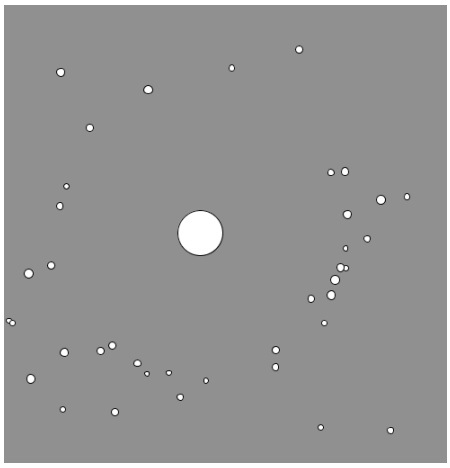
**CNAM • MUX104 Synthèse d'image et réalité virtuelle • 2020 • Pierre Cubaud < cubaud @ cnam.fr >**

# Exercices. Trajectoires

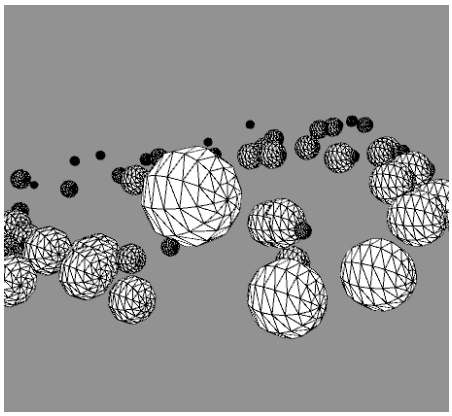


### Exercice 1. Un système solaire 2D

Coder une classe Planete qui gère l'evolution d'une planete de masse négligeable soumise à l'attraction d'une étoile centrale massive. On appliquera une force d'attraction inversement proportionnelle au carré de la distance soleil-planète. On placera initialement la planète au hasard dans l'écran, le soleil étant placé au centre. Le vecteur vitesse initial est dans un premier temps défini au hasard lui aussi.

On appliquera ensuite un vecteur vitesse initial correspondant à une orbite elliptique selon la loi de Kepler (voir sur Wikipedia, les formules de la rubrique "Vitesse orbitale").

Après un test avec une seule planète, modifier le code pour animer N>1 planètes. On ne gère pas les collisions ni l'attraction des planètes entre elles (on est donc très loin de la vraie physique !)



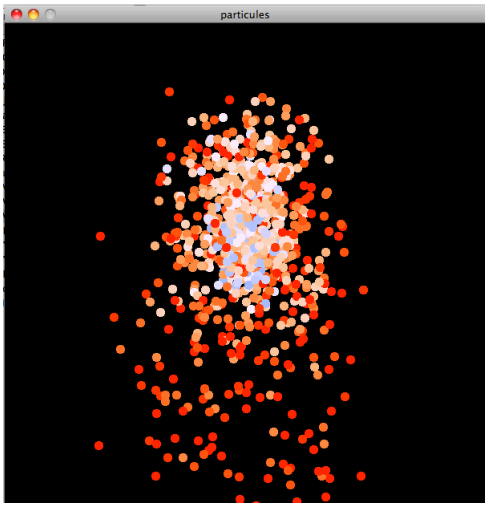
### Exercice 2. Un système solaire 3D

En s'inspirant de l'exercice précédent, donner une version 3D de l'animation.

Pour les fonctions 3D de Processing à utiliser, on se basera sur le TP extrait de MUX101 diffusé dans la séance 1 d'introduction.

### Exercice 3. Particules

Coder le programme de production de particules vu en cours. On utilisera une classe Particule et un arraylist pour gérer la liste des particules actives.
On peut affecter aux particules une couleur correspondant à celle d'un "corps noir" dont la température décroit pendant la durée de vie de la particule. L'intensité rayonnée par un corps noir (comme du charbon de barbecue) est calculée par la formule de Planck (voir Wikipédia). Les tableaux ci-dessous donnent le dosage RVB correspondants à quelques valeurs de températures. Il faut multiplier cette valeur par 255 pour avoir un dosage effectif.




```
// valeurs RGB corps noir pour une temp de 1000 a 10000K avec un pas de 500K
float[] kelvinR = {1.000,1.000,1.000,1.000,1.000,1.000,1.000,1.000,1.000,1.000,1.000,1.000,0.907,0.827,0.762,0.711,0.668,0.632,0.602};

float[] kelvinG = {0.007,0.126,0.234,0.349,0.454,0.549,0.635,0.710,0.778,0.837,0.890,0.937,0.888,0.839,0.800,0.766,0.738,0.714,0.693};

float[] kelvinB = {0.000,0.000,0.010,0.067,0.151,0.254,0.370,0.493,0.620,0.746,0.869,0.988,1.000,1.000,1.000,1.000,1.000,1.000,1.000};
```