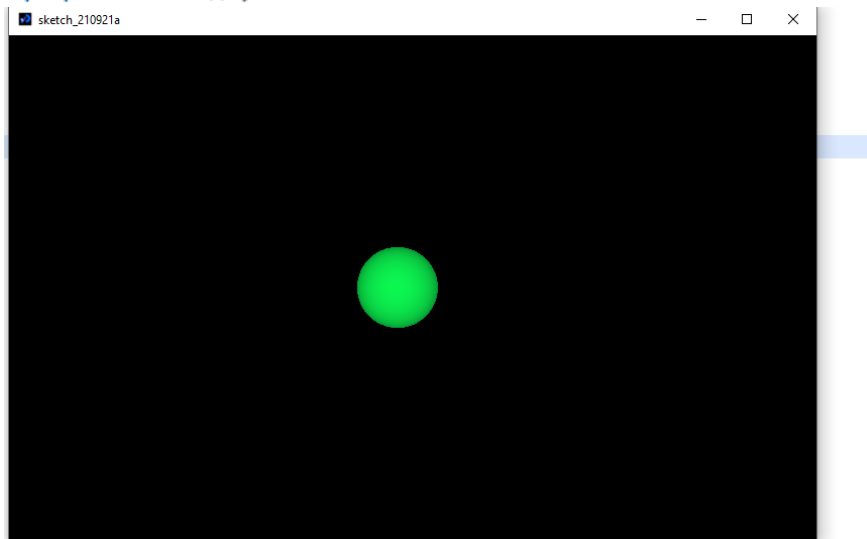


Compte rendu du mini-projet en Processing

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()`.

```
lights();  
pushMatrix();  
noStroke();  
PVector v = new PVector(x,y,z);  
stroke(255);  
line(x,y,z,v.x,v.y,v.z);  
translate(x,height/2,0);  
noStroke();  
fill(12, 247, 81);  
move();  
sphere(40);  
popMatrix();
```



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.

```

void move(){ //La fonction qui va nous permettre de deplacer la sphere
if(keyPressed==true){
    if(keyCode == RIGHT){
        x = x + 5;
    }
    if(keyCode == LEFT){
        x = x - 5;
    }
}
}
}

```

2-Création des cubes :

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

```

//coordonnées cube bleu
float x1 = 300;
float y1 = height/2;
float z1 = -1600;

//coordonnées cube rouge
float x2 = 500;
float y2 = height/2;
float z2 = -3100;

void show()
{
//////////////////////////////////////////////////// blue box
lights();
pushMatrix();
noStroke();
v1 = new PVector(x1,y1,z1);
stroke(255);
line(x1,y1,500,v1.x,v1.y,v1.z);
translate(x1, y1, z1);
noStroke();
fill(10, 98, 240);
rotateX(frameCount / 150.0);
rotateY(frameCount / 150.0);
rotateZ(frameCount / 150.0);
box(40);
popMatrix();

////////////////////////////////////////////////// red box
lights();
pushMatrix();
noStroke();
PVector v2 = new PVector(x2,y2,z2);
stroke(255);
line(x2,y2,500,v2.x,v2.y,v2.z);
translate(x2,y2, z2);
noStroke();
fill(255, 25, 0);
rotateX(frameCount / 150.0);
rotateY(frameCount / 150.0);
rotateZ(frameCount / 150.0);
box(40);
popMatrix();
}

```

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 où vont apparaître 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

```
void moveBox()  
{  
    z1 =z1+4;  
    z2=z2+4;  
  
    if(z1==300)  
    {  
        z1 = -1600;  
        x1 = random(250,550);  
    }  
    if(z2==500)  
    {  
        z2 = -3100;  
        x2= random(250,550);  
    }  
}
```

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

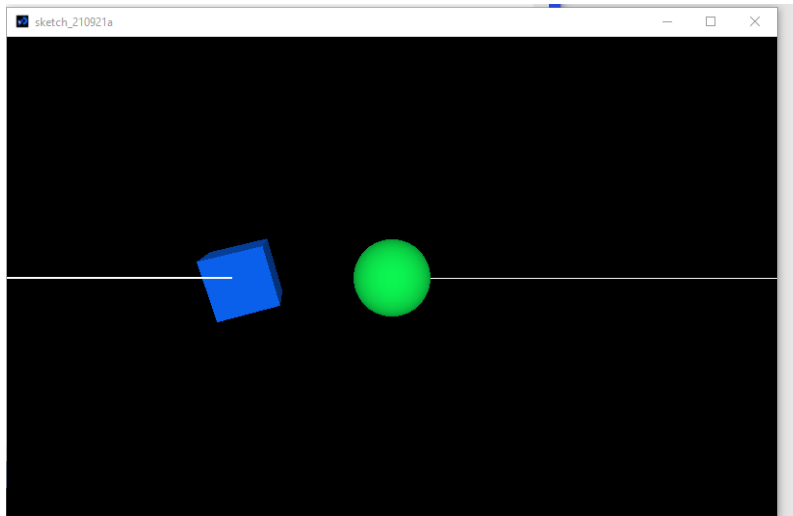
J'ai commencé par créer les vecteurs pour la sphère et les deux cubes qui vont me permettre de voir le point de collision.

```
//vecteurs des box et de la sphere  
PVector v;  
PVector v1;  
PVector v2;
```

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

```
v1 = new PVector(x1,y1,z1);  
stroke(255);  
line(x1,y1,500,v1.x,v1.y,v1.z);
```

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()`.

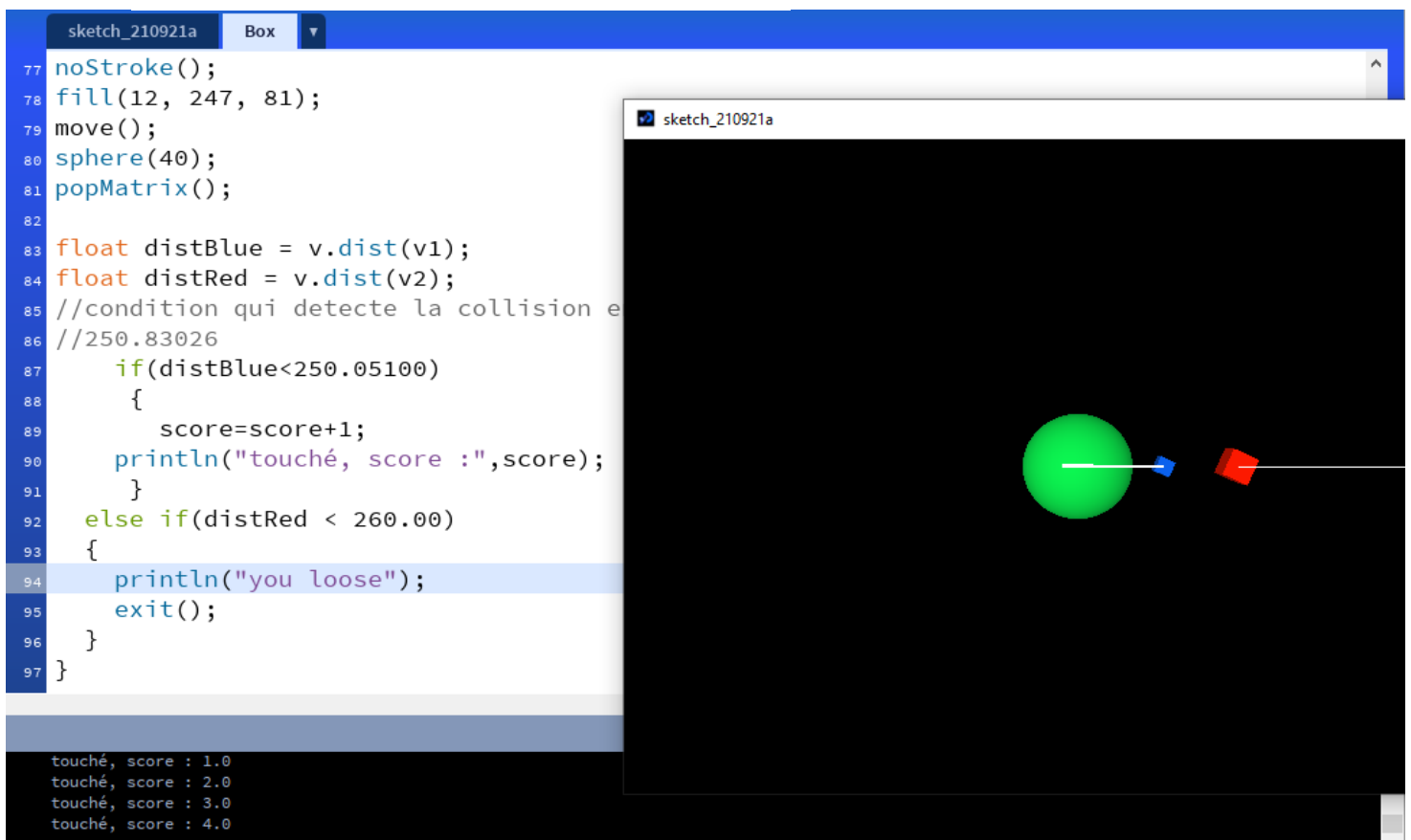
```
float distBlue = v.dist(v1);
float distRed = v.dist(v2);
//condition qui detecte la collision entre la sphere et les deux cubes
//250.83026
if(distBlue<250.05100)
{
    score=score+1;
    println("touché, score :",score);
}
else if(distRed < 260.00)
{
    println("you loose");
    exit();
}
```

Finalement dans la classe principale j'ai créé la méthode setup() on va définir la taille de la fenêtre la couleur de l'arrière-plan et où j'ai instancié la classe où se trouvent les autres objets.

Ensuite 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 emplacements aléatoires.

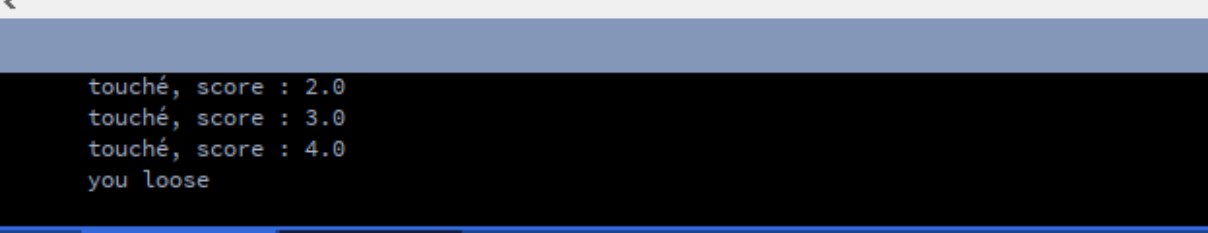
```
float x=400;//coordonnées de translation
float y;
float z;

Box box;
void setup(){
  size(800,500,P3D);
  background(0);
  box = new Box();
}
void draw(){
  background(0);
  box.show();
  box.moveBox();
}
```



```
sketch_210921a  Box
77 noStroke();
78 fill(12, 247, 81);
79 move();
80 sphere(40);
81 popMatrix();
82
83 float distBlue = v.dist(v1);
84 float distRed = v.dist(v2);
85 //condition qui detecte la collision e
86 //250.83026
87 if(distBlue<250.05100)
88 {
89   score=score+1;
90   println("touché, score :",score);
91 }
92 else if(distRed < 260.00)
93 {
94   println("you loose");
95   exit();
96 }
97 }

touché, score : 1.0
touché, score : 2.0
touché, score : 3.0
touché, score : 4.0
```

A screenshot of a terminal window with a dark background and a light blue header bar. The terminal displays four lines of text in a monospaced font: 'touché, score : 2.0', 'touché, score : 3.0', 'touché, score : 4.0', and 'you loose'.

```
touché, score : 2.0  
touché, score : 3.0  
touché, score : 4.0  
you loose
```

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'aimerais bien savoir s'il y'a de meilleure solution que celle que j'ai utilisé.