HAUTE ÉCOLE D'INGÉNIEUR DU VALAIS

RAPPORT

PROJET LUNARLANDRY

TEAM PLS

INF1

Pablo Stoeri
Landry Reynard
Samy Francelet

19 juin 2020

Table des matières

1 Introduction

1.1 Contexte

1.2 Objectif du document

1.3 Spécifications

1.4 Problématique

2 Méthodes

2.1 Méthodes

Pour ce projet, nous sommes partis de grandeurs et de données physique comme la gravité, la masse et l'accélération. Cela nous a permis de construire une base solide de physique pour ensuite venir ajouter les différents éléments de programmation derrière. De manière générale, nous pensons qu'il est important de se concentrer sur les faits réels de la vie avant de commencer à coder des interfaces graphiques ou autres. Voici la physique que nous avons utilisé :

$$\overrightarrow{F} = M \cdot \overrightarrow{a}$$

Qui est transformé pour l'accélération en chute libre par :

$$\overrightarrow{F}_{grav} = M \cdot \overrightarrow{g}$$

Où g = -9.81 ms-2 sur la terre et g = -1.62 ms-2 et M est la masse de l'objet soumis à la gravité, dans notre cas le vaisseau spatial. Ensuite, nous avons aussi utilisé le coefficient de frottement en rapport avec la densité de l'air :

$$\overrightarrow{F}_{friction} = -k \cdot \overrightarrow{v}$$

Où k est le coefficient de friction de l'atmosphère,en l'occurrence k = 0 sur la lune vu qu'il n'y a pas d'air. De ce fait, tant que le vaisseau est en l'air, seul la force de gravité a un impact sur le vaisseau lorsque les moteurs sont éteints.

3 Résultats

3.1 Choix d'implémentation et variations effectuées

3.1.1 PhysicsSimulator

non

3.1.2 Ground

OUi

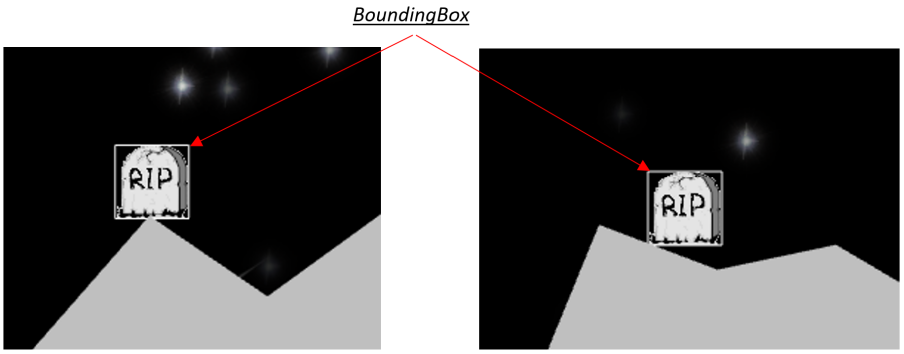

FIGURE 3.1 – Collision entre un point du polygone et le rectangle et inversement

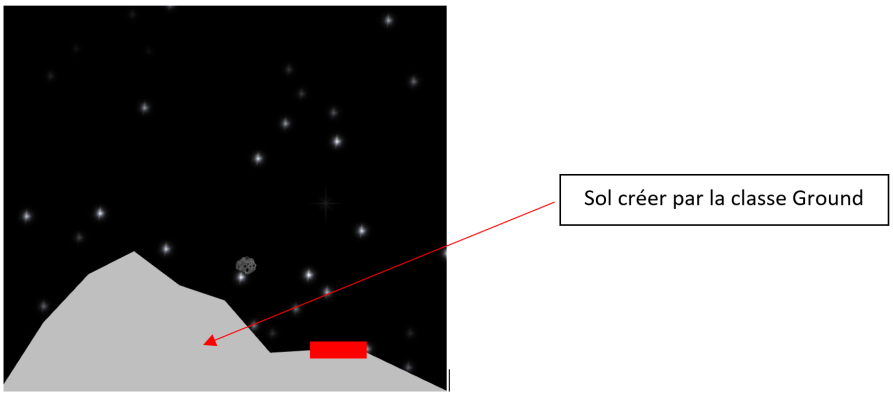

FIGURE 3.2 – Image d'illustration du polygone

4 Bilan

4.1 Problèmes rencontrés et solutions apportées

4.2 Description des fonctionnalités

4.3 Améliorations possbibles

4.4 Conclusion

5 Annexe

5.1 Test Prog

```
0  public class SortApplication {
2    static void displayArray(int[] array) {
       String value = "";
4      for (int i = 0; i < array.length; i++) {
         value = value + array[i] + ",";
6      }
       System.out.println(value);
8    }
10   public static void main(String[] args) {
       //Creation Tableaux
12     int MaxValue = 140000;
       int a[][] = new int[10][];
14     int b[][] = new int[10][];
       int c[][] = new int[10][];
16
18     for (int i = 0; i < a.length; i++) {
         a[i] = ArrayFactory.createRandomArray(MaxValue, 50000);
20     }
       for (int i = 0; i < a.length; i++) {
22       b[i] = ArrayFactory.createInvertedSortedArray(MaxValue);
       }
24     for (int i = 0; i < a.length; i++) {
         c[i] = ArrayFactory.createShuffleArray(MaxValue);
26     }
28
       long startTimeRandom = System.nanoTime();
30     for (int i = 0; i < a.length; i++) {
         SelectionSort.sort(a[i]);
32     }
       long endTimeRandom = System.nanoTime();
34
       long startTimeInverted = System.nanoTime();
36     for (int i = 0; i < a.length; i++) {
         SelectionSort.sort(b[i]);
38     }
```

```
    long endTimeInverted = System.nanoTime();
40
    long startTimeShuffle = System.nanoTime();
42  for (int i = 0; i < a.length; i++) {
      SelectionSort.sort(c[i]);
44  }
    long endTimeSuffle = System.nanoTime();
46

48  System.out.println((endTimeRandom - startTimeRandom) + " [ns]");
    System.out.println((endTimeInverted - startTimeInverted) + " [ns]");
50  System.out.println((endTimeSuffle - startTimeShuffle) + " [ns]");

52  }

54 }
```