Rapport programmation concurrente

ŚALVAN Fabien, L3 informatique

10 octobre 2017

Résumé

Ce rapport comprend l'explication detaillée de chaque exercice du TP de programmation concurrente a rendre. Pour chaque exercice , les fonctions principales sont expliquées ainsi que l'architecture du programme.

0.1 Introduction

Dans le cadre du TP de programmation concurrente , on a eu a faire un exercice impliquant des balles , avec les consignes suivantes

- Les balles ne peuvent sortir du cadre de la fenêtre et ricochent contre les bords
- Un bouton permet de démarrer/arrêter le mouvement de toutes les balles. Lorsque les balles sont en mouvement, ce bouton a pour étiquette “stop” et lorsque les balles sont à l’arrêt il a pour étiquette “start”
- Un bouton “+” permet d’ajouter une nouvelle balle de couleur aléatoire. Lorsqu’un seuil fixé au préalable est atteint, plus aucune balle ne peut être ajoutée.
- Un bouton “-” permet de supprimer une balle.
- — Lorsqu’une collision se produit, les balles impliquées sont supprimées
- Un score est affiché en permanence et est incrémenté à chaque collision.
- Une horloge affiche le temps écoulé pendant que les balles sont en mouvement ; cette horloge doit stopper son décompte lorque les balles sont arrêtées et reprendre son décompte lorsque les balles sont à nouveau en mouvement.

Les bibliotheques utilisées sont swing pour java et tkinter pour python

0.2 balles en java

Pour faire cet exercice , il a fallu diviser le code en plusieurs classes , chacune ayant son role , et certaines etant executées en parralele afin de maximiser les performances.

0.2.1 Fenetre

Fenetre : cette classe est derivée de la classe jframe de java , sert a ouvrir une fenetre pour faire le programme en affichage graphique

0.2.2 Affiche

Affiche : Ce thread s’occupe juste de rafraichir en boucle la classe Fenetre , afin de pouvoir afficher les changement de positions des balles ainsi que l’UI . En changeant le temps de sleep de ce thread , on peut modifier la vitesse de la simulation.

```
while(true){
fenetre.repaint();
sleep(1);
}
```

0.2.3 Ball supervisor

Ball supervisor : ce thread gere la physique des balles , c’est a dire il s’occupe en meme temps de leur deplacement , ainsi que des collisions ce thread , comme tout les autres ce thread s’occupe de modifier la position des balles a chaque fois , en fonction de leurs position et de leurs vitesse . Ilgere aussi les collisions.

```
while(true){
if(fenetre.pause == false){
for(int i=0;i<fenetre.dessins.size();i++){
fenetre.dessins.get(i).new_pos();
}
for(int i=0;i<fenetre.dessins.size();i++){
for(int j=0;j<fenetre.dessins.size();j++){
if(i !=j){
```

```
if(fenetre.dessins.get(i).collision(fenetre.dessins.get(j))){
fenetre.dessins.remove(j);
fenetre.dessins.remove(i);
fenetre.score_value +=1 ;
}
}
} }
sleep(10);
}
else{
sleep(10);
}
```

0.2.4 Ball

Ball : cette classe est utilisée pour instancier les balles . elle contient toutes les fonctions necessaire au fonctionnement des balles . Elle contient entre autres les fonctions utilisées par ball supervisor pour gerer la collision et la physique .

- distance : calcule la distance de la balle avec une autre
  ```
  public double distance (Ball other){
  int dx = other.x - x;
  int dy = other.y - y;
  return Math.sqrt((dx*dx) + (dy*dy));
  }
  ```
- colision : determine si une autre balle est en colision ou pas
  ```
  public boolean collision(Ball other){
  if (((int)distance(other) <= (size+ other.size))){
  return true;
  }
  else{
  return false;
  }
  ```
- new pos : determine la prochaine position de la balle
  ```
  public void new_pos(){
  wall_colision();
  x = x+vx;
  y = y+vy;
  }
  ```
- ajout : ajoute une balle

0.2.5 Timer

Timer : thread utilisé pour faire fonctionner le chronometre de la fenetre. Il est lancé dans un thread a part afin que les autres thread n'interferent pas avec lui , de maniere a ne pas perdre en precision

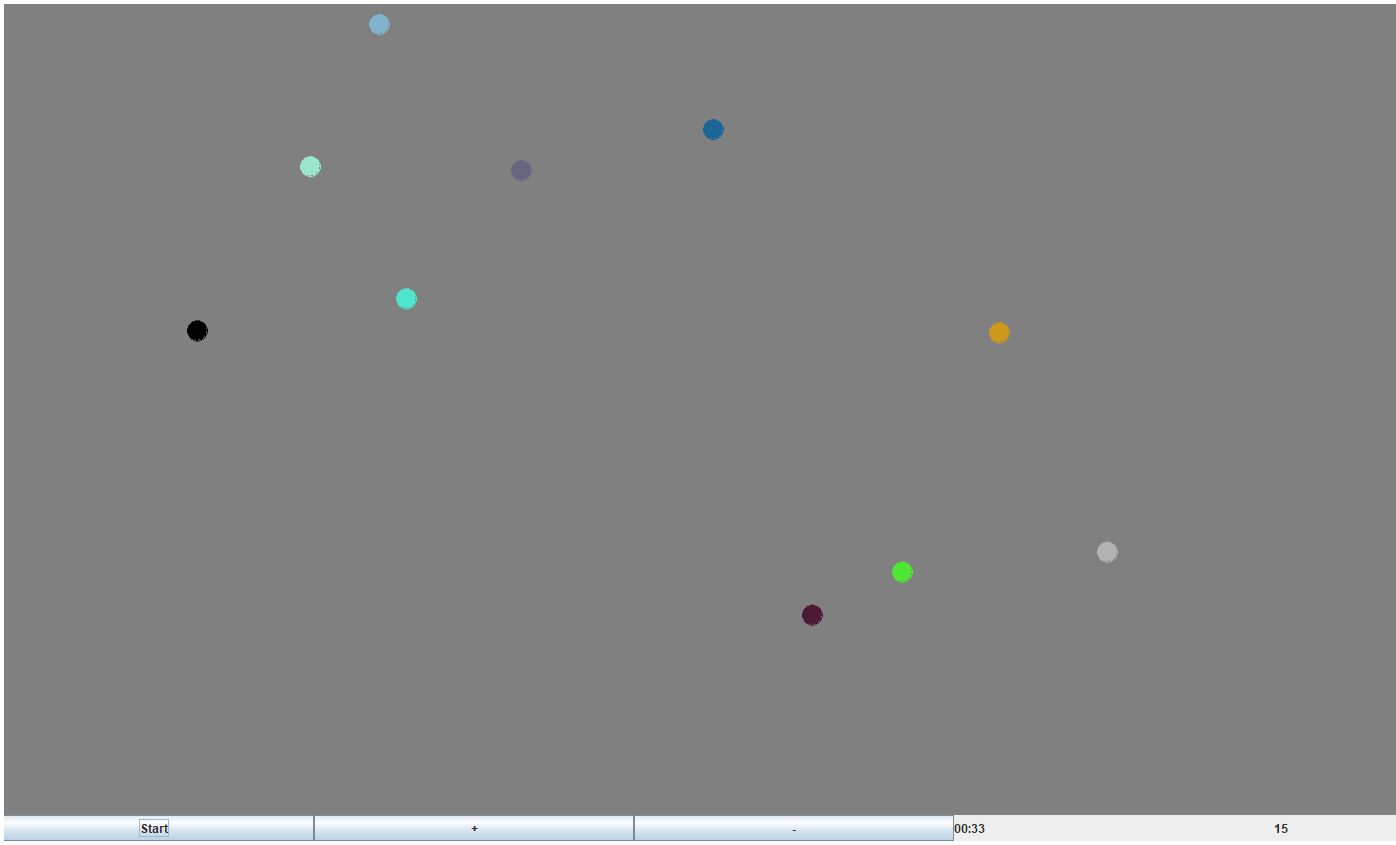
```
while(true){
if(fenetre.pause == false){
fenetre.time +=1;
sleep(1000);
}
else{
sleep(1000);
}
```

0.2.6 Main

Main : classe pour executer le programme . le main est juste chargé de demarrer tout les threads neccessaires au bon fonctionnement du programme , et de créer une boucle infinie pour garder le programme ouvert

```
public static void main(String[] args) {

Fenetre f = new Fenetre("fenetre");
Ball_supervisor s = new Ball_supervisor(f);
Affiche a = new Affiche(f);
Timer t = new Timer(f);
a.start();s.start();
t.start();
}
```



et voici le resultat

0.2.7 points delicats/interessants

— points delicats : il fallait reaprendre a programmer une fenetre en java , ce qui a necessité une certain temps . De plus , il a été assez delicat de coder la collision entre les balles

— points interessants :ca a été interessant d'arriver a programmer quelque chose en fenetre et de plus complexe que les exercices precedents. Il a aussi été interessant de pouvoir créer un systeme de collision , ce qui permet de comprendre comment marchent les colisions en general dans des logiciels plus complexe

0.3 balles en python

0.3.1 main

La classe main permet d'afficher la fenetre , ainsi que de gerer l'UI. Elle contient les fonctions permettant de gerer l'UI de la fenetre

— pause : gere la pause
— retrait : enleve une balle
— close : ferme la fenetre
— ajout : ajoute une balle

0.3.2 ball

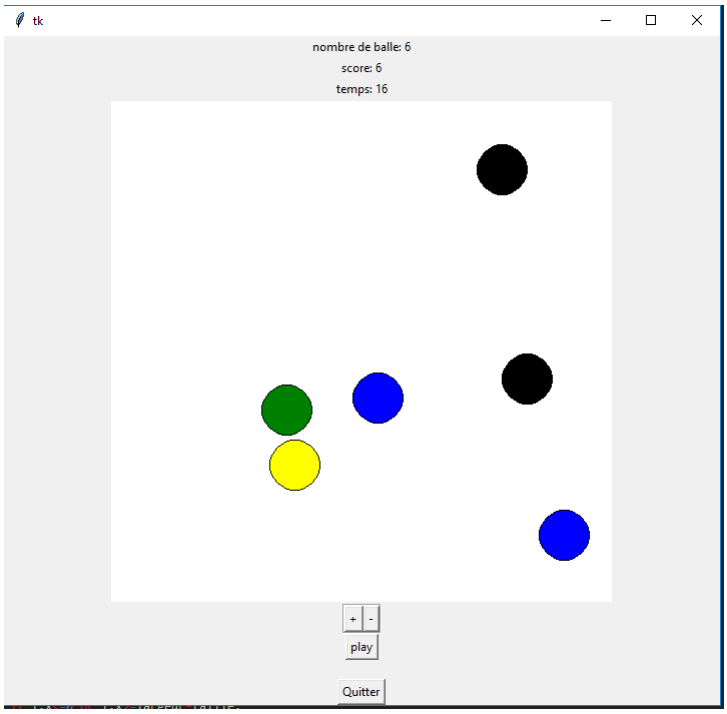
Comme en java , la classe ball contient le code necessaire a instancier des balles . Les collisions entre les balles et les murs sont aussi gerée dans cette classe , grace a la fonction :

```
def collision(self,p):
x=p.x-self.x
y=p.y-self.y
dist=x*x+y*y
if (sqrt(dist))<=(taille):
fenetre.nb-=2
fenetre.nb_ball["text"]=("nombre de balle: {}".format(fenetre.nb))
fenetre.score+=2
fenetre.n_score["text"]=("score: {}".format(fenetre.score))
fenetre.canvas.delete(self.name)
ball.liste.remove(self)
fenetre.canvas.delete(p.name)
ball.liste.remove(p)
```

0.3.3 calcul

Le thread calcul permet de gerer la physique , c'est a dire de deplacer les balles en fonction de leurs position et de leur vitesse

Et voici le resultat final :



0.3.4 points delicats/interessants

points delicats : il fallait reaprendre a programmer en python , etant donné que on en a fait que en 1ere année et avec des choses basiques , en plus des difficultées liées a la programmation en fenetre et en thread

0.4 Conclusion

En conclusion , ce cours a permis de bien comprendre les difficultées et les interet de la programmation par thread , ainsi que de pouvoir les mettre en application dans plusieurs languages de programmation .

0.5 Bibliographie

— documentation java = http://www.oracle.com/technetwork/java/javase/documentation/index.html
— documentation python = https://docs.python.org/3/