Partie II :

Construction du programme

Nous nous lançâmes, une fois la nature du projet établie, dans une étape de conception. Point de codage précipité là, mais plutôt une concertation quant à la forme qu'aura le produit fini : les fonctions à utiliser, leurs interactions, leur répartition entre différents fichiers pour améliorer la lisibilité du tout, mais aussi une concertation quant aux problèmes de fond : comment traiter le système, quel intégrateur utiliser, comment animer le système.

Il apparut assez rapidement qu'il serait éminemment plus aisé pour nous de travailler en deux dimensions. Premièrement car la conservation du moment cinétique dans un système tel que le système solaire, notre exemple de référence, a pour conséquence que tous les corps se situent dans un même plan, et deuxièmement car les méthodes d'animation en trois dimensions que nous avions pu trouver se montraient substantiellement plus obscures que leurs homologues planes.

Parallèlement, l'intégrateur que nous allions employer était toujours sujet à controverse. Nous avions déniché sur la toile nombre de méthodes, toutes plus lourdes, complexes ou saugrenues les unes que les autres. D'un autre côté, il y avait *odeint*, le mystérieux intégrateur de *Scipy*. La doc du module nous indiqua sans aucune autre forme de préambule qu'il se base sur une bibliothèque FORTRAN. Devant notre circonspection et notre indécision, notre chargé de TD nous proposa d'utiliser *odeint* sans crainte, arguant qu'il ferait parfaitement l'affaire. L'avenir confirma ses dires.

Concernant la forme du programme, il serait découpé en « quatre plus un » fichiers principaux, afin d'améliorer la lisibilité, *via* des séparations thématiques : les relations entre les

corps, le système dans son ensemble, les fonctions relatives à l'animation et le main, véritable chef d'orchestre coordonnant le tout avec les données entrées par l'utilisateur grâce à un dernier fichier de configuration. Étant donné que nous nous sommes tenus à notre « cartographie » initiale du code, nous invitons tout lecteur éclairé à se référer à la disposition des fonctions dans notre code, ainsi qu'à leur description.

Pour résumer toutefois, notons que le module *corps* contient les fonctions chargées de déterminer différents paramètres liant un corps donné et l'un de ses congénères, comme la distance entre eux ou la force d'interaction entre eux, ainsi que des données propres à un corps donné, telle son énergie cinétique. *systeme* déborde quant à lui de fonctions gérant tout l'univers en évolution, ce qui comprend le calcul des équations du mouvement, la détermination de la position de tous les éléments du système mais surtout de quoi faire avancer le système dans le temps, à l'aide de tous les éléments précités et d'*odeint*. La partie suivante abordera plus en détails les raisons profondes de l'apparition de plusieurs méthodes d'itération du système. *animation*, comme son nom le suggère, s'occupe de rendre toutes les données calculées appréciables pour les yeux de l'utilisateur, à grands renforts de données paramétrables et d'affichage avec *FuncAnimation* de *matplolib*. Enfin, *main* gère les options, les paramètres de l'utilisateur, réunit tous les autres modules et fait presque le café ; c'est, comme le veut l'éthique de la programmation, la colonne vertébrale, le nexus du programme – ou toute autre métaphore du genre.

C'est donc en suivant notre plan que nous avançâmes au fil des séances et des vacances, pas à pas, Bruno mettant à profit son expérience dans l'informatique pour ce premier contact avec *Python* et Denis apprenant tout de l'animation.