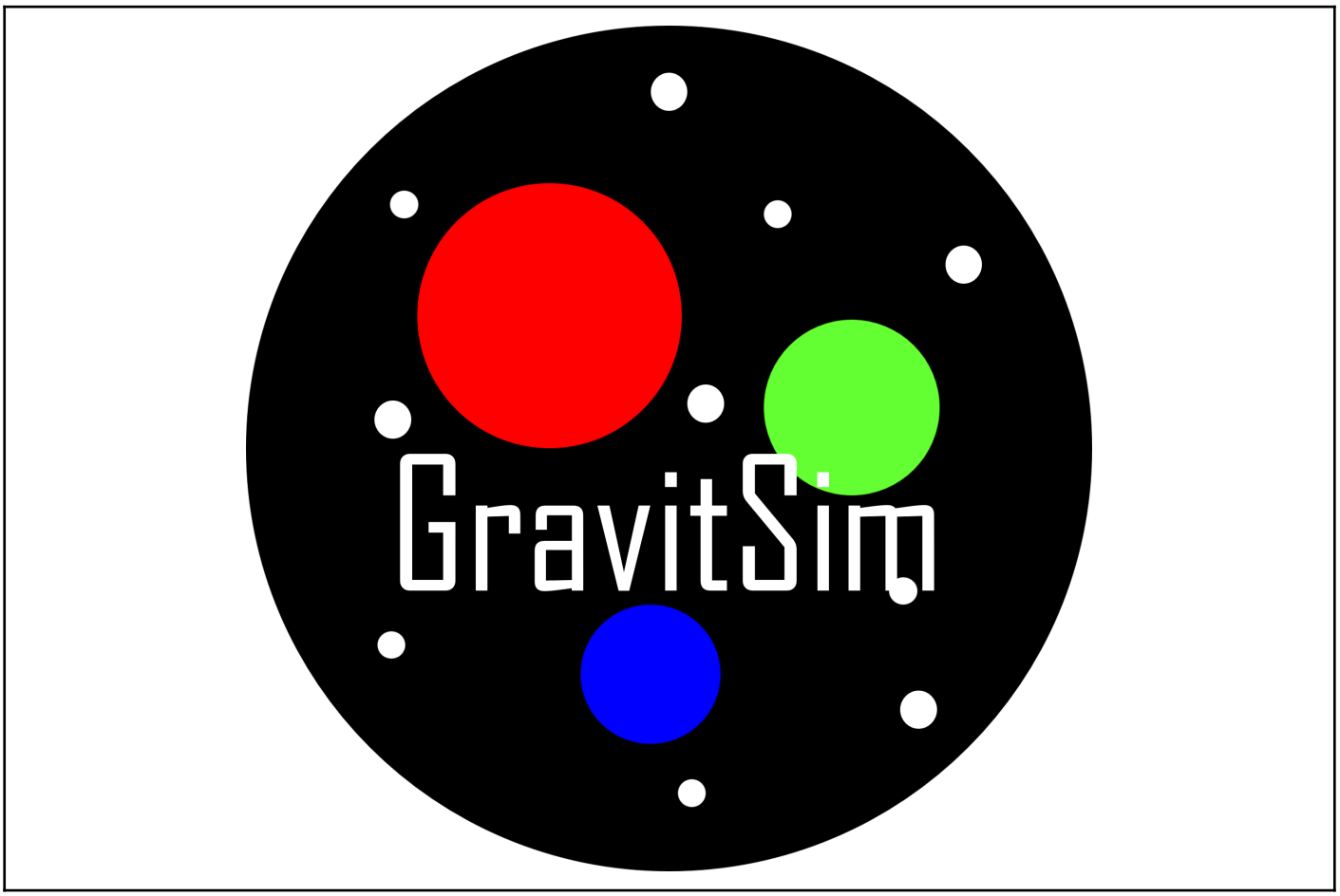
LES TROPHÉES NSI

ÉDITION 2024

DOSSIER DE CANDIDATURE

PRÉSENTATION DU PROJET



NOM DU PROJET :	GRAVITSIM
MEMBRE DE L'ÉQUIPE :	ARTHUR - YOSHIDA
MEMBRE DE L'ÉQUIPE :	GASPARD - SCIARRETTA
MEMBRE DE L'ÉQUIPE :	MANUELA - TEISSERE
NIVEAU D'ÉTUDE :	PREMIÈRE
ÉTABLISSEMENT SCOLAIRE :	LYCÉE PASTEUR SÃO PAULO
ENSEIGNANT DE NSI :	JOHANN DOLIVET

> PRÉSENTATION GÉNÉRALE :

Un des grands intérêts de la computation dans le domaine de la science est la modélisation de situations concrètes par l'ordinateur. Ces simulations peuvent permettre d'illustrer le comportement des phénomènes soit trop complexes, trop chers voire même physiquement impossibles pour améliorer la connaissance sur le sujet, vulgariser un thème, ou pour faire preuve de concept.

Un type de simulation souvent faite en ordinateur est une simulation gravitationnelle, celle-ci qui est du type d'une simulation N-Body (Une simulation qui modélise les intéractions entre un nombre N de corps) qui, dans le cas des simulations gravitationnelles, avec une formule assez simple: $F = G \times \frac{M_1 \times M_2}{d^2}$ permet l'illustration de phénomènes qui peuvent sembler à la fois réguliers, organiques et même chaotiques. Cette simplicité apparente qui explose en complexité est quelque chose qui fascine, et c'est exactement cela que notre groupe a choisi de développer pendant la période du second trimestre.

L'idée initiale était de faire une simulation gravitationnelle (sans le graphisme) qui est faite directement sur python, une interface graphique pour pouvoir facilement percevoir la position des corps qui est faite avec le module pygame (Ici, on utilise un moteur de jeu pour une simulation physique, ce qui peut paraître contre-intuitif; Cependant pour ce type de simulation les moteurs de jeu sont très bien adaptées et sont faciles à manipuler) et une interface tkinter afin de modifier l'état initial des corps, ainsi que pouvoir gérer la simulation et changer ses paramètres.

La modularité était aussi une des principaux fondements du projet, avec les expectatives de facilement modifier la position, vitesse, masse et couleurs des corps, ainsi que les constantes (comme la constante gravitationnelle), et le paramétrage de la simulation.

Il y a même mis en place des manières de facilement modifier la règle N-Body, qui, avec aurait permit même de modéliser des interactions électromagnétiques (Le support pour les interactions électromagnétiques n'est pas encore mis en place) voire même faire une simulation rudimentaire des effets de la relativité.

Et pour pouvoir facilement modifier les paramètres initiaux, il y a aussi un système qui permet de charger des "presets" contenant des systèmes planétaires et gravitationnels (Inclu dans le fichier "Presets" du projet).

Pour le développement de ces presets, et pour une édition plus facile à l'utilisateur, un système d'affichage de trajectoires a aussi été créé, qui prévoit les trajectoires et collisions prises par les corps.

> ORGANISATION DU TRAVAIL :

Distribution des tâches:

- L'interface tkinter pour paramétrer la simulation, ainsi que le logo a été fait par Manuela G. Il s'agit d'un endroit bien travaillé dans les cours où Manuela a beaucoup de connaissances.
- L'interface graphique pygame, ainsi qu'une partie de la simulation gravitationnelle ainsi que les projections de trajectoires associées ont été faites par Gaspard S. C'est un travail varié qui nécessite des compétences dans plusieurs sujets.
- Le système des presets, ainsi qu'une grande partie de l'infrastructure des classes comme les fonctions par Arthur Y.

Temps pris dans le projet:

Lorsqu'on ne sait pas combien de temps on a passé dans le projet, c'est bien au-delà des 50 heures avec la quasi-totalité fait hors de l'établissement scolaire en autonomie. Ça été un projet

développé pendant un peu moins d'un an, commençant au deuxième trimestre comme un projet pour l'école, jusqu'à ce qu'il devienne beaucoup plus quand on nous a annoncé qu'il concourrait pour les Trophées NSI.

Outils utilisées:

Pour coder, on a utilisé VSCode, pour la communication et organisation on a utilisé discord, pour le contrôle de versions on a utilisé github.

> LES ÉTAPES DU PROJET :

I. Simulation gravitationnelle

En algorithmique, la formule physique de la gravitation ne peut pas être directement appliquée dans l'ordinateur, car celle-ci ne comprend pas d'unité de temps. Donc il est nécessaire de faire une simulation par étape (dans un modèle qui mathématiquement ressemble aux suites) qui comprend un multiplicateur général qui est l'équivalent de l'unité de temps.

De plus, une des grandes contraintes lors d'une simulation N-Body est l'équilibre entre la précision et la performance: une précision plus haute est plus chère computationnellement, et un programme plus performant est à la base d'une simulation moins précise, et vice-versa. C'est ainsi qu'il faut trouver un équilibre entre ces deux, et ça était un grand enjeu lors de la simulation gravitationnelle.

Alors en ayant décidé une simulation par étape (et aussi à cause de la raison que ça serait déterministe), il est facile de créer une simulation par étape en utilisant la structure de modularité, et en utilisant les *classes* de python (qui est comment les sprites (lutins) et les corps sont sauvés en mémoire), il est facile de faire une manière versatile de calculer les forces gravitationnelles pour tous les corps. Ceux-ci sont contenus dans la liste *Bodies* dans *Jeu.py* et à chaque étape de la simulation (les étapes sont gérées dans l'interface pygame et l'interface tkinter):

- Les forces sont calculées grâce à la formule gravitationnelle.
- L'accélération est calculé pour chaque corps avec la relation $\Delta vitesse = \frac{Force}{Masse}$
- La vitesse du corps (appelé *momentum* dans le programme) et l 'accélération sont sommées sous forme vectorielle d'après le théorème de Thalès (Qui évite des conversions en angles puis vecteurs et sommes inutiles) par la relation $v = v + \Delta v$.
- Puis quand tous les forces sont calculées et appliquées sur la vitesse, la position est actualisé par la formule $P_{n+1} + P_n + v$ sur tous les corps.

Après, ça a été un travail significatif pour trouver des paramètres qui conviennent pour les simulations plus générales.

Outre de cela, il a été mis en place un système rudimentaire de collisions, qui vérifie si l'égalité $rayon_a + rayon_b <$ distanceab est correcte, puis consolide les deux corps en un. Ce système a été mis en chaîne après le calcul des positions.

II. Interface tkinter

L'interface tkinter créer pour cette simulation est utilisé pour régler les paramètres de la simulation qui est en train d'être exécutée. Ces deux fenêtres: *Édition* et *Simulation* contiennent des

champ d'entrées et des boutons qui permettent à l'utilisateur de personnaliser sa simulation d'après une base de presets. De plus, après avoir exploité le logiciel, l'utilisateur peut également sauver ses presets, afin de les pouvoir réutiliser. Parmi les personnalisations possibles on observe celle du momentum, celle de la position de la planète et celle de la vitesse du programme.

III. Interface pygame

Le but principal de l'interface pygame est de facilement et intuitivement illustrer la position et la trajectoire des corps. Pour cela, on modélise les corps comme des cercles parfaits, avec un rayon de formule $\frac{4}{3}\pi R^3 \times 1500$. Ensuite, car pygame n'est pas fait pour glisser la caméra, il existe un système de caméra qui permet la "translation" de la caméra et aussi le "zoom" de celle-ci, qui en réalité change les "sprites" (planètes) de position et de taille en relation à la position et la taille de la caméra. La position et "zoom" de la caméra est gouvernée par les relations $posRelative = posSprite - posCamera$ et par $échelle_{sprite} = \frac{Dimensions_{caméra}}{Dimensions_{monde}}$.

De plus, les trajectoires sont calculées en série avec les règles puis une ligne (élément pygame) est dessinée entre les positions successives des trajectoires puis colorées en fonction de la couleur du corps. Un système de visualisation des collisions a été mis en place qui projette les collisions qui se passent. Une des choses qu'on a fini par ajouter dans le développement après des tests d'utilisation du logiciel sont les références de trajectoires: celles ci prennent un corps comme référence, et soustraient des trajectoires de tous les autres corps la vitesse du corps de référence, avec la relation pour tous les points successifs de la trajectoire

$$positionCorps_{n+1} = positionCorps_n + vitesseCorps_n - vitesseRef_n$$

IV. Modularité

GravitSim a comme un des buts principaux la modularité. Celle-ci est achevée par un système de sauvegarde et de chargement de presets qui est faite grâce à l'interface tkinter. Ensuite, afin de sauvegarder le preset (qui revient à sauver une liste d'objets dans un document) on utilise le module pickle propre à python pour les serializer et les mettre dans un document. Ensuite, afin de charger le document, le module pickle avec l'interface tkinter est à nouveau utilisé, puis toutes les interfaces sont actualisées

> FONCTIONNEMENT ET OPÉRATIONNALITÉ :

Pour chaque partie, les systèmes de la partie ont été testés individuellement.
L'interface pygame a été testée au fur et à mesure de son progrès: en créant des systèmes directement en code (sans l'interface graphique), en faisant attention pour aborder tous les cas normaux d'une simulation (Contre-exemple: un corps dans un corps, un corps de masse nulle, etc.).
Ensuite, il y a eu des tests très approximatifs pour tester des choses comme la performance (le résultat de ce test varie fortement entre la configuration de l'ordinateur et la quantité de corps, donc un résultat exact est difficile). Dû à l'infinité de possibilités de configuration possible, des algorithmes de tests qui prennent en compte toutes les possibilités n'ont pas étés fait.

La dernière étape de notre projet a été de faire la jonction de l'interface et le reste du programme. C'était une période chaotique, avec des problèmes de tous les types, et inattendus. Une durée considérable de notre temps a été dédiée à cela. Au moment de l'union beaucoup de boutons ne réalisaient pas leurs fonctions. Une communication constante entre nous a pris place, afin que l'interface remplisse tous les critères et personnalisations établies. Avec une communication efficace

entre le groupe, les algorithmes de l'interface tkinter et pygame ont pu communiquer et concevoir un logiciel qui marche sans problèmes majeurs.

Puis on nous a informés qu'on allait concourir dans les trophées NSI, donc on a repris le projet pour finir quelques des derniers points qu'on a pas suffisamment eu de temps pour finir. C'est comme les trajectoires qui n'étaient pas toujours évidentes quand il y avait une multitude de corps, où on a mis notre visualisation de collisions et de références de trajectoires. On a pris soin de laisser quelques éléments plus intuitifs, comme les intéractions avec les souris, les coordonnées affichées, etc. Et bien évidemment, on a fait une multitude de petits changements et débogages pour avoir le logiciel plus fluide. C'est aussi quand on a pris le temps de bien revoir les presets, et d'en introduire quelques nouveaux (comme celui des trois corps).

Dû à la complexité du projet, l'unique manière de s'assurer qu'il n'y aurait pas de bugs est de le tester le plus possible, dans tous les cas qu'on puisse penser; et de trouver les points de faillites les plus communs (Comme les cas où on a deux corps dans un seule point).

En général, le projet final est assez bien conforme aux attentes. L'idée qui était projetée dans la tête des élèves au début du trimestre, s'est bien concrétisée en fin de cette période. Malgré quelques ambitions plus vastes, on se trouve tous satisfaits de notre travail.

> OUVERTURE :

Idées d'améliorations:

- Des formules et mathématiques plus précisément choisies, et plus correctement appliquées.
- Une échelle réelle des corps, des forces, du temps, etc.'
- Une précision paramétrable, pour pouvoir faire des simulations qui nécessitent plus de précision (comme par exemple une simulation des points lagrangians)
- Une interface graphique plus intuitive.
- La disposition de plus d'informations (statistiques, positions en temps réel, etc).
- L'optimisation de la loi de gravitation et du système de collisions (Multithreading, utilisation de la GPU, chunking, algorithmes plus efficaces, etc.)
- Mettre en place plus de règles, comme une simulation électromagnétique, de la relativité, etc.

Notre opinion sur le projet:

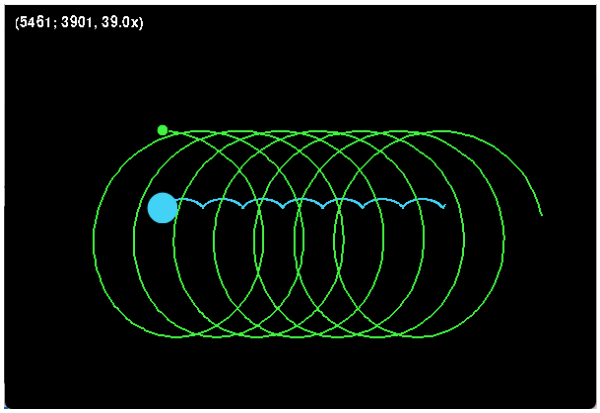
Notre projet reste un projet d'école, pour mieux qu'on veuille créer le projet parfait, on n'a pas les capacités pour le faire. Cela dit, on a bien appris en faisant le projet, des nouvelles compétences et des nouvelles erreurs ont été réalisées. On aurait dû sans doute refonder le projet sur des bases plus fortes, comme par exemple définissant directement les calculs entre coordonnées et vecteurs afin de pas avoir de manuellement calculer toutes les sommes de vecteurs. L'interface tkinter, malgré ce qui est déjà avec python et ce qu'on appris dans les cours n'a pas été le meilleur choix pour faire l'interface, en effet ça nous a beaucoup limitées. Bref, il y a beaucoup à améliorer, à refaire.

En effet, notre projet nous a permis de développer des nouvelles compétences, bien sur en programmation, mais aussi en physique et en design de logiciels. On a aussi pu explorer et comprendre un des phénomènes qui gouvernent l'univers, trouvant encore plus de situations, qui comme déjà mentionné, fascine.

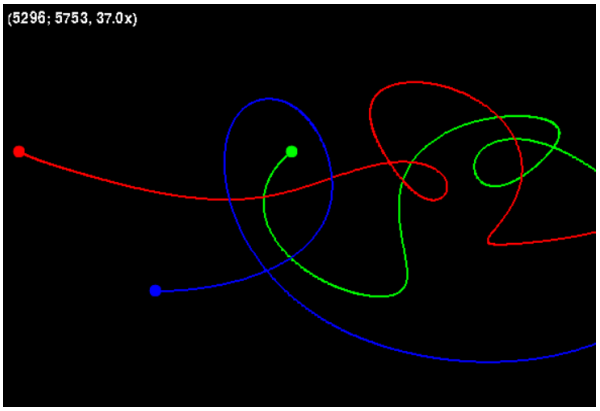
> ANNEXE: Quelques phénomènes observés:

Cette partie rassemble quelques phénomènes qu'on a pu observer grâce à notre logiciel (qui malgré tout, était son objectif), des phénomènes réels, et quelques cas spécifiques.

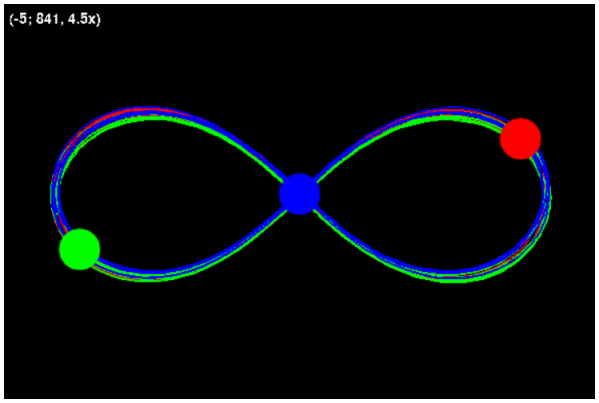
Orbites: Un des phénomènes astrophysiques les plus simples sont les orbites, où la trajectoire parabolique d'un corps plus petit est suffisamment rapide pour faire une révolution complète d'un corps plus grand. Voici notre preset *OrbiteSimple:*



Chaos: Cependant, quand on ajoute un troisième corps, les trajectoires sont plus prévisibles: on parle d'un ensemble chaotique, c'est-à-dire, un ensemble où les corps ne vont jamais revenir à leur position initiale. Voici notre preset *chaos_2:*



Solution 3 corps: Pourtant, il existe quelques solutions particulières du problème des trois corps. L'une d'elles trouvées par Alain Chenciner et Richard Montgomery dans un papier où figure un système de trois corps de masses égales qui semblent faire un chiffre 8 dans leur trajectoire, d'autant plus fascinant. Voici notre preset *Solution3Corps:*



Système solaire: Pour se divertir, on a bien évidemment recréé le système solaire sans souci d'échelle. Voici notre preset *SystemeSolaire:*