
Planéte2: Simulé masse=3 posx=0 posy=350 vitx=0.030 vity=0
X: Vaisseau masse=0.001 posx=75 posy=333 vitx=0 vity=0.017 pprincipal=0.0001 pretro=0.0000001
```

FIGURE 1 – Fichier décrivant un système planétaire.

1.2 Affichage et simulation

Votre programme doit afficher le système et sa dynamique au cours du temps [1], en respectant les lois de la mécanique. C'est une des fonctionnalités principales de ce projet.

Au lancement du programme, chacun des objets est affiché dans sa position initiale. Dès le lancement de la simulation, les objets commencent à se déplacer (à l'exception des objets fixes qui restent fixes). La vitesse et la trajectoire des corps simulés (types *Simulé* et *Vaisseau*) sont régies par les lois de la mécanique, c'est à dire influencées par la force de gravitation exercée par tous les autres corps du système. Les corps sur des trajectoires *Ellipse* ou *Cercle* suivent leur trajectoire à une vitesse définie par la période de rotation, et sans être influencés par la force de gravitation. Notez que les trajectoires *Ellipse* et *Cercle* sont demandés seulement pour le Livrable 2.

Des exigences plus détaillées sur l'affichage sont données plus tard. Vous trouverez sur Moodle une vidéo correspondant à la simulation du système de la Figure 1.

1.3 Guidage du vaisseau

Votre programme doit permettre à l'utilisateur de guider le vaisseau en activant son propulseur principal ou ses rétrofusées. C'est la deuxième fonctionnalité principale de ce projet.

Les propulseurs sont actifs lorsque l'utilisateur appuie sur les touches flèche du clavier: $\downarrow$ pour le propulseur principal, $\leftarrow$ et $\rightarrow$ pour les rétro fusées latérales, $\uparrow$ pour la rétrofusée. L'effet des propulseurs est d'exercer une force dans la direction opposée et provoquer un changement du vecteur vitesse du vaisseau (par ex. $\rightarrow$ fera tourner le vaisseau à gauche).

1.4 Interaction, interface, usabilité, aspect ludique

En implémentant les trois fonctionnalités précédentes (chargement du système, affichage et simulation, et guidage du vaisseau), vous aurez construit un simulateur minimal.

La prochaine étape consiste à **rendre votre programme facile et agréable à utiliser**, voire ludique. Voici une liste non exhaustive de fonctionnalités que vous pourriez ajouter. Certaines sont obligatoires (voir la description des livrables). N'hésitez pas à ajouter vos propres améliorations.

1.4.1 Affichage des corps

L'affichage du vaisseau est orienté (par ex. un triangle, ou une image) pour indiquer la direction de son vecteur vitesse. Les différents corps du système sont affichés de manière différente qu'on peut spécifier dans le fichier de définition du système à l'aide d'une des propriétés optionnelles *couleur*, *texture*, ou *sprite*. La valeur de *couleur* est une couleur en java, pour les deux autres la valeur est un nom de fichier qui contient la ressource image à utiliser. Par exemple
`couleur=BLUE`
`sprite=jupiter.jpg`.

Vous pouvez également faire varier le diamètre apparent des étoiles et planètes en fonction de leur masse.

Soyez créatives et proposez un affichage esthétique et informatif !

1.4.2 Informations détaillées

La vitesse et les coordonnées du vaisseau sont affichés en permanence dans un le *tableau de bord*. À la demande de l'utilisateur, le logiciel affiche diverses informations sur les corps composant le système telles que leur nom, masse, vecteur de vitesse.

Le taux de remplissage des réservoirs est également donné (voir 1.4.7) sur le tableau de bord. Ou mieux, on donne le temps de propulsion restant pour chacun de types de propulseurs.

1. Vous trouverez sur Moodle des précisions concernant la simulation au cours du temps.

1.4.3 Vue personnalisée, vues multiples

Pour permettre un guidage plus fin du vaisseau, l'utilisateur peut choisir la portion du système qu'il voit. Vous pouvez offrir des fonctions de zoom et dezoom, centrage sur l'étoile ou sur le vaisseau, rotation pour suivre la direction du vaisseau (vaisseau toujours orienté vers le haut), etc. Vous pouvez même afficher plusieurs vues, par ex. une vue globale du système à côté d'une vue rapprochée centrée sur le vaisseau.

1.4.4 Retour visuel, animations

Vous pouvez ajouter différentes informations visuelles utiles, par ex. une animation lorsque les propulseurs sont actifs, une indication lorsque le vaisseau subit très fortement l'attraction d'une planète, etc.

1.4.5 Collisions

Lorsqu'on simule la mécanique céleste, chaque corps est confondu avec son centre de gravité, et n'a donc pas de voluem. Cette simplification peut avoir des effets surprenants, par ex. un corps qui "rebondit" sur l'étoile au lieu de s'y écraser. Pour améliorer le réalisme vous pouvez implémenter la collision des corps. Pour cela il faut associer une taille à chacun des corps, par exemple calculée à partir de sa masse [2], et détecter lorsque deux corps rentrent en collision, ce qui résulterait en un seul corps dont la masse est la somme des deux.

1.4.6 *dt* adaptatif

Le problème du corps qui "rebondit" sur le soleil est dû à l'imprécision de la simulation numérique, et ne pourra pas toujours être résolu par la gestion des collisions. Pour rendre la simulation plus précise vous pouvez temporairement utiliser un *dt* plus petit, par exemple lorsque le vaisseau subit des forces de grande intensité.

1.4.7 Objectif à atteindre pour le vaisseau

On peut rendre le simulateur plus ludique en fixant un objectif au joueur, par ex. se mettre en orbite autour d'une planète, quitter le système solaire, etc. On peut également fixer un temps limite pour atteindre l'objectif et/ou une quantité de carburant limitée.

L'objectif doit faire partie de la définition du problème. Lorsque l'utilisateur pense avoir atteint l'objectif, il doit le signaler au programme suite à quoi les propulseurs sont désactivés. Le programme détermine alors si l'objectif est réellement atteint.

Voici les éléments de syntaxe et les critères de réussite pour quelques objectifs.

— Se mettre en orbite autour d'un objet du système. Cet objectif est atteint si le vaisseau effectue 3 tours autour de l'objet orbité se maintenant dans un intervalle de distance défini par *distmin* et *distmax*.
`OBJECTIF_ORBITE autourde=Mars distmin=20 distmax=40 tlimite=300 carbup=50 carbur=100`
Le paramètre *tlimite* (optionnel) est donné en secondes et indique le temps dont on dispose pour atteindre l'objectif. Les paramètres *carbup* et *carbur* (optionnels) indiquent la quantité de carburant disponible pour le propulseur principal et les rétrofusées, respectivement. Ils sont donnés en nombre de secondes pendant lesquelles on peut faire fonctionner un propulseur du type pour vider le réservoir.

— Quitter le système planétaire. Cet objectif est atteint lorsque le vaisseau se trouve au delà d'une certaine *distance* de l'étoile et sa vitesse est supérieure à la troisième vitesse cosmique. Les paramètres *tlimite*, *carbup* et *carbur* sont optionnels.
`OBJECTIF_VOYAGER distance=500 tlimite=300 carbup=50 carbur=100`

Si vous avez d'autres idées d'objectifs, postez une demande sur le forum du projet sur Moodle pour que les enseignants puissent fixer une syntaxe.

N'hésitez pas à publier sur Moodle vos défis (fichiers de définition de système) pour permettre aux autres étudiants de les tester sur leur propre implémentation !

1.4.8 Précalcul

Le guidage peut s'avérer très difficile: quelques secondes de propulseur en trop et votre vaisseau s'écrase sur une planète ou se retrouve expulsé du système planétaire. Pour pouvoir corriger sa trajectoire il faut savoir où on va. Ainsi votre programme peut pré-calculer et montrer la trajectoire du vaisseau définie par la vitesse actuelle, montrant où se trouvera le vaisseau dans disons 20 secondes si on ne modifie pas les conditions actuelles de propulsion. Le précalcul pourrait utiliser un *dt* plus grand dans quel cas le calcul serait pas précis mais pourrait être effectué plus rapidement. On

2. La taille d'un corps utilisée par la collision n'est pas forcément la taille affichée, pour des raisons d'échelle (vaisseau beaucoup plus petit que les autres corps).

peut même imaginer que le précalcul est fait en permanence et la trajectoire future précalculée est affichée en permanence devant le vaisseau.

1.4.9 Dessin des trajectoires

Faites laisser une trace aux objets lors de leur déplacement (i.e. dessiner leur propre trajectoire), vous obtiendrez quelques jolis dessins.

1.4.10 Cryo-sommeil

Les déplacements interstellaires sont très longs, c'est pourquoi les œuvres de science fiction ont inventé le cryo-sommeil. Pour éviter à l'utilisateur de longs temps d'attente, donnez la possibilité de faire un saut dans le temps. Pour votre programme cela consisterait à faire une simulation accélérée d'une certaine durée et montrer directement l'état du système juste après. Attention, simulation accélérée ne veut pas dire augmenter le pas de temps (risque de divergence numérique), mais plutôt quitter la simulation au cours du temps et faire le nombre de pas de simulation qui correspond à la durée demandée. Par exemple, si vous voulez vous plonger en cryo-sommeil pendant 1 minute, le pas de simulation est $dt = 0.025$ secondes et $fa = 1$, alors vous devez faire $60 \times \frac{1}{0.025} = 2400$ pas de simulation et afficher directement la configuration obtenue sans afficher les configurations intermédiaires.

2 Livrables

Vous devrez rendre deux versions de votre projet appelées *Livrables*. Pour chacune de ces versions il y a des fonctionnalités exigées et des fonctionnalités optionnelles.

2.1 Livrable 1 (dû pour le 3/11/2019)

Il vous est demandé un simulateur basique qui implémente toutes les fonctionnalités de base et quelques fonctionnalités supplémentaires, comme décrit ci-dessous.

Ces fonctionnalités de base sont exigées:

— **Charger** un système qui ne contient pas d'objets de type *Ellipse* ou *Cercle*. Le nom de fichier contenant le système est donné comme paramètre à la ligne de commande. Un message d'erreur doit être affiché en cas d'erreur de syntaxe dans le fichier (champ manquant, type de valeur incorrect pour un champ, PARAMS n'est pas la première ligne, plusieurs vaisseaux, etc.).

— **Simuler** le système en utilisant la méthode d'intégration Euler explicite (vue en TD de Modélisation);

— **Afficher** la simulation au cours du temps. L'affichage par défaut met les coordonnées $(0, 0)$ au centre. Les corps célestes ont une forme circulaire, le vaisseau a une forme triangulaire ou similaire permettant d'identifier sa direction. Vous devez également afficher en permanence un *tamleau de bord* contenant le temps absolu depuis le début de la simulation, la position et la vitesse du vaisseau.

— Permettre les interactions utilisateur pour **guider le vaisseau**.

De plus, vous devez implémenter au moins 2 parmi ces fonctionnalités supplémentaires (au choix):

— Affichage des corps (voir 1.4.1)

— Informations détaillées (voir 1.4.2)

— Animations (voir 1.4.4)

— Collisions (voir 1.4.5)

— Dessin des trajectoires (voir 1.4.9)

2.2 Livrable 2 (dû pour le 12/01/2020)

La nouvelle version doit enrichir votre logiciel ainsi:

— Supporter les objets de type *Ellipse* ou *Cercle*. Ceci inclut la gestion des erreurs de format de fichier pour ces types d'objets.

— Utiliser une méthode d'intégration plus précise (vue en TD de Modélisation).

— Permettre l'affichage de plusieurs vues, dont au moins une vue globale centrée en coordonnées $(0, 0)$ et une vue rapprochée centrée sur le vaisseau.

— Mise en place des collisions (bonus pour M3202C).

— Supporter au moins un des deux types d'objectifs (bonus pour M3202C).

— Le code du programme doit suivre le patron de conception Modèle-Vue-Contrôleur (MVC) pour implémenter le simulateur (M), l'affichage (V) et l'interaction avec l'utilisateur (C) (obligatoire pour M3105).

De plus, vous devez implémenter au moins 2 parmi ces fonctionnalités supplémentaires (au choix):
— Vue personnalisée (voir 1.4.3)
— *dt* adaptatif (voir 1.4.6)
— Objectifs (voir 1.4.7)
— Précalcul de la trajectoire (voir 1.4.8)
— Cryo-sommeil (voir 1.4.10)

3 Critères de notation

La réalisation du projet contribue à la note de trois modules. Nous expliquons ci-dessous les critères qui seront retenus pour chacune des notes. **Il est de votre responsabilité de montrer très régulièrement votre avancement à l'enseignant·e en projet pour savoir dans quel mesure votre travail respecte les attentes.**

3.1 M3302 Projet tuteuré

Les objectifs et compétences visées de ce module sont:
— produire un rendu qui respecte les contraintes
— s'organiser pour travailler en groupe et respecter les délais

Ainsi la notation se basera principalement le résultat, prenant en compte quelques indicateurs sur votre organisation. Concrètement:
— on note sur 3 points le fait que tous les membres de l'équipe ont travaillé, et que le travail a progressé de manière régulière (d'après les contributions **significatives** sur git);
— on note sur 1 point le respect des contraintes de rendu (seront définies sur Moodle);
— les fonctionnalités du programme sont notées sur 16 points, dont:
  — 8 points (Livrable 1) / 7 points (Livrable 2) pour les fonctionnalités obligatoires,
  — le reste pour les fonctionnalités supplémentaires. Les points seront donnés à l'appréciation des enseignants sur la qualité, la quantité et la complexité des fonctionnalités supplémentaires.

3.2 M3202C Modélisation mathématique

La note prendra en compte les fonctionnalités qui mettent en œuvre ce que vous aurez appris en Modélisation mathématique, en particulier
— calcul correct des trajectoires simulées
— calcul correct des trajectoires définies par des courbes
— vue correcte dans l'espace 2D (mise à l'échelle de la vue, calcul correct des coordonnées, affichage correct du vecteur vitesse)
— mise en place des collisions
— détection d'objectif atteint

3.3 M3105 COA

L'évaluation sera basée sur votre utilisation des concepts de POO avancée vus en cours:
— patrons de conception (hormis: MVC, Iterateur, Singleton, Obervateur)
— composition
— définition d'interfaces (java) adequates
— utilisation appropriée de la visibilité et de l'encapsulation
— double-dispatch

Le livrable s'accompagenra d'un petit rapport qui pour chacun de ces éléments, donnera des exemples d'utilisation et justifiera de leur bonne utilisation selon vous (pourquoi avoir utilisé cette solution au lieu d'une autre à ce moment).