Activité n°4 – Scripter avec Blender : La gravitation universelle

1 – Premier contact avec la gravitation

Ouvrir le fichier **act4--introblend.blend**.

- En activant la louche lecture de la fenêtre TIMELINE, vous allez pouvoir lancer une animation automatique et interactive.
- Vous pouvez déplacer les différents objets à l'aide d'un clic-droit pour les sélectionner puis les déplacer à l'aide des flèches.

02° Vers quel objet semble être attirés les autres objets ?

03° En plus de la distance, quel autre paramètre semble être très important pour savoir comment les planètes vont graviter (ou non) autour du Soleil ?

Pour l'instant, la simulation est loin d'être une véritable simulation : les masses et les distances ne sont pas respectées.

Nous allons passer sur une autre simulation où cette fois les choses seront mieux définies.

2 – La simulation de la rotation de la Terre et de la Lune

04° Ouvrir le fichier **blender-meca-4.blend**.

Ce fichier va vous permettre de simuler l'effet de la gravitation universelle sur la Terre, la Lune et le Soleil. Nous allons également voir une fonctionnalité importante de la programmation : la boucle FOR.

Si vous voulez voir de près l'objet sélectionné, vous pouvez taper sur le point du clavier numérique. Si vous voulez basculer entre le mode CAMERA et le mode 3D, vous pouvez taper sur le 0 du clavier numérique.

05° Passer en vue caméra si ce n'est pas le cas et lancer la simulation. La Lune tourne-t-elle bien autour de la Terre lorsque la Terre tourne autour du Soleil ?

Voici le programme :

```
1 - Importation du module Blender Python et des éléments de math
from bpyMeca import Astre, Gestion
from math import cos, sin, pi
```

Cette partie permet d'importer dans le script les bibliothèques qui ne sont pas incluses de base dans Python.

```
# 2-1 - Déclaration des rayons
RAYON_SOLEIL = 695700E3 # rayon en m
RAYON_TERRE =    6371E3 # rayon en m
RAYON_LUNE =     1737E3 # rayon en m
RAYON_MARS =     3396E3 # rayon en m
```

06° Calculer le rapport entre le rayon de la Terre et le rayon de la Lune.

07° Exprimer sur votre copie le rayon de la Terre en km.

```
# 2-2 - Déclaration des distances initiales
ua = 150E9 # distance en m
DIST_TERRE_SOLEIL = ua # distance en m
DIST_TERRE_LUNE = 356700E3 # distance en m
DIST_MARS_SOLEIL = 249E9 # distance en m
```

08° Exprimer la distance Terre-Soleil en km. Utiliser ensuite l'unité la plus adaptée parmi km, Mm et Gm.

```
# 2-3 - Déclaration des masses
MASSE_SOLEIL = 1.989E30 # masse en kg
MASSE_TERRE =  5.972E24 # masse en kg
MASSE_LUNE =   7.342E22 # masse en kg
MASSE_MARS =    6.39E23 # masse en kg
```

09° Les masses sont-elles données en notation scientifique ? Combien de chiffres significatifs a-t-on pour les différentes masses ?

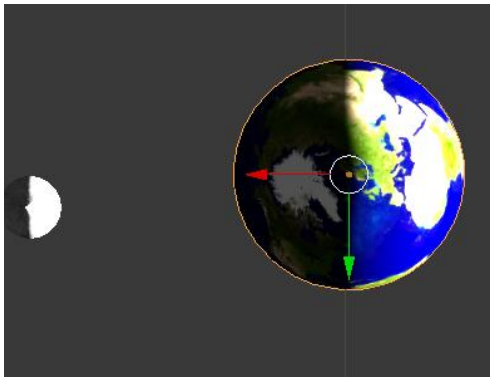
10° Calculez le rapport entre la masse du Soleil et la masse de la Terre.

11° Calculez le rapport entre la masse de la Terre et la masse de la Lune.

```
#2-4 - Déclaration de l'inclinaison, des vitesses de translation et
rotation de la Terre
vit_terre = (0, 29780, 0)
vit_ang_terre = 2*pi/(23.93*3600) # Vitesse angulaire siderale Terre en
rad.s-1
inclinaison_terre = 23/360*2*pi # Inclinaison en radians, 23° sinon
```

La variable **vit_terre** contient 3 valeurs de vitesse initiale. Dans l'ordre, v_X, v_Y et v_Z.

12° Quelle est la seule vitesse initiale non nulle dans cette simulation ? Est-ce cohérent par rapport aux axes visibles sur la première image :

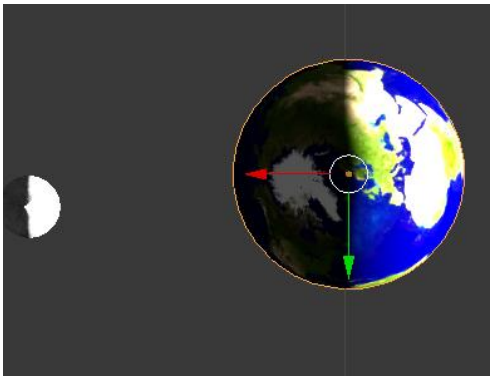


13° S'agit-il des vitesses dans le référentiel géocentrique ou héliocentrique ?

14° Au vu des ombres, positionner la Terre et la Lune sur un schéma dont le Soleil est le centre.

```
# 2-5 - Déclaration de l'inclinaison, des vitesses de translation et
rotation de la Lune
vit_lune = (0, 29780+1000, 0)
vit_ang_lune = 2*pi/(27.3*24*3600) # Vitesse angulaire siderale Terre en
rad.s-1
inclinaison_lune = 6.7/360*2*pi # Inclinaison en radians
```

15° La vitesse du centre de masse de la Lune dans le référentiel géocentrique est de 1000 $m.s^{-1}$.
Pourquoi avoir donner une vitesse initiale de **29780+1000** ?



```
# 3 - Déclaration des constantes de simulation
CARREAU = ua/5 # On définit ici l'unité de mesure pour les distances
INTERVALLE_TEMPS = 2*3600 # Temps simulé en s entre chaque keyframe
FRAMES_ENTRE_CHAQUE_KEYFRAME = 24 # Nbr de frames entre chaque keyframe
```

Pour rappel, **CARREAU** va nous permettre de signaler l'échelle voulue pour cette simulation.
Ici, un carreau représente donc un cinquième d'unité astronomique, soit 30.10^9 m.

La variable **INTERVALLE_TEMPS** contient la durée (en secondes) entre chaque point de calcul.

16° Quel est l'intervalle de temps en heures entre deux calculs ?

La variable **FRAMES_ENTRE_CHAQUE_KEYFRAME** contient le nombre de frames entre deux calculs.

17° La simulation est configurée pour afficher 24 frames par seconde. A quelle durée de temps simulé correspondrait une durée de 2h dans le monde réel ?

```
# 4 - Création des objets-astres Python associés aux objets 3D de Blender
Astre.definir_echelle(CARREAU)
soleil = Astre("Soleil", zoom = 10, rayon = RAYON_SOLEIL, masse = MASSE_SOLEIL)
terre = Astre("Terre", zoom = 20, rayon = RAYON_TERRE, masse = MASSE_TERRE, vitesse = vit_terre,
inclinaison = inclinaison_terre, vit_ang_propre = vit_ang_terre)
lune = Astre("Lune", zoom = 20, rayon = RAYON_LUNE, masse = MASSE_LUNE, vitesse = vit_lune,
inclinaison = inclinaison_lune, vit_ang_propre = vit_ang_lune)
mars = Astre("Mars", zoom = 0, rayon = RAYON_MARS, masse = MASSE_MARS)
camera = Gestion("Camera")
liste_objets = [soleil, terre, lune, camera]
for objet in liste_objets:
  objet.supprimer_keyframes(0, 11000, 1)
```

Le début est standard par rapport aux activités précédentes :

- On choisit l'échelle pour un carreau Blender avec la méthode definir_echelle.
- On crée les objets Python permettant de manipuler les objets Blender 3D :
  - **soleil**, **terre**, **lune** et **mars** pour les astres
  - **camera** pour gérer la caméra de Blender depuis le code Python.

La vraie nouveauté vient de la façon de supprimer les anciennes Keyframes :

- On crée une liste (délimitée par les crochets [..]) qui contient les objets à gérer.
- On lit un à un les objets contenus dans la liste à l'aide de la boucle FOR.

En gros :

```
liste_objets = [soleil, terre, lune, camera]
for objet in liste_objets:
    objet.supprimer_keyframes(0, 11000, 1)
```

revient à écrire :

```
soleil.supprimer_keyframes(0, 11000, 1)
terre.supprimer_keyframes(0, 11000, 1)
lune.supprimer_keyframes(0, 11000, 1)
camera.supprimer_keyframes(0, 11000, 1)
```

Avec uniquement quatre objets dans la liste, cela semble futile mais avec 300, ce serait différent !

POINT IMPORTANT : Les actions à accomplir plusieurs fois doivent nécessairement être décalées avec 4 espaces. Sinon, votre programme va planter.

POINT IMPORTANT 2 : Il faut augmenter le 11000 si vous avez créé des Keyframes au-delà.

```
# 5 - Placement initial des objets-astres

soleil.placement_initial(x = 0, y = 0, z = 0)
terre.placement_initial(x = DIST_TERRE_SOLEIL, y = 0, z = 0)
mars.placement_initial(x = DIST_MARS_SOLEIL, y = 0, z = 0)
lune.placement_initial(x = DIST_TERRE_SOLEIL+DIST_TERRE_LUNE, y = 0, z = 0)

camera.enregistrer_pos(x = 0 , y= 0, z= DIST_TERRE_SOLEIL*0.1)
camera.viser(terre)
```

Rien de bien nouveau non plus.

```
# 6 – Création de l'animation

for x in range(1,30*12):
    Fx,Fy,Fz = terre.force_grav_subie(soleil)
    Fx2,Fy2,Fz2 = terre.force_grav_subie(lune)
    Fx=Fx+Fx2
    Fy=Fy+Fy2
    Fz=Fz+Fz2
    terre.gerer_les_forces(Fx, Fy, Fz, INTERVALLE_TEMPS)

    Fx,Fy,Fz = lune.force_grav_subie(soleil)
    Fx2,Fy2,Fz2 = lune.force_grav_subie(terre)
    Fx=Fx+Fx2
    Fy=Fy+Fy2
    Fz=Fz+Fz2
    lune.gerer_les_forces(Fx,Fy,Fz,INTERVALLE_TEMPS)

    terre.bouger(INTERVALLE_TEMPS, x, FRAMES_ENTRE_CHAQUE_KEYFRAME )
    lune.bouger(INTERVALLE_TEMPS, x, FRAMES_ENTRE_CHAQUE_KEYFRAME )

    camera.filmer(terre, 0, 0, ua/20, x*FRAMES_ENTRE_CHAQUE_KEYFRAME)
```

Ici, nous allons commencer à voir comment le logiciel gère cette simulation :

```
for x in range(1,30*12):
```

Cela veut dire de faire les actions décalées plusieurs fois.
Une première fois avec la variable x contenant 1.
Puis avec x contenant 2 … jusqu'à 30*12 non inclus.
Pourquoi 30*12 ?
Initialement, la simulation simule deux heures entre deux calculs. 2*12 correspond donc à 24h.
Je voulais une simulation traitant de 30 jours. D'où le 30*12.
Nous allons donc faire les calculs suffisamment de fois pour traiter 30 jours de simulation.

18° Modifiez le code pour ne voir que 15 jours de simulation.

```
Fx,Fy,Fz = terre.force_grav_subie(soleil)
Fx2,Fy2,Fz2 = terre.force_grav_subie(lune)
Fx=Fx+Fx2
Fy=Fy+Fy2
Fz=Fz+Fz2
terre. (Fx, Fy, Fz, INTERVALLE_TEMPS)
```

Ici, on calcule les forces Fx en x, Fy en y et Fz en z qu'exercent le Soleil sur la Terre.

Même chose avec Fx2, Fy2 et Fz2 qui sont les forces qu'exercent la Lune sur la Terre.

On additionne les deux et on remplace le résultat dans Fx, Fy et Fz : ces variables contiennent alors les forces totales qui s'exercent sur la Terre.

La dernière ligne permet de prendre en compte ces forces : dans gerer_les_forces, on calcule et mémorise les positions que devra prendre la Terre (2h plus tard dans le monde réel, 24 frames plus loin dans la simulation).

```
Fx,Fy,Fz = lune.force_grav_subie(soleil)
Fx2,Fy2,Fz2 = lune.force_grav_subie(terre)
Fx=Fx+Fx2
Fy=Fy+Fy2
Fz=Fz+Fz2
lune.gerer_les_forces(Fx,Fy,Fz,INTERVALLE_TEMPS)
```

On fait la même chose mais pour la Lune.

19° Quelles sont les astres dont on cherche à prévoir l'influence gravitationnelle sur la Lune ici ?

```
terre.bouger(INTERVALLE_TEMPS, x, FRAMES_ENTRE_CHAQUE_KEYFRAME )
lune.bouger(INTERVALLE_TEMPS, x, FRAMES_ENTRE_CHAQUE_KEYFRAME )

camera.filmer(terre, 0, 0, ua/20, x*FRAMES_ENTRE_CHAQUE_KEYFRAME)
```

Dernière phase : on demande au programme de réellement faire bouger la Terre, la Lune et on enregistre cela avec une Keyframe.

On fait la même chose avec la camera à l'aide de la méthode filmer :

```
camera.filmer(terre, 0, 0, ua/20, x*FRAMES_ENTRE_CHAQUE_KEYFRAME)
```

Cette méthode permet de placer la caméra en la positionnant au centre de l'axe : il faut ensuite fournir trois arguments qui correspondent au décalage relative en x, y et z de la caméra par rapport à la Terre.

Finalement, la caméra s'oriente pour regarder l'astre voulu.

20° En regardant les trois arguments, dire si la caméra est censée filmer la Terre par le dessous, le dessus ou au même niveau ?