Analyse des algorithmes de tris

DIARE Youssouf
DIALLO Boubacar Sadio
Olangassicka Franck Loick
EMAM Mohamed El Mamy

Université de Caen Normandie
L3 Info
Groupe 2A

January 14, 2025

1. Présentation du projet
   - Problématique
   - Objectifs
2. Décomposition des taches
   - Générateur de donnée (désordre)
   - Implémentation des algorithmes
   - visualisation
   - Le Héros
     - L'Ennemi
3. Début de l'expérimentation
4. conclusion

Problematique

Je suis la problematique .

Les objectifs

ici il y aura la listes des objectifs

cette partie parle du generateur de desordre

Implementation des algorithmes

cette partie parlera des algos que l'on a implementer et si on a utiliser un design paterne pour le faire

visualisation des algorithme

cette partie parle de la visualisation des algos

Initialisation du Héros

```
#Initialisation du héros
self.vie = vie
self.vieMax = vie
self.vieMin = 0
self.degat = degat
self.compteurTour = 0

#Initialisation de l'ennemi
self.ennemi = Ennemi(vieEnnemi,nombreDeTour, self.fenetre)

#Importation de l'image du héro
self.image = pygame.image.load("Img/heros.png").convert_alpha()
self.imageDegat = pygame.image.load("Img/herosDamage.png")
self.imageHeal = pygame.image.load("Img/herosHeal.png")
```

Figure: Extrait code init() class Heros

Fonctions du Héros

```
def VieHeros (self,nombreDeCasse):
    # nombreDeCasse[1] designe la couleur des boules cassées
    if nombreDeCasse[1] == "rouge" or nombreDeCasse[1] == "jaune" or nombreDeCasse[1] == "violet" :
        #nombreDeCasse[0] designe le nombre de boules cassées
        if nombreDeCasse[0] >= 4 : # 4  est le nombre de casses nécessaires pour atttaquer
            self.mettreDegat(nombreDeCasse[0])
            self.compteurTour = 0 #Reinitialisation du compteur tour

    elif  nombreDeCasse[1] == "bleu" or nombreDeCasse[1] == "vert" :
        if nombreDeCasse[0] >= 4 :# 4  est le nombre de casses nécessaires pour atttaquer
            if self.vie >= self.vieMax:#Condition limitant la variable "vie" à 1000;
                self.vie += 0
            else:
                self.vie += self.degat*(ceil(nombreDeCasse[0]/2)-1)*1.75 #Formule permettant d'augmenter les soins selon le
                #nombre de boules cassées
            self.compteurTour = 0 #Reinitialisation du compteur tour

    if nombreDeCasse[0] < 4:
        self.compteurTour += 1 #Incrémentation du compteur
    if self.compteurTour == self.ennemi.nombreDeTour :
        self.vie -= self.degat*15/self.ennemi.nombreDeTour #Formule permettant d'augmenter les dégats des ennemis à chaque niveau
        self.compteurTour = 0 #Reinitialisation du compteur tour
```

Figure: Extrait code la fonction VieHeros de la class Heros

Fonctions du Héros

```
def mettreDegat(self ,nombreDeCasse):
```

```
def affichageHerosEnnemi(self):
```

```
def victoireDefaite(self):
    if self.vie <= self.vieMin :
        return "perdu" #Si la vie est plus basse que les points de vie minimums, perdu
    elif self.ennemi.vie <= self.vieMin:
        return "victoire" #Si la vie de l'ennemie est inférieur au pdv minimum, victoire
```

Figure: Extrait code fonctions de la class Heros

Initialisation de l'Ennemi

```
self.vie = vie
self.nombreDeTour = nombreDeTour #Nbr de tours avant une attaque
self.ennemivieMax = vie

#Importation de l'image de l'ennemi selon le niveau
if nombreDeTour == 6: #Nbr de tours = 6 signifie niveau 1
    self.image = pygame.image.load("Img/slime.png")
    self.imageDegat = pygame.image.load("Img/slimeDamage.png")

elif self.nombreDeTour == 5: #Nbr de tours = 5 signifie niveau 2
    self.image = pygame.image.load("Img/squelette.png")
    self.imageDegat = pygame.image.load("Img/squeletteDamage.png")
```

Figure: Extrait code init() de la class Ennemi

Experimentation

blabla sur l'expe

.

conclusion