Documentation SpaceInvaders EL OUAZIZI Walid

Sommaire

- Alien
- Bullet
- AlienBullet
- Defender
- Fleet
- Game
- Score
- SpaceInvaders

Alien

Notre classe Alien gère chaque Alien qui compose notre flotte
Les attributs de cette classe sont :

```
id -> l'id du sprite de l'alien
alive -> boolean determinant l'etat de notre Alien
life -> entier designant le nb de PV restant pour notre alien
fired_bullets -> tableau de AlienBullet correspondant aux balles tirées par notre
alien
```

Les méthodes de cette classe sont :

```
init(self, hp) -> Initialise l'Alien avec l'attribut life a la valeur de hp
touched_by(self, canvas, projectile) -> Renvoie True si l'Alien est touche par le
projectile, False sinon
install_in(self, canvas, x, y, image, tag) -> Installe notre alien sur le canvas
aux coords (x, y) avec image et tag
move_in(self, canvas, dx, dy) -> Deplace l'Alien de (dx, dy) sur le canvas
fire(self, canvas) -> Fais tirer l'Alien sur le canvas
```

Bullet

Notre classe Bullet gère chaque Bullet tirées par notre Defender
Les attributs de cette classe sont :

```
id -> L'id du sprite de la bullet
radius -> Int Le rayon de la bullet
color -> String La couleure de la Bullet
speed -> Int La vitesse de notre Bullet
shooter -> Defender Le sprite a l'origine de la Bullet
```

Les méthodes de cette classe sont :

```
init(self, shooter) -> Initialise la Bullet avec l'attribut shooter a la valeur de
shooter
install_in(self, canvas) -> Installe la Bullet sur le canvas aux coords du shooter
move_in(self, canvas) -> Deplace la Bullet de son attribut speed sur l'axe Y-
```

AlienBullet

Notre classe AlienBullet gère chaque Bullet tirées par nos Aliens
Les attributs de cette classe sont :

```
id -> L'id du sprite de la bullet
radius -> Int Le rayon de la bullet
color -> String La couleure de la Bullet
speed -> Int La vitesse de notre Bullet
shooter -> Alien Le sprite a l'origine de la Bullet
```

Les méthodes de cette classe sont :

```
init(self, shooter) -> Initialise la Bullet avec l'attribut shooter a la valeur de
shooter
install_in(self, canvas) -> Installe la Bullet sur le canvas aux coords du shooter
move_in(self, canvas) -> Deplace la Bullet de son attribut speed sur l'axe Y+
```

Defender

Notre classe Defender gère notre joueur le Defender
Les attributs de cette classse sont :

```
width -> Int La largeur du sprite
height -> Int La hauteur du sprite
move_delta -> Int La valeur de deplacement du sprite
id -> L'ID du sprite
max_fired_bullets -> Int Le nb max de balles tirèes
fired_bullets -> Tableau de Bullet tirées par le Defender
score -> Int score du Defender
pseudo -> String pseudo du Defender
life -> Int HP du Defender
game -> Game Instance de la Game du Defender
```

Les méthodes de cette classe sont :

```
init(self, game, life) -> Initialise le Defender avec game à game et life à life
install_in(self) -> Installe le sprite
move_in(self, dx) -> Deplace le sprite de dx
fire(self) -> Tire un Bullet
updateScore(self) -> Ecrit le score du joueur dans le fichier de score
touched_by_fleetshot(self) -> Verifie que l'utilisateur est touche par un tir
Alien, enleve un PV si oui
manageDeath(self) -> Met fin à la partie si life = 0
getScore(self) -> Renvoie le score courant
```

Fleet

Notre classe Fllet gère la flotte d'Alien
Ses attributs sont :

```
aliens_lines -> Int nb ligne d'Alien
aliens_columns -> Int nb colonnes d'Alien
aliens_inner_gap -> Int ecart entre chaque alien
alien_x_delta -> Int deplacement x de notre flotte
alien_y_delta -> Int deplacement y de notre flotte
alienImage -> TkImage gif de notre sprite d'Alien
explosionImage -> TkImage gif de notre sprite d'explosion
start_x -> Int x de demarrage du dessin de la Fleet
start_y -> Int y de demarrage du dessin de la Fleet
aliens_fleet -> Tableau d'Alien qui compose la Fleet
hp -> Int PV de chaque Alien de la Fleet
game -> Game l'instance de Game lancé
```

Ses méthodes sont :

```
init(self, game, hp) -> Initalise la flotte avec game à game et hp à hp
install_in(self) -> Installe la Fleet sur la frame dans le canvas
move_in(self) -> Deplace la Fleet et gere la perte de Defender si la FLeet descend
au Defender
manage_touched_aliens_by(self, defender) -> Gere les Aliens touches par defender
alien_shoot(self) -> Fias tirer un Alien au hasard
fleet_alive(self) -> Rnevoie True si la Fleet comporte au moins 1 Alien en vie,
False sinon
manageDeath(self) -> Gere la mort de chaque Alien qui n'a plus de vie
```

Game

Notre classe Game gere notre partie
Ses attributs sont :

```
mainClass -> SpaceInvaders La classe SpaceInvaders qui a fait appel a Game
pseudo -> String Pseudo courant
frame -> TKinterFrame La frame courante
fleet -> Fleet une nouvelle Fleet
defender -> Defender un nouveau Defender
height -> Int Hauteur du cnavas
width -> Int Largeur du canvas
canvas -> TKinterCanvas nouveau canvas
explosionList -> Tableau des id de sprite des explosions a supprimer
```

Ses méthodes sont :

```
init(self, mainClass, pseudo, ...) -> Initialise la partie avec la vie le pseudo
et la classe SpaceInvaders
start_animation(self) -> Demarre l'animation
animation(self) -> Effectue les op de l'animation puis boucle sur elle même
move_bullets(self) -> Deplace les bullets
move_aliens_fleet(self) -> Déplace la Fleet
check_win(self) -> Verifie si la partie est gagnée
alienShoot(self) -> Determine si on fais tirer un Alien ou non
removeAllExplosion(self) -> Supprime tout les sprites avec le tag explosion
removeExplosion(self, id) -> Gère la suppression des explosions une à une
getCanvas(self) -> Renvoie le canvas courant
getDefender(self) -> Renvoie le Defender courant
getFleet(self) -> Renvoie la Fleet courante
getMainClass(self) -> Renvoie la classe Principale
```

Score

Notre classe Score gère l'écriture et la lecture de Score
Elle n'a aucun attribut et sert uniquement a apporter 2 fonctiones a notre programme
Ses méthodes sont :

```
read() -> Renvoie le dictionnaire des score lu a travers le fichier de score
write() -> Ecrit un dictionnaire dans le fichier de score
```

SpaceInvaders

Notre classe Space Invaders est la classe principale du jeu
Elle gère le menu et la scene de jeu
Ses attributs sont :

```
root -> Contient la fenetre principale de classe Tk
score -> Contient la classe Score
frame -> Contient la frame a afficher (menu ou jeu)
```

```
game -> Instance de Game si le jeu est lancé, None sinon
pseudo -> Le pseudo du joueur
```

Ses méthodes sont :

```
init(self) -> Initalise notre classe principale
keypress(self, event) -> Gere nos touches (tir et deplacement)
play(self) -> Lance le mainloop de TKinter
startGame(self) -> Lance une game selon le pseudo et les HP du Defender/Alien
clearFrame(self) -> Nettoie completement notre frame
drawMenu(self) -> Dessine notre Menu dans l'attribut frame
getFrame(self) -> Renvoie la frame
```