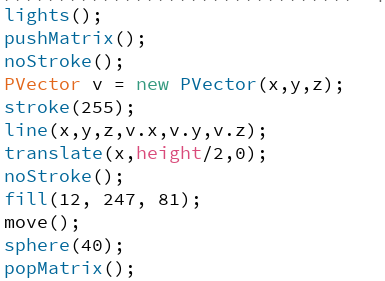
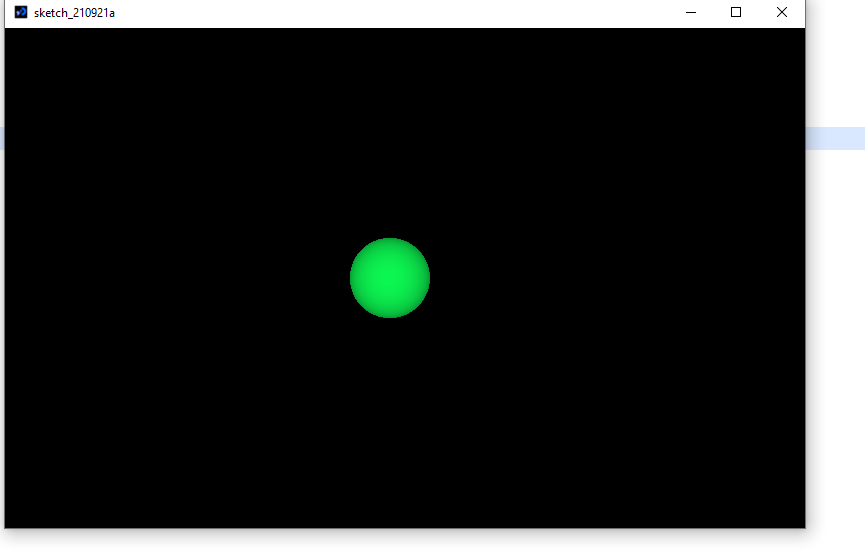
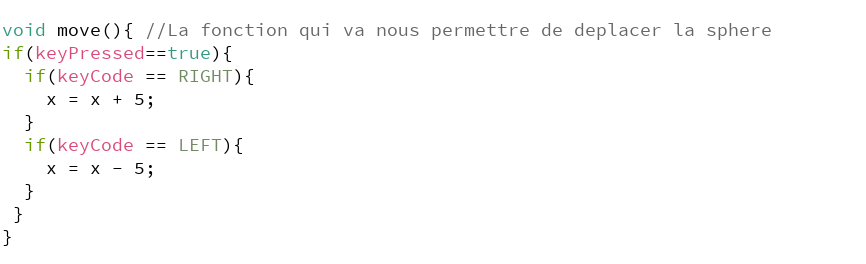
Compte rendu du mini-projet en Processing

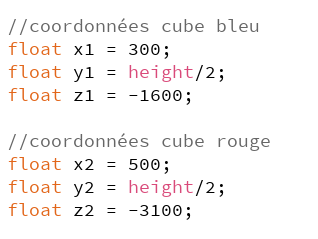
**1-Création de la sphère :**

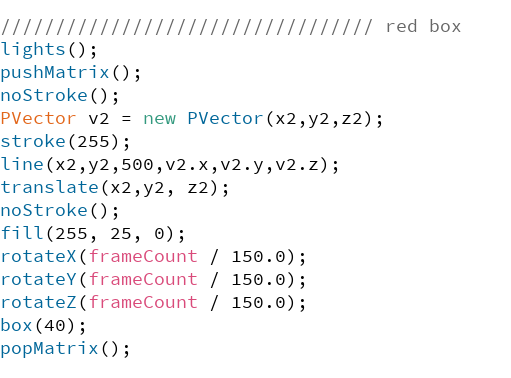
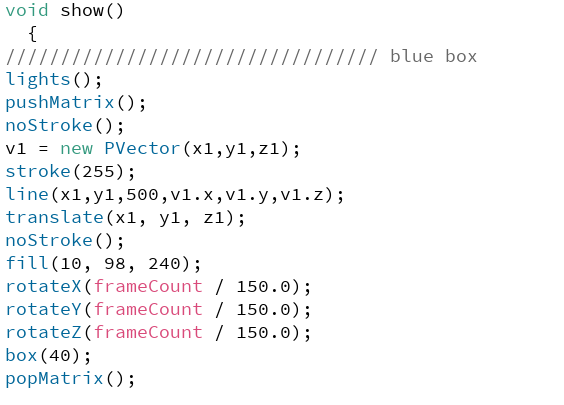
En utilisant sphere() avec une taille de 40 j’ai ensuite ajouté la lumière avec lights() et la couleur verte avec la fonction fill().

J’ai ensuite créer une fonction move() qui vas me permettre de faire bouger la sphère avec les flèches du clavier avec la fonction keyPressed qui vas détecter le bouton touché et faire le mouvement avec la fonction keyCode qui sera soit LEFT soit RIGHT et donc on pourra faire la translation de x ave une vitesse.

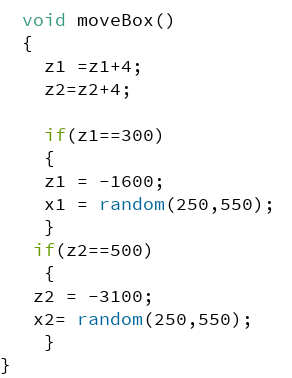


**2-Création des cubes :**

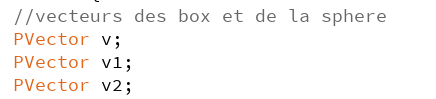
J’ai créé une classe box dans laquelle j’ai créé les coordonnées des deux cubes.



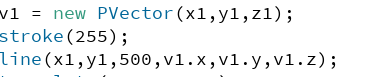
J’ai ensuite créé une méthode show dans laquelle je vais créer les cubes et ajouter la lumière et la couleur rouge et bleu pour les deux cubes avec une taille de 40 j’ai ensuite ajouté les coordonnées ou vont apparaitre les cubes avec translate et j’ai ajouté l’animation de la rotation autour du cube lui-même avec une vitesse précise en utilisant la fonction frameCount.

J’ai ensuite créer une fonction moveBox() qui vas me permettre de faire apparaitre les cubes dans des emplacement aléatoire sur l’axe des x en utilisant la fonction random et de faire un mouvement sur l’axe z afin que cela peut avoir l’interaction avec la sphère(collision) la vitesse de la translation sur l’axe des z est de 4

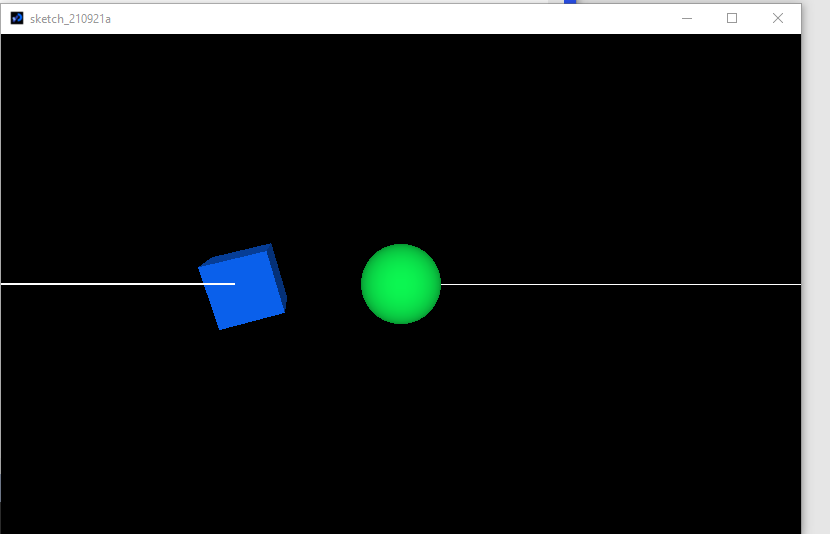
**3-Ajout de la méthode qui vas nous permettre de détecter la collision :**

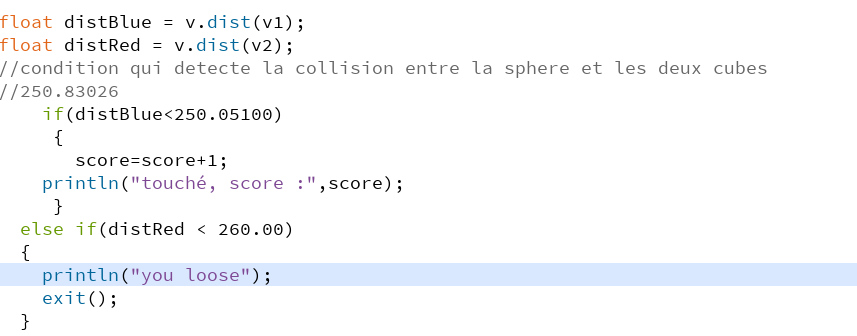
J’ai commencé par créer les vecteurs pour la sphère et les deux cubes qui vont me permettre de voir le point de collision.

J’ai ensuite donnée a chacun les coordonnées spécifique que j’ai choisi à chacun d’eux avec new PVector(x,y,z) j’ai ensuite ajouté line pour voir exactement ou vas se trouver le point de rencontre je me suis inspiré de la vidéo de Daniel Shiffman qu’on a faite en cours concernant la rotation d’une planète autour du soleil donc j’ai utilisé line()pour tracer la ligne avec les coordonnées des formes



Voilà un aperçu : j’ai mis 500 concernant les coordonnées de z afin que je puisse voir la collision avec la sphère

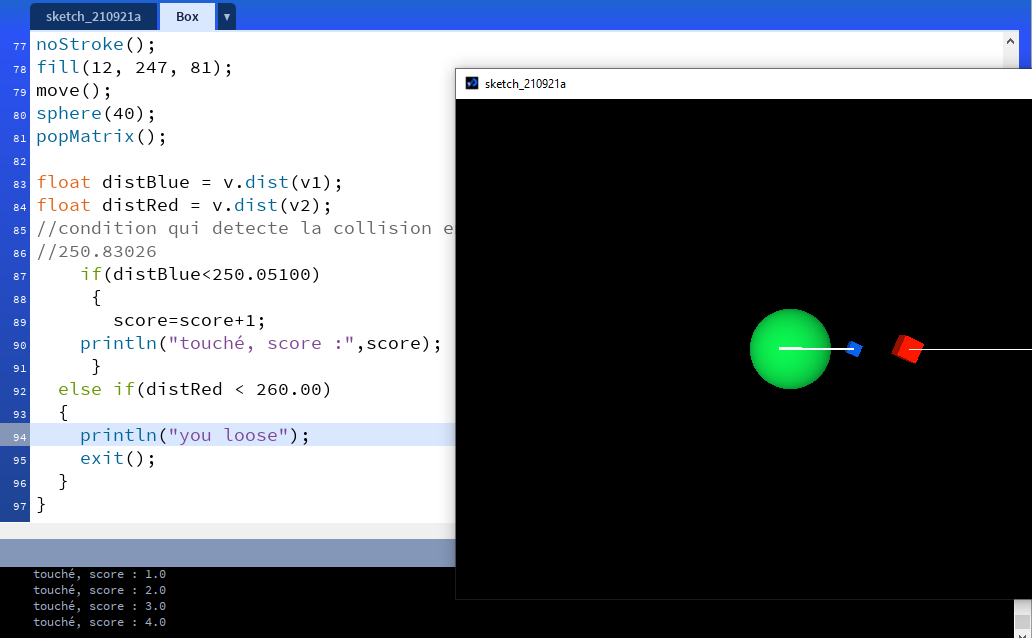


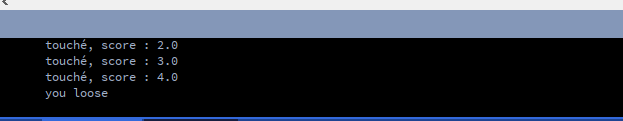
J’ai ensuite créer deux variable de type float qui vont représenté la distance entre les deux cubes et la sphère, j’ai ensuite utilisé la fonction dist() qui vas calculer la distance entre les deux vecteurs que j’ai créer et celui de la sphère ensuite j’ai essayé d’afficher les distances lorsque le cube s’approche de la sphère ensuite j’ai pris une distance approximative lorsque le cube touche la sphère et je l’ai utilisé dans if .Si la distance du cube est inférieur à la distance qu’on a trouvé auparavant lorsque le cube est en contact avec la sphère on augmente le score avec une variable score que j’ai ajouté et que j’incrémente à chaque fois qu’il y’a collision sinon si le cube rouge est touché on envoie un message comme quoi le joueur à perdu et on sort du programme avec exit().

Finalement dans la classe principale j’ai créé la méthode setup() on je vais définir la taille de la fenêtre la couleur de l’arrière-plan et ou j’a instancié la classe ou se trouve les autres objet.

Une image contenant texte

Description générée automatiquementEnsuite j’ai créé la fonction draw afin de m’afficher tout ce que j’ai défini : les cubes et la sphère et en appelant les méthodes de mouvement de la sphère et l’apparition des cubes sur les emplacement aléatoire.





**4-Difficultés rencontrés :**

La partie qui m’a pris le plus de temps est la partie concernant la collision et le score car la méthode que j’ai utilisée ne me semble pas la meilleur car prendre une valeur approximative à celle de la collision n’est pas toujours fiable mais j’ai trouvé cela intéressant puisque j’ai utilisé presque tous les notions vue en cours surtout la partie des objets 3d concernant le système solaire. J’aimerai bien savoir s’il y’a de meilleure solution que celle que j’ai utilisé.