université
PARIS-SACLAY

# ***Rapport du projet***

# ***Frogger***

# ***Pour le module d'IPO***


**- DAHOUMANE Ahcene**

**- MAKELA N'semi Euloge**

**Introduction :**

On a coder dans ce projet un jeu qui nous permet de déplacé une grenouille dans un environnement de jeu en évitons de tapé des voitures qui circule aléatoirement dans cette environnement, ce jeu se compose de 3 partie :

1- **Partie 1 :** avec un environnement fourni on doit juste créer une grenouille et la faire déplacer jusqu'à la ligne d'arrivée et affichons un message victoire ou défaite dans le cas où les voitures touche la grenouille avant la ligne d'arrivé . On a commencé par les méthodes testwin() et testlose() dans la Class game. Puis on codé la Class frog en mettent les constructeur de class ainsi que implémenter les méthodes de l'interface Ifrog ajouter les déplacements de la grenouille dans l'environnements , pour cela on a utilisé « le switch » ANNEXE1 , mais on ces rendu compte qu'il faut utilisé if(ANNEXE2) car on a besoin d'un deuxième condition pour éviter que la grenouille dépasse l'environnement.

```
public void move(Direction key){
        switch (key) {
                case up:
                        p = new Case (p.absc ,p.ord+1);
                        break;
                case down:
                        p = new Case (p.absc ,p.ord-1);
                        break;
                case right:
                        p = new Case (p.absc+1 ,p.ord);
                        break;
                case left:
                        p = new Case (p.absc-1 ,p.ord);         }
}
```

ANNEXE1

```
public void move(Direction key) {
    this.direction = key;
    if (key == Direction.up ) {
        game.setScore(1);
    }
    if (key == Direction.down) {
        game.setScore(-1);
    }
    if (key == Direction.left && (position.absc - 1 >= 0)) {
        this.position = new Case( absc: position.absc - 1,  ord: 0);
    }
    if (key == Direction.right && (position.absc + 1 < this.game.width)) {
        this.position = new Case( absc: position.absc + 1,  ord: 0);
    }
```

ANNEXE2

**Partie2 :**Dans cette partie-là ,c'est à notre tour de créer l'environnement qui contient les voiture et les lignes dont on va placé ces voiture , pour cela on a besoin de 3 Class  (environnement – Car – lane ) , on mettra dans Car les diffèrent méthodes de voitures comme (DeplaceVoiture(fait apparaitre les voiture de gauche à droite et de droite à gauche) …etc.) et dans Lane les méthodes comme(update….etc.).
On a rencontré sur cette partie quelque difficulté ,car au début on a pas ajouté sur l'environnement la ligne de départ et celle d'arrivé (ANNEXE 3) ainsi que notre jeux nous donne une fenêtre gris au début car au lieu d'appelé addToGraphique on a appelé plus tôt mayAddCar (ANNEXE4).

```
lanes.add(new Lane(game,  ord: 0,  density: 0.0D));
for(int i = 1; i < game.height-1; i++) {
    Double density = game.randomGen.nextDouble() * (0.7-0.2);
    Lane e = new Lane(game, i, density);
    lanes.add(e);
}
lanes.add(new Lane(game,  ord: game.height-1,  density: 0.0D));
```

ANNEXE3

```
public void update() {
    temp++;
    if(temp >= speed) {
        for (Car maVoiture : cars) {
            maVoiture.DeplaceVoiture();
        }    //temp++;
            mayAddCar();
            temp = 0;
            //if(RandomAleatoire()){ mayAddCar();}
    }
        for(Car maVoiture : cars) {
            maVoiture.addToGraphics();
        }
}
```

ANNEXE4

**Partie3 :** Dans cette partie on veut que notre jeux vas jusqu'à l'infini donc y'aura pas de victoire , on a commencé par créer des classe EnvInf et FrogInf , on c'est basé sur le principe que la grenouille ne bouge pas en haut ou en bas , c'est juste que a chaque fois qu'on veut monté ca nous rajoute une ligne sur l'environnement .
On a rencontré comme difficulté sur cette partie que même si nos voiture percute la grenouille on perd pas , on ces rendu compte que le problème viens de IsSafe de Enivrement  donc on à réglé ça en changent son contenue(ANNEXE5)

```
@Override
public boolean isSafe(Case c) {
    return this.lanes.get(c.ord).isSafe(c);
}
```

ANNEXE5

**Partie4** : dans cette partie on ajouter un Timer qui nous calcule le temps de jeu qui commence a compté du moment où on a exécuté le jeu jusqu'à ou on perd et on a affiche le temps totale de la partie .
On a utilisé au début system.currentTime sauf que cette méthode là on a pas réussi à la stoppé donc on a du se servir de TimerSchedule.

**Conclusion :** ce projet nous a appris plusieurs façon de codé et qu'on peut codé une méthode de plusieurs façon , et comment appelé des méthodes en java ainsi que les porté des class et d'interface ainsi on a découvert comment utilisé GitHub et comment travaillé en groupe sur un projet .