NICOLAS CORBARD, Terminale S

***Présentation du projet :***

Notre projet consiste à représenter graphiquement en deux dimensions les différentes orbites planétaires de notre système solaire, ainsi que le mouvement des planètes sur ces mêmes orbites.

Ce projet nous permet d’appliquer des concepts de programmation Python à des connaissances scientifiques que l’on étudie en Mathématiques mais aussi en Physique. Ainsi, ce projet nous permet de revoir nos connaissances tout en en explorant de nouvelles dans un autre domaine.

Pour ma part, ce projet avait une valeur assez particulière en raison de ma passion pour l’astrophysique qui me conduira probablement à des simulations de mouvements à l’aide d’outils informatiques. Cela représentait donc mes premiers pas dans une certaine approche de ma passion.

Ainsi, ce projet ne présente pas d’amélioration par rapport aux programmes déjà existants, car ce domaine (la simulation de mouvements spatiaux) est déjà très développé. En revanche, il nous permit d’appréhender les méthodes de résolution des problèmes rencontrés par les personnes ayant réalisé des programmes de simulation.

***Analyse du besoin et recherche d’idées :***

Au début de notre réflexion sur notre projet, nous avions pour objectif de créer un programme capable de présenter une vulgarisation scientifique des mouvements qui animent notre système solaire. Nous voulions également réaliser un programme permettant à l’utilisateur de créer une planète, mais par manque de temps, nous avons du abandonner cette idée. Nous nous sommes fixés différents objectifs à respecter :

* Fidélité raisonnable à la réalité (échelles, dimensions, temps…)
* Coder en Python
* Vue en deux dimensions
* Code compréhensible pour quelqu’un ne connaissant pas la programmation

Pour réaliser ce projet, nous avons donc utilisé la programmation Python, avec notamment les librairies graphique et mathématique. En effet, notre programme fait appel aux librairies Tkinter, permettant la création de fenêtre d’affichage, ainsi que la mise en mouvement d’objets associés à des planètes, mais aussi à la librairie Math, permettant d’importer les fonctions mathématiques telles que les puissances. Enfin, nous avons mobilisé la librairie Time qui permet, comme son nom l’indique, d’apporter au programme la notion d’écoulement du temps. De plus, pour trouver les informations nécessaires à la mise en œuvre de ce projet demandant des données précises par rapport à la notion d’échelle, nous nous sommes référés à diverses sources d’internet (sites spécialisés ou non).

***Structure globale du projet :***

L’ensemble du projet est constitué d’un seul fichier python et de plusieurs fichiers images (Background, astres avec les planètes et le Soleil).

CREATION DES FENETRES D’AFFICHAGE ET DES REPRESENTATION DES ASTRES

(Import des images)

IMPORT DES LIBRAIRIES

(Tkinter, Math, Time)

***Répartition des tâches et démarche collaborative :***

LANCEMENT DE LA BOUCLE D’ANIMATION DES PLANETES

(Mobilisation des fonctions de mouvement/ de calcul)

INTERACTIONS AVEC L’UTILISATEUR

(Entrées du nombre d’images par seconde et de la vitesse de déplacement des planètes)

INITIALISATION DE LA POSITION DES ASTRES ET DE LEURS CARACTERISTIQUES

(Angles de départ pour chaque planète, propriétés de chaque astre, coordonnées de chaque astre, tracés des trajectoires de chaque planète)

|  |  |  |
| --- | --- | --- |
| Tâches | Planning | Noms |
| Rédaction d'un cahier des charges simplifié | 13 janvier | Nicolas |
| - Recherche de l'équation de la trajectoire d'un mouvement elliptique  - Affichage d'un cercle à l'aide de la librairie Tkinter  - Questionnement sur la représentation à l'échelle ou non des orbites  - Utilisation de l'équation d'un cercle | 20 janvier | * Nicolas * Vincent * Julien * Nicolas/ Vincent |
| - Avancée dans l'interface graphique de notre projet de système solaire, avec une vue de haut et en 2D. On continue les recherches ! | 27 janvier | Vincent/ Julien/ Nicolas |
| - Affichage d'orbites de différentes planètes, mais pas du déplacement de la planète car nous n'avons pas encore trouvé la formule permettant de donner l'équation horaire des planètes. | 02 février | Nicolas |
| - Implémentation graphique du soleil, calcul des dimensions de son image, calcul des orbites des planètes affichables sur un écran. différents tests de mise à l'échelle.  - Explication claire dans le code afin de montrer l'importance de chacune des lignes de code. Différents tests pour l'affichage du nom des astres et provoquer un changement de couleur lors du passage du curseur sur le nom de l'astre | 03 février | * Nicolas * Julien |
| -mise en mouvement circulaire uniforme d’un cercle codé dans tkinter | 03 Mars | Vincent |
| - Application de la méthode trouvée le 3 Mars sur le code déjà réalisé où l'on voyait toutes les planètes immobiles. Désormais, les planètes sont en mouvement mais il reste encore à adapter les vitesses de déplacements. | 23 Mars | Vincent/ Nicolas/ Julien |
| - Calcul des nouvelles coordonnées d'une planète après une seconde, en utilisant les lois trigonométriques (cos x; sin x)  - Recherche et tests des principes d’animation et de déplacements des planètes dans le temps (nombre d’images par seconde, temps de pause entre deux déplacements) | 14 Avril | Nicolas |
| - Application des principes mathématiques trouvés à la séance précédente, tout en ajoutant la possibilité de faire varier le temps de pause entre chaque animation, ce qui fait varier la vitesse des planètes. | 20 Avril | Julien/ Vincent |
| - Ajout de la possibilité pour l'utilisateur de choisir la vitesse des planètes à travers une saisie. | 25 Avril | Vincent |
| - Ajout des images des planètes à la place des simples cercles qui représentaient jusqu'alors les planètes (suppression des diamètres des cercles présents dans le code)  - Ajout des textes sur la fenêtre des planètes gazeuses | 28 Avril | * Vincent * Julien |
| - Rédaction de la partie commune du dossier | 5 Mai | Nicolas |

Lors de la réalisation de ce projet, notre principal moyen de transmission des informations a été le système de messagerie Messenger, permettant des échanges rapides et simples d’informations à travers des pièces jointes, et la création d’un groupe commun. Cependant, nous avons également du nous voir très régulièrement hors des cours pour « brainstormer » et échanger nos idées. Enfin, on peut également citer la plateforme Github sur laquelle nous notions nos avancées personnelles et collectives.

***Réalisation personnelle :***

Lorsque nous avons commencé le projet, la première tâche dont je me suis chargé a été la recherche du moyen scientifique, mathématiques d’aborder le problème de la mise en mouvement des planètes, qui a été associée à une mise en mouvement circulaire et non pas elliptique, car, en effet, une représentation elliptique aurait entrainé des calculs bien trop complexes (problèmes liés à l’excentricité de l’orbite terrestre).

Or, pendant notre année de cours, nous avons vu en Physique les lois Newtoniennes, avec notamment l’étude des mouvements des objets. En revanche, nous n’avions pas vu le système d’équation horaire permettant de donner la position d’un astre à chaque instant, ce qui correspondait à une étude d’un mouvement circulaire uniforme. Pendant la première partie de la réalisation, je me suis donc consacré à la recherche d’une équation horaire d’un mouvement circulaire uniforme, afin de la coder par la suite en Python. Malheureusement, cette équation nécessitait des connaissances trop avancées pour être utilisées et comprises en Terminale, avec notamment l’utilisation de coordonnées polaires et non pas cartésiennes.

J’ai donc cherché un autre moyen de représenter une évolution de la position d’un astre selon une trajectoire circulaire dans un repère cartésien représenté par la fenêtre de tkinter. J’ai fini par avoir l’idée d’utiliser la trigonométrie avec les relations mobilisant sinus et cosinus. J’ai commencé à envisager les distances entre le Soleil et les astres comme étant un côté du triangle. Avec cette méthode, le calcul des nouvelles coordonnées de la planète après un certain temps était rendu facile, puisque l’on connaissait l’angle selon lequel la planète devait se déplacer en une certaine durée.

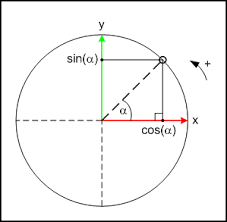


Schéma de trigonométrie

Il a fallu également trouver le moyen de représenter la dynamique de déplacement des planètes en maîtrisant l’échelle du temps d’affichage par rapport au temps de révolution des planètes. Pour cela, nous avons décidé tout d’abord de prendre comme référence le temps souhaité par l’utilisateur (en secondes) pour une révolution de la Terre autour du Soleil. En m’inspirant des principes d’animation des jeux vidéo, j’ai compris qu’en définissant la fréquence d’affichage en images par seconde, nous pouvions calculer les vitesses de déplacement de chacune des planètes en angles ainsi que le temps de pause nécessaire entre deux déplacements. Ainsi, j’ai réalisé des tests en utilisant la librairie python Time et ses fonctions « sleep() » et « time() ».

J’ai également réalisé d’autres tâches dont une, correspondant à la création de la fenêtre d’affichage, mais j’ai aussi réalisé tous les calculs de distances par rapport aux problèmes d’échelles, ce qui m’a ensuite permis de calculer et d’afficher les trajectoires de chaque planète du système solaire. J’ai également répondu aux problèmes de représentation des distances en divisant le système solaire en deux fenêtres où deux échelles de distances différentes s’appliquent.

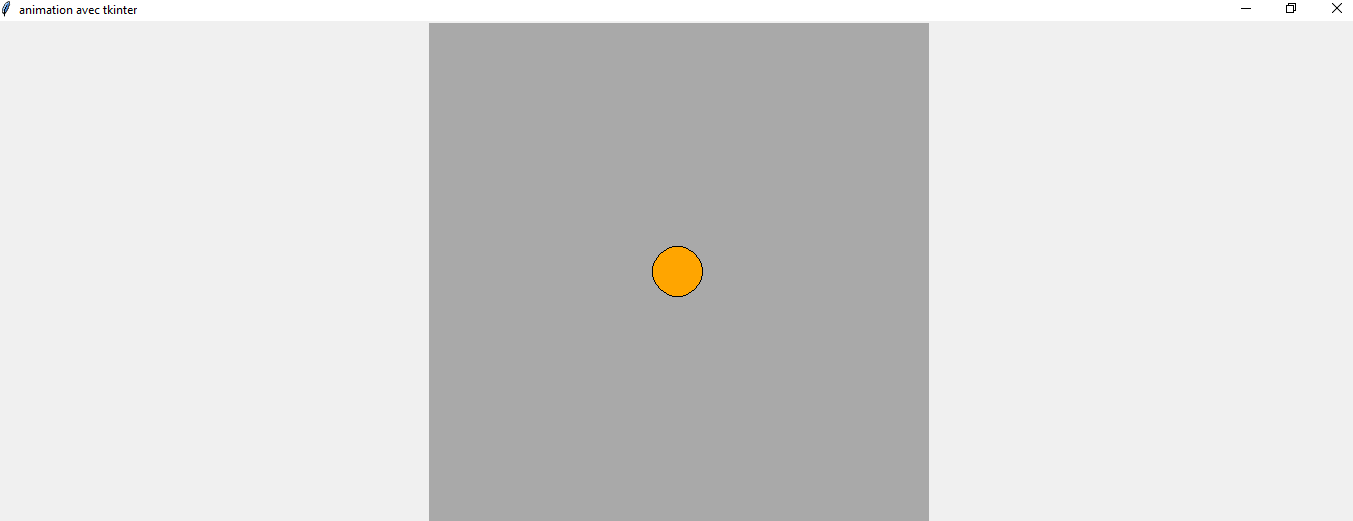
Je me suis également chargé de toute la partie visuelle des planètes puisque je me suis chargé de la sélection, de l’amélioration, et de la réduction, et du codage des images utilisées dans le programme.



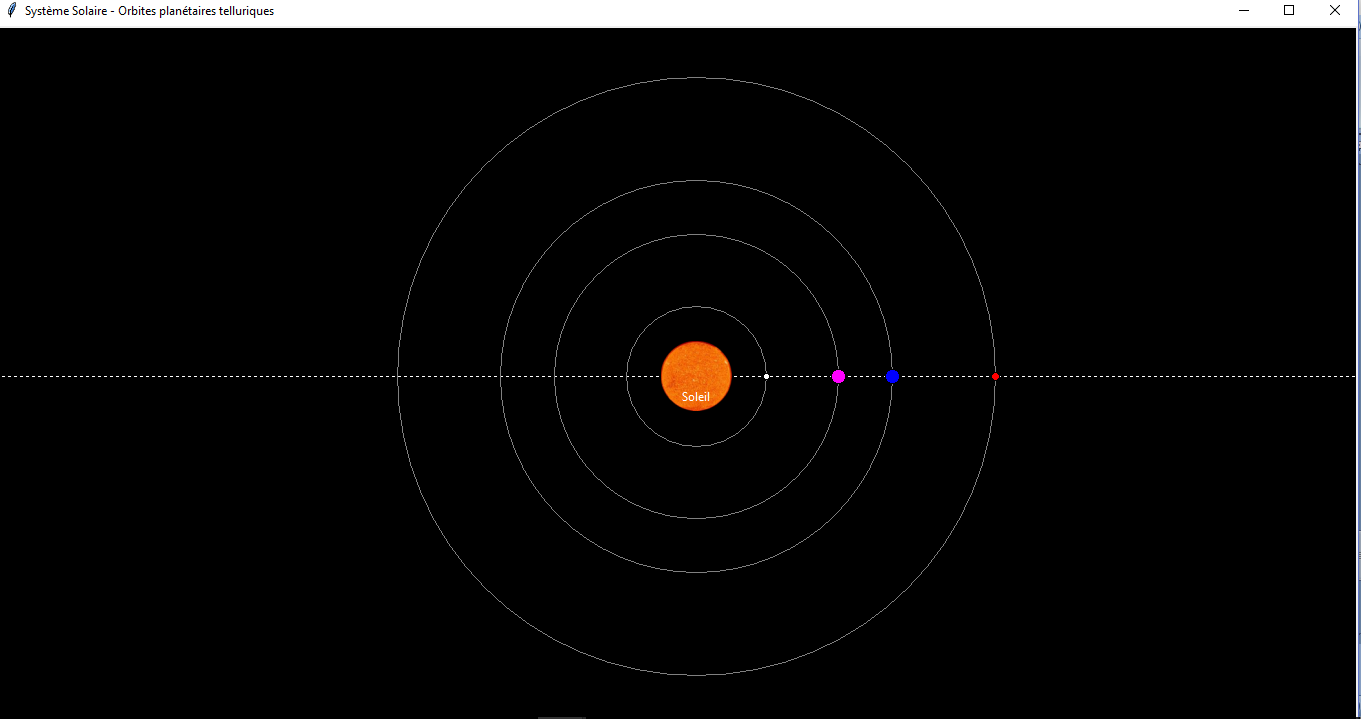
Sur cette image, on peut voir l’une des deux frames (frame des planètes telluriques)

Avec cette capture d’écran, on se rend compte que la taille des planètes est arbitraire, car impossible à reproduire avec une échelle, mais que les distances Soleil-Astres sont, quant à elles, parfaitement à l’échelle. Les planètes, qui n’étaient à l’origine que des cercles en mouvement, sont représentées par des images leur correspondant. On peut également voir les différentes trajectoires affichées en gris, ainsi que les planètes restant sur le tracé de ces trajectoires grâce à la méthode « cosinus/ sinus ».

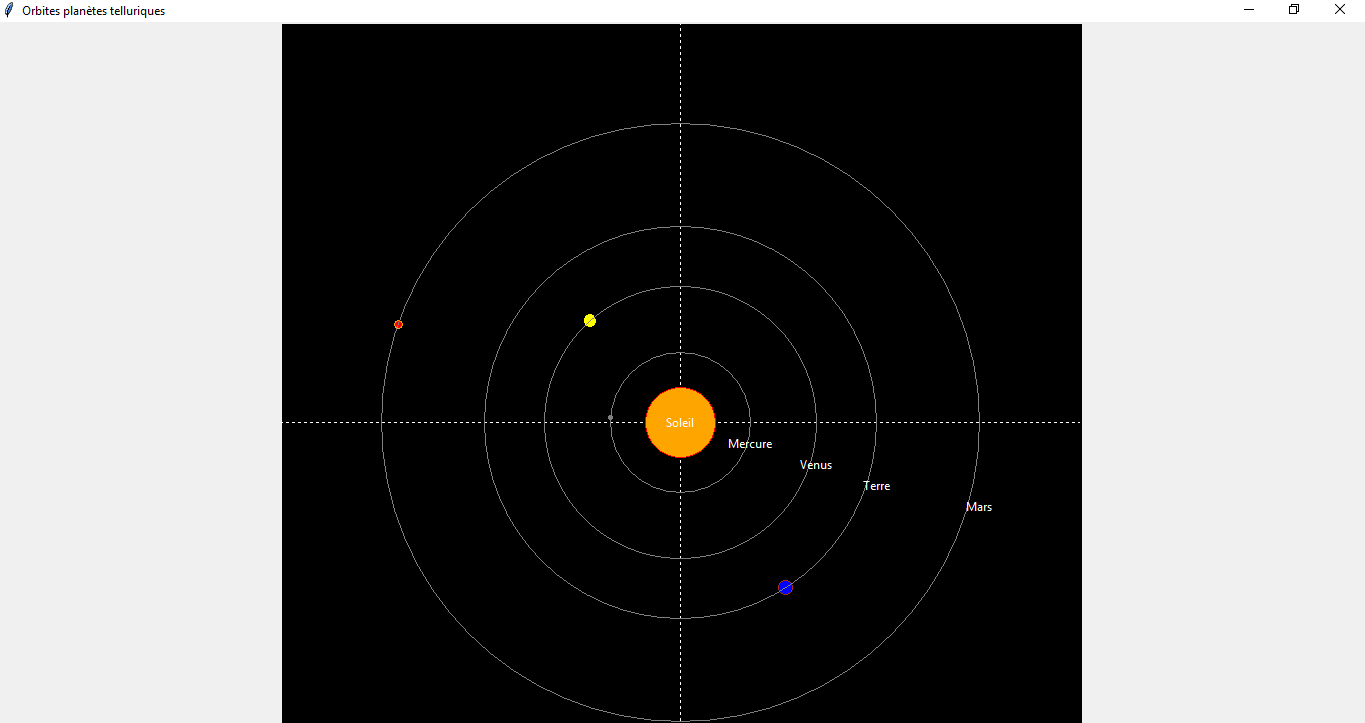
***Présentation des tests :***



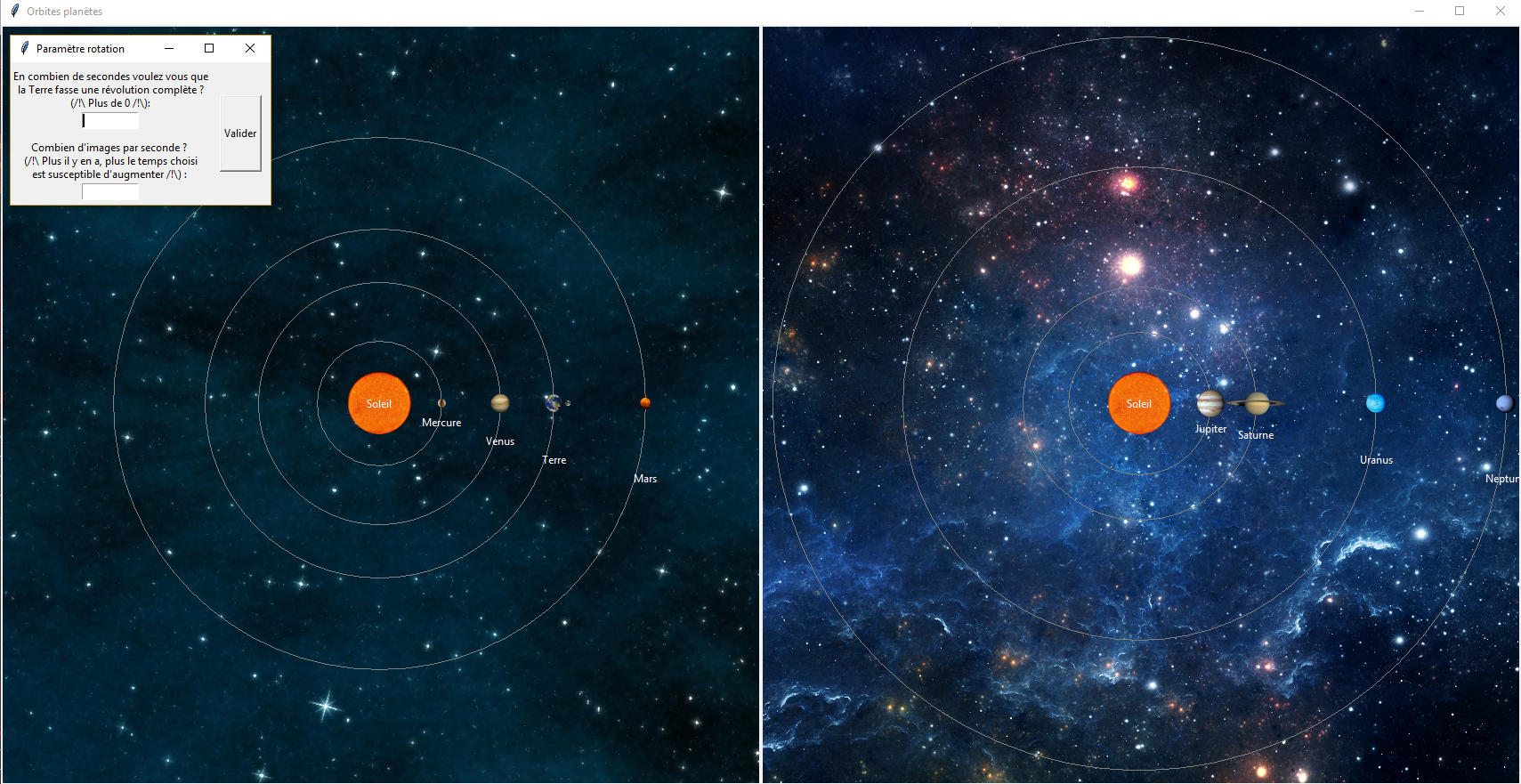
Sur cette image, on peut voir mon premier test d’affichage d’un cercle grâce à Tkinter.



Ici, on peut voir la version immobile du système solaire que j’ai réalisé, où nous testions encore les tracés de trajectoire et les échelles de distances. De plus, les planètes ne sont pas encore associées à des images, mais bien à des cercles de couleur.



Sur ce test, on peut voir que les planètes suivent bel et bien leurs orbites après avoir introduit les fonctions de mouvement mobilisant cosinus et sinus.



Enfin, cette dernière capture d’écran montre l’aspect final du projet, avec l’interaction entre l’utilisateur et le programme, la totalité des images, ainsi que le mouvement des planètes autour du Soleil. On peut également noter l’ajout de dernière minute correspondant au mouvement de la Lune autour de la Terre.

***Améliorations possibles :***

Ce programme restant tout de même assez simpliste par rapport aux phénomènes se déroulant réellement, de nombreuses améliorations restent possibles.

En effet, on peut tout d’abord penser au fait que les orbites planétaires ne sont pas exactement circulaires, puisqu’elles sont en réalité elliptiques. Les ellipses étant beaucoup plus complexes à interpréter mathématiquement en raison de leur excentricité par rapport à un cercle, nous avions pris la décision de coder des mouvements circulaires. Cela entraine également le problème des vitesses, car dans un mouvement circulaire, la vitesse est constante, alors que dans un mouvement elliptique, elle ralentie ou accélère selon la position de l’astre sur l’ellipse.

De plus, bien que les distances entre les astres soit à l’échelle, la taille des astres a été, quant à elle, réglée de manière arbitraire, car cette échelle ne pouvait pas être transcrite sur une petite fenêtre. Sans compter le fait qu’une fenêtre unique pour toutes les planètes serait préférable, mais là encore, les distances étaient trop grandes.

D’un point de vue plus esthétique, une amélioration pourrait consister en la vision 3D du mouvement en ajoutant un axe de profondeur. Nous avons également la possibilité d’un affichage des données de chaque planète qui apparaitrait après un simple « clic » sur la planète. Enfin, il aurait été possible d’ajouter une interaction supplémentaire avec l’utilisateur à travers la saisie de caractéristiques menant à la création et à la simulation du mouvement d’une planète imaginaire.

***Bilan :*** Ce projet m’a donc permis d’aborder des connaissances plus avancées par rapport à celle que nous abordions en cours dans un domaine me passionnant, à savoir la physique, ou plus précisément l’astrophysique. J’ai pu ainsi apprendre la représentation polaire d’un point dans un repère et ses utilisations au-delà du théorique. J’ai donc pu me motiver d’autant plus quant à mon envie de découvrir les moyens de simulation du mouvement des corps célestes, et plus généralement des phénomènes s’opérant dans le cosmos.

Lien du projet : <https://github.com/VincentFaivre/SolarSystem/projects/1>

1. ################
2. # MODULES #
3. ################
4. from tkinter import \* #Module pour afficher tout le programme
5. from math import \* #Module qui importe des fonctions mathématiques (notamment le cosinus et le sinus)
6. import time #Module pour gérer la notion du temps dans le programme




12. ###########################
13. # OSSATURE DU PROGRAMME #
14. ###########################

17. # FENËTRE MERE #
19. fen = Tk()
20. fen.title('Orbites planètes')
21. screenLongueur = 850
22. screenHauteur = 850
23. midScreenLongueur = screenLongueur/2
24. midScreenHauteur= screenHauteur/2






32. # IMAGES #
34. #Fond
35. fondTelluriqueImage = PhotoImage(file='F:/SystemeSolaire/Images/EspaceBleu.png')
36. fondGazeuseImage = PhotoImage(file='F:/SystemeSolaire/Images/EspaceMagnifique.png')
37. #Etoile
38. soleil\_TelluriqueImage = PhotoImage(file='F:/SystemeSolaire/Images/Soleil\_Tellurique.png')
39. soleil\_GazeuseImage = PhotoImage(file='F:/SystemeSolaire/Images/Soleil\_Tellurique.png')
40. #Planetes
41. terreImage = PhotoImage(file='F:/SystemeSolaire/Images/Terre.png')
42. mercureImage = PhotoImage(file='F:/SystemeSolaire/Images/Mercure.png')
43. venusImage = PhotoImage(file='F:/SystemeSolaire/Images/Venus.png')
44. marsImage = PhotoImage(file='F:/SystemeSolaire/Images/Mars.png')
45. jupiterImage = PhotoImage(file='F:/SystemeSolaire/Images/Jupiter.png')
46. saturneImage = PhotoImage(file='F:/SystemeSolaire/Images/Saturne.png')
47. uranusImage = PhotoImage(file='F:/SystemeSolaire/Images/Uranus.png')
48. neptuneImage = PhotoImage(file='F:/SystemeSolaire/Images/Neptune.png')
49. #Sattelite
50. luneImage = PhotoImage(file='F:/SystemeSolaire/Images/Lune.png')
51. #Autres
52. rotationSolaireImage = PhotoImage(file='F:/SystemeSolaire/Images/RotationSolaire.png')



57. # FRAME TELLURIQUES #
59. cadreTelluriques = Frame(fen)
60. cadreTelluriques.pack(side = LEFT)
62. can = Canvas(cadreTelluriques, width = screenLongueur, height = screenHauteur) # Le Canevas des Telluriques
63. can.pack()
64. fondTellurique = can.create\_image(midScreenLongueur, midScreenLongueur, image=fondTelluriqueImage)
66. #axeV = can.create\_line(midScreenLongueur, 0, midScreenLongueur, screenHauteur, fill="white", dash=(4, 4))
67. #axeH = can.create\_line(0, midScreenHauteur, screenLongueur, midScreenHauteur, fill="white", dash=(4, 4))


71. # FRAME GAZEUSES #
73. cadreGazeuses = Frame(fen)
74. cadreGazeuses.pack(side = RIGHT)
76. canG = Canvas(cadreGazeuses, width = screenLongueur, height = screenHauteur)
77. canG.pack()
78. fondGazeuse = canG.create\_image(midScreenLongueur, midScreenLongueur, image=fondGazeuseImage)
80. #axeV = canG.create\_line(midScreenLongueur, 0, midScreenLongueur, screenHauteur, fill="white", dash=(4, 4))
81. #axeH = canG.create\_line(0, midScreenHauteur, screenLongueur, midScreenHauteur, fill="white", dash=(4, 4))




87. ###################
88. # LES ASTRES #
89. ###################

92. '''
93. A chaque fois, on remplace 'Astre' par le nom de celui-ci
95. diamAstre = [Diamètre de l'Astre] (pixels)
96. rayonAstre = diamAstre/2 [Rayon de l'Astre] (pixels)
97. distAstre\_Soleil = [Distance de l'Astre par rapport au Soleil] (pixels)
98. revolAstre = [Période de révolution de l'Astre autour du Soleil en jours] (jours)
99. rapportRevolAstre\_Terre = [Rapport entre la période de révolution de la Terre et celle de l'Astre] (jours)
101. COORDONNEES
102. xAstre = midXScreen = xMSoleil [Positionnement X du centre de l'Astre]
103. yAstre = midYScreen = yMSoleil [Positionnement Y du centre de l'Astre]
105. xGAstre et yGAstre ont beau être pratique dans Tkinter, on ne va pas les utiliser et uniquement prendre comme référence les points xM et yM
106. qu'on appellera d'ailleurs x et y tout simplement

109. trajetAstre = can.create\_oval(xSoleil-distAstre\_Soleil, ySoleil-distAstre\_Soleil, xSoleil+distAstre\_Soleil, ySoleil+distAstre\_Soleil,
110. ### On crée d'abord le trajet avant l'astre parce que l'astre va donc se placer par dessus le trait du trajet car il est appelé après ###
112. astre = can.create\_oval(xAstre-rayonAstre, yAstre-rayonAstre, xAstre+rayonAstre, yAstre+rayonAstre)
113. ou
114. astre = can.create\_image(xAstre, yAstre, image='F:/SystemeSolaire/Images/Astre.png')
115. '''
117. # Angles de départ de chaque astres #
119. tMe = 0 #Mercure
120. tV = 0 #Venus
121. tT = 0 #Terre
122. tMa = 0 #Mars
123. tJ = 0 #Jupiter
124. tS = 0 #Saturne
125. tU = 0 #Uranus
126. tN = 0 #Neptune
127. tL = 0 #Lune



132. ###########
133. # ETOILE #
134. ###########

137. # ----- Soleil (Canevas des Telluriques) ----- #
138. # Caractéristiques
139. diamSoleil = 70 #en pixels
140. rayonSoleil = diamSoleil/2
141. xSoleil, ySoleil = midScreenLongueur, midScreenHauteur
143. # Création de l'objet ou astre 'soleil'
144. soleil\_Tellurique = can.create\_image(xSoleil, ySoleil, image=soleil\_TelluriqueImage)
146. # Texte 'Soleil'
147. xTextSoleil = xSoleil
148. yTextSoleil = ySoleil
149. textSoleil = can.create\_text(xTextSoleil, yTextSoleil, text='Soleil', fill='white', activefill='red')







158. # ----- Soleil (Canevas des Gazeuses) ----- #
159. diamSoleil = 40
160. rayonSoleil = diamSoleil/2
161. xSoleil, ySoleil = midScreenLongueur, midScreenHauteur
163. # Création de l'astre 'soleil'
164. soleil\_Gazeuse = canG.create\_image(xSoleil, ySoleil, image=soleil\_GazeuseImage)

167. # Texte 'Soleil'
168. xTextSoleil = xSoleil
169. yTextSoleil = ySoleil
170. textSoleil = canG.create\_text(xTextSoleil, yTextSoleil, text='Soleil', fill='white', activefill='red')
171. ###########################
172. # PLANETES TELLURIQUES #
173. ###########################
174. # ------------------ #
175. # Terre #
176. # ------------------ #
177. #Propriétés
178. diamTerre = 14
179. rayonTerre = diamTerre/2
180. distTerre\_Soleil = 196
181. revolutionTerre = 365.256363051 #jours soit 1 an
182. rapportRevolTerre\_Terre = 1
183. #Coordonnées
184. xTerre = xSoleil + distTerre\_Soleil
185. yTerre = ySoleil
186. # Trajectoire de la Terre
187. trajetTerre = can.create\_oval(xSoleil-distTerre\_Soleil , ySoleil-distTerre\_Soleil, xSoleil+distTerre\_Soleil, ySoleil+distTerre\_Soleil, outline = "grey")
188. # Création de l'astre 'terre'
189. terre = can.create\_image(xTerre, yTerre, image=terreImage)
190. # Texte 'Terre'
191. xTexteTerre = xTerre
192. yTexteTerre = yTerre + rayonTerre\*9
193. texteTerre = can.create\_text(xTexteTerre, yTexteTerre, text='Terre', fill='white', activefill='#71945A') # color = earth green
194. # ------------------------#
195. # Mercure #
196. # ------------------------#
197. #Propriétés
198. diamMercure = 6
199. rayonMercure = diamMercure/2
200. distMercure\_Soleil = 70
201. revolutionMercure = 87.969256 #jours
202. rapportRevolMercure\_Terre = revolutionTerre/revolutionMercure
203. #Coordonnées
204. xMercure = xSoleil + distMercure\_Soleil
205. yMercure = ySoleil
206. # Trajectoire de Mercure
207. trajetMercure = can.create\_oval(xSoleil-distMercure\_Soleil, ySoleil-distMercure\_Soleil, xSoleil+distMercure\_Soleil, ySoleil+distMercure\_Soleil, outline='gray')
208. # Création de l'astre 'mercure'
209. mercure = can.create\_image(xMercure, yMercure, image=mercureImage)
210. # Texte 'Mercure'
211. xTexteMercure = xMercure
212. yTexteMercure = yMercure + rayonTerre\*3
213. texteMercure = can.create\_text(xTexteMercure, yTexteMercure, text='Mercure', fill='white', activefill='brown')
214. # -------------------- #
215. # Venus #
216. # -------------------- #
217. #Propriétés
218. diamVenus = 13.6
219. rayonVenus = diamVenus/2
220. distVenus\_Soleil = 136.2
221. revolutionVenus = 224.6997056 #jours
222. rapportRevolVenus\_Terre = revolutionTerre/revolutionVenus
223. #Coorodonnées
224. xVenus = xSoleil + distVenus\_Soleil
225. yVenus = ySoleil
226. # Trajectoire de Vénus
227. trajetVenus = can.create\_oval(xSoleil-distVenus\_Soleil, ySoleil-distVenus\_Soleil, xSoleil+distVenus\_Soleil, ySoleil+distVenus\_Soleil, outline='gray')
228. # Création de l'astre 'venus'
229. venus = can.create\_image(xVenus, yVenus, image=venusImage)
230. # Texte 'Venus'
231. xTexteVenus = xVenus
232. yTexteVenus = yVenus + rayonTerre\*6
233. texteVenus = can.create\_text(xTexteVenus, yTexteVenus, text='Venus', fill='white', activefill='orange')
234. # -------------------- #
235. # Mars #
236. # -------------------- #
237. #Propriétés
238. diamMars = 8
239. rayonMars = diamMars/2
240. distMars\_Soleil = 299
241. revolutionMars = 686.979852
242. rapportRevolMars\_Terre = revolutionTerre/revolutionMars
243. #Coordonnées
244. xMars, yMars = xSoleil+distMars\_Soleil, ySoleil
245. # Trajectoire de Mars
246. trajetoire = can.create\_oval(xSoleil-distMars\_Soleil, ySoleil-distMars\_Soleil, xSoleil+distMars\_Soleil, ySoleil+distMars\_Soleil, outline='gray')
247. # Création de l'astre 'mars'
248. mars = can.create\_image(xMars, yMars, image=marsImage)
249. # Texte 'Mars'
250. xTexteMars = xMars
251. yTexteMars = yMars + rayonTerre\*12
252. texteMars = can.create\_text(xTexteMars, yTexteMars, text='Mars', fill='white', activefill='red')
253. #----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------#
254. # ---------------------- #
255. # Jupiter #
256. # ---------------------- #
257. #Propriétés
258. diamJupiter = 20
259. rayonJupiter = diamJupiter/2
260. distJupiter\_Soleil = 80
261. revolutionJupiter = 4332.589 #jours
262. rapportRevolJupiter\_Terre = revolutionTerre/revolutionJupiter
263. #Coordonnées
264. xJupiter = xSoleil+distJupiter\_Soleil
265. yJupiter = ySoleil
266. # Trajectoire de Jupiter
267. trajetJupiter = canG.create\_oval(xSoleil-distJupiter\_Soleil, ySoleil-distJupiter\_Soleil, xSoleil+distJupiter\_Soleil, ySoleil+distJupiter\_Soleil, outline='gray')
268. # Création de l'astre 'jupiter'
269. jupiter = canG.create\_image(xJupiter, yJupiter, image=jupiterImage)
270. #Texte 'Jupiter'
271. xTexteJupiter = xJupiter
272. yTexteJupiter = yJupiter + rayonTerre\*4
273. texteJupiter = canG.create\_text(xTexteJupiter, yTexteJupiter, text='Jupiter', fill='white', activefill='#F4A460') #colour = light brown
274. # ---------------------- #
275. # Saturne #
276. # ---------------------- #
277. #Propriétés
278. diamSaturne = 18
279. rayonSaturne = diamSaturne/2
280. distSaturne\_Soleil = 131
281. revolutionSaturne = 10759.23
282. rapportRevolSaturne\_Terre = revolutionTerre/revolutionSaturne
283. #Coordonnées
284. xSaturne = xSoleil+distSaturne\_Soleil
285. ySaturne = ySoleil
286. # Trajectoire de Saturne
287. trajetSaturne = canG.create\_oval(xSoleil-distSaturne\_Soleil, ySoleil-distSaturne\_Soleil, xSoleil+distSaturne\_Soleil, ySoleil+distSaturne\_Soleil, outline='gray')
288. # Création de l'astre 'saturne'
289. saturne = canG.create\_image(xSaturne, ySaturne, image=saturneImage)
290. #Texte 'Saturne'
291. xTexteSaturne = xSaturne
292. yTexteSaturne = ySaturne + rayonTerre\*5
293. texteSaturne = canG.create\_text(xTexteSaturne, yTexteSaturne, text='Saturne', fill='white', activefill='#edc9af') #colour = sand
294. # ---------------------- #
295. # Uranus #
296. # ---------------------- #
297. #Propriétés
298. diamUranus = 14
299. rayonUranus = diamUranus/2
300. distUranus\_Soleil = 266
301. revolutionUranus = 30685.4
302. rapportRevolUranus\_Terre = revolutionTerre/revolutionUranus
303. #Coordonnées
304. xUranus = xSoleil+distUranus\_Soleil
305. yUranus = ySoleil
306. # Trajectoire de Uranus
307. trajetUranus = canG.create\_oval(xSoleil-distUranus\_Soleil, ySoleil-distUranus\_Soleil, xSoleil+distUranus\_Soleil, ySoleil+distUranus\_Soleil, outline='gray')
308. # Création de l'astre 'uranus'
309. uranus = canG.create\_image(xUranus, yUranus, image=uranusImage)
310. #Texte 'Uranus'
311. xTexteUranus = xUranus
312. yTexteUranus = yUranus + rayonTerre\*9
313. texteUranus = canG.create\_text(xTexteUranus, yTexteUranus, text='Uranus', fill='white', activefill='cyan')
314. # ----------------------- #
315. # Neptune #
316. # ----------------------- #
317. #Propriétés
318. diamNeptune = 14
319. rayonNeptune = diamNeptune/2
320. distNeptune\_Soleil = 412
321. revolutionNeptune = 60266
322. rapportRevolNeptune\_Terre = revolutionTerre/revolutionNeptune
323. #Coordonnées
324. xNeptune = xSoleil+distNeptune\_Soleil
325. yNeptune = ySoleil
326. # Trajectoire de Neptune
327. trajetNeptune = canG.create\_oval(xSoleil-distNeptune\_Soleil, ySoleil-distNeptune\_Soleil, xSoleil+distNeptune\_Soleil, ySoleil+distNeptune\_Soleil, outline='gray')
328. # Création de la planète 'neptune'
329. neptune = canG.create\_image(xNeptune, yNeptune, image=neptuneImage)
330. #Texte 'Neptune'
331. xTexteNeptune = xNeptune
332. yTexteNeptune = yNeptune + rayonTerre\*12
333. texteNeptune = canG.create\_text(xTexteNeptune, yTexteNeptune, text='Neptune', fill='white', activefill='#85addb') # color = blue
334. # --------------------- #
335. # Lune #
336. # --------------------- #
337. #Propriétés
338. diamLune = 6
339. rayonLune = diamLune/2
340. distLune\_Terre = 16 #valeur complètement gratuite
341. revolutionLune = 27.3220
342. rapportRevolLune\_Terre = revolutionTerre/revolutionLune
343. #Coordonnées
344. xLune = xTerre+distLune\_Terre
345. yLune = yTerre
346. # Création de l'astre 'lune'
347. lune = can.create\_image(xLune, yLune, image=luneImage)
348. #########################
349. # FENETRE UTILISATEUR #
350. #########################
351. # Gérer la vitesse de raffraichissement de la visualisation graphique
352. tempsRevolTerre = 0
353. nombreImagesParSeconde = 0
354. def recupererValeur():
355. global tempsRevolTerre, nombreImagesParSeconde
356. tempsRevolTerre = float(entreeTempsRevolTerre.get())
357. nombreImagesParSeconde = float(entreeNombreImagesParSeconde.get())
358. window = Tk() #Deuxieme fenetre pour gérer les paramètres du programme
359. window.title('Paramètre rotation')
360. frameW = Frame(window)
361. frameW.grid()
362. frame1 = Frame(frameW)
363. frame1.grid(row = 0, column=0, pady=5)
364. frame2 = Frame(frameW)
365. frame2.grid(row = 1, column=0, pady=5)
366. frame3 = Frame(frameW)
367. frame3.grid(row = 0, column=1, rowspan=2)
368. label1 = Label(frame1, text="En combien de secondes voulez vous que\nla Terre fasse une révolution complète ?\n(/!\ Plus de 0 /!\):").grid()
369. entreeTempsRevolTerre = Entry(frame1, width=10)
370. entreeTempsRevolTerre.grid()
371. entreeTempsRevolTerre.focus\_set()
372. label2 = Label(frame2, text="Combien d'images par seconde ? \n(/!\ Plus il y en a, plus le temps choisi\nest susceptible d'augmenter /!\) : ").grid()
373. entreeNombreImagesParSeconde = Entry(frame2, width=10)
374. entreeNombreImagesParSeconde.grid()
375. boutonValider = Button(frame3, text='Valider', relief=RAISED, height = 5, command=recupererValeur)
376. boutonValider.grid(padx = 10)
377. while tempsRevolTerre == 0 and nombreImagesParSeconde == 0: # Tant que l'utilisateur n'a pas rentré 2 valeurs
378. window.update()
379. #######################################################
380. # MOUVEMENT CIRCULAIRE POUR CHAQUE PLANETE #
381. #######################################################
382. def animationMercure(tMe):
383. global xMercure, yMercure
384. angle = radians(tMe)
385. xMercure = xSoleil + distMercure\_Soleil\*cos(angle)
386. yMercure = ySoleil - distMercure\_Soleil\*sin(angle)
387. can.coords(mercure, xMercure, yMercure)
388. def animationVenus(tV):
389. global xVenus, yVenus
390. angle = radians(tV)
391. xVenus = xSoleil + distVenus\_Soleil\*cos(angle)
392. yVenus = ySoleil - distVenus\_Soleil\*sin(angle)
393. can.coords(venus, xVenus, yVenus)
394. def animationTerre(tT):
395. global xTerre, yTerre
396. angle = radians(tT)
397. xTerre = xSoleil + distTerre\_Soleil\*cos(angle) #Mouvement en x de droite à gauche de la Terre
398. yTerre = ySoleil - distTerre\_Soleil\*sin(angle) #Mouvement en y de haut en bas de la Terre
399. can.coords(terre, xTerre, yTerre)
400. #print("A tT=", tT, ", on a xTerre =", xTerre, "et yTerre=", yTerre)
401. def animationMars(tMa):
402. global xMars, yMars
403. angle = radians(tMa)
404. xMars = xSoleil + distMars\_Soleil\*cos(angle)
405. yMars = ySoleil - distMars\_Soleil\*sin(angle)
406. can.coords(mars, xMars, yMars)
407. def animationJupiter(tJ):
408. global xJupiter, yJupiter
409. angle = radians(tJ)
410. xJupiter = xSoleil + distJupiter\_Soleil\*cos(angle)
411. yJupiter = ySoleil - distJupiter\_Soleil\*sin(angle)
412. canG.coords(jupiter, xJupiter, yJupiter)
413. def animationSaturne(tS):
414. global xSaturne, ySaturne
415. angle = radians(tS)
416. xSaturne = xSoleil + distSaturne\_Soleil\*cos(angle)
417. ySaturne = ySoleil - distSaturne\_Soleil\*sin(angle)
418. canG.coords(saturne, xSaturne, ySaturne)
419. def animationUranus(tU):
420. global xUranus, yUranus
421. angle = radians(tU)
422. xUranus = xSoleil + distUranus\_Soleil\*cos(angle)
423. yUranus = ySoleil - distUranus\_Soleil\*sin(angle)
424. canG.coords(uranus, xUranus, yUranus)
425. def animationNeptune(tN):
426. global xNeptune, yNeptune
427. angle = radians(tN)
428. xNeptune = xSoleil + distNeptune\_Soleil\*cos(angle)
429. yNeptune = ySoleil - distNeptune\_Soleil\*sin(angle)
430. canG.coords(neptune, xNeptune, yNeptune)
431. def animationLune(tL):
432. global xLune, yLune
433. angle = radians(tL)
434. xLune = xTerre + distLune\_Terre\*cos(angle) #Mouvement en x de droite à gauche de la Terre
435. yLune = yTerre - distLune\_Terre\*sin(angle) #Mouvement en y de haut en bas de la Terre
436. can.coords(lune, xLune, yLune)
437. ##########################################
438. # MISE EN MOUVEMENT DES PLANETES #
439. ##########################################
440. condition = True
441. while condition:
442. #MERCURE
443. animationMercure(tMe)
444. #VENUS
445. animationVenus(tV)
446. #TERRE
447. animationTerre(tT)
448. #MARS
449. animationMars(tMa)
450. #JUPITER
451. animationJupiter(tJ)
452. #SATURNE
453. animationSaturne(tS)
454. #URANUS
455. animationUranus(tU)
456. #NEPTUNE
457. animationNeptune(tN)
458. #LUNE
459. animationLune(tL)
460. fen.update() # Nécessaire pour voir le mouvement
461. pauseAnim = 1/nombreImagesParSeconde # Fréquence d'affichage définissant le temps de pause entre 2 déplacements des astres
462. deplacementTerreDegres = 360 /(tempsRevolTerre\*nombreImagesParSeconde) # Calcul de l'angle en degrés d'un déplacement de la Terre sur son orbite
463. tMe += deplacementTerreDegres\*rapportRevolMercure\_Terre
464. tV += deplacementTerreDegres\*rapportRevolVenus\_Terre
465. tT += deplacementTerreDegres
466. tMa += deplacementTerreDegres\*rapportRevolMars\_Terre
467. tJ += deplacementTerreDegres\*rapportRevolJupiter\_Terre
468. tS += deplacementTerreDegres\*rapportRevolSaturne\_Terre
469. tU += deplacementTerreDegres\*rapportRevolUranus\_Terre
470. tN += deplacementTerreDegres\*rapportRevolNeptune\_Terre
471. tL += deplacementTerreDegres\*rapportRevolLune\_Terre
472. time.sleep(pauseAnim)