Rapport mini-Projet Algorithmique

Réalisé par :

AJLI Zakaria

Année universitaire 2021-2022

Introduction

Le but de ce projet est de générer automatiquement des rectangles alignés sur les axes et donc on essaye de trouver les rectangles qui intersectent d'autres rectangles et cela en utilisant 3 méthodes vu en cours : force brute, diviser pour régner et « sweep-Line ».

Création des rectangles

Pour la création des rectangles aléatoire j'ai créé une classe rectangle et j'ai ensuite créer a l'intérieur de la classe une méthode qui crée des rectangles dans des emplacement aléatoire sur les axes et avec des tailles aléatoire aussi.

projet_Algorithmique_2021_2022 | rectangles

```
class Rectangle
{
  float x=random(0,width-50);
  float y=random(0,height-50);
  float w = random(0,40);
  float h = random(0,70);
void show()
{
  strokeWeight(3);
  rect(x, y, w, h);
}
}
```

J'ai ensuite créé une Arraylist où je vais stocker tous les rectangles afin que je puisse les utiliser pour appliquer les algorithmes vus en cours avec n qui représente le nombre de rectangle qu'on veut afficher.

```
int n = 3;
Rectangle rectangle;
ArrayList<Rectangle>rect = new ArrayList<Rectangle>();
void setup()
{
```

Fonction d'intersection

J'ai ensuite créé une fonction qui détecte si les rectangles se coupent

Une méthode de type Boolean qui renvoie vrai s'il y'a collision.

Dans cette partie du code et plus précisément dans cette fonction je me suis renseigné sur la façon de détecter la collision sur une vidéo que j'ai trouvé sur YouTube voilà son lien et j'ai aussi consulter un autre lien que j'ai trouvé intéressant qui parle des collisions en général avec des points des cercles … voilà les liens :

https://www.youtube.com/watch?v=O8VyU4j6dT0&t=913s

https://happycoding.io/tutorials/processing/collision-detection

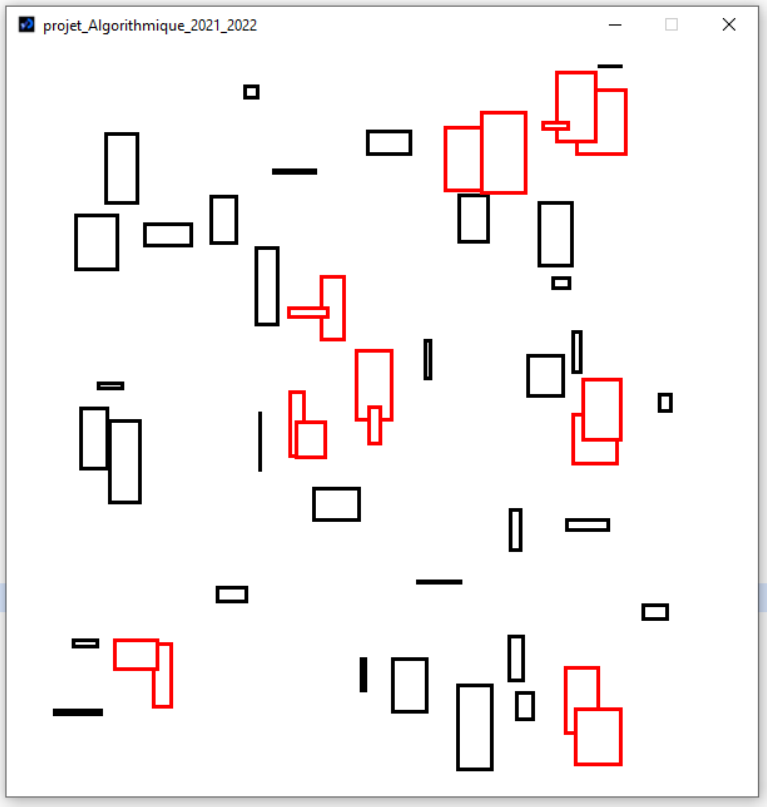
```
//fonction d'intersection
boolean intersection(Rectangle rect1,Rectangle rect2)
{
  distanceX = abs((rect1.x+rect1.w/2)-(rect2.x+rect2.w/2));
  distanceY = abs((rect1.y+rect1.h/2)-(rect2.y+rect2.h/2));
  float largeur = rect1.w/2+rect2.w/2;
  float hauteur = rect1.h/2+rect2.h/2;
  if(distanceX < largeur){
     if(distanceY < hauteur){
       return true;
     }
  }
  return false;
}
```

Brute force

```
//Brute force
  int start = millis();
  for(int i=0;i<n;i++){
    for(int j=i+1;j<n;j++){
   if(intersection(rect.get(i),rect.get(j))==true){
    stroke(255,0,0);
    rect.get(i).show();
    rect.get(j).show();

   }
    }
    int temp = millis()- start;
    println("temps brute force = "+temp);
```

J'ai créé deux boucles for qui vont me permettre de vérifier pour chaque rectangle si il y'a collision et j'ai afficher le temps de calcule pour 50 rectangle en utilisant la fonction millis() qui retourne un nombre en milliseconde depuis le start du brute force, ce qui m'as donné comme résultat :



Sweep-line

Diviser pour régner

Conclusion

Concernant la création des rectangles je n’ai eu aucun problème j’ai eu quelque difficulté avec l’algorithme sweep-line je n’ai pas réussi à l’implémenter et j’ai aussi rencontré des difficultés avec la méthode diviser pour régner en globalité ce projet m’as pris toute la semaine pour le travailler avec une moyenne de deux heure par jour. C’est un projet qui m’a beaucoup aidé j’ai bien compris la méthode force brute et les deux autres méthodes diviser pour régner et sweep-line mais je n'ai pas réussi à les implémenter.