ÉCOLE NATIONALE D'INGÉNIEURS DE BREST

RÉVÉLATEUR D'INGÉNIEURS
DEPUIS 1961

Cahier de conception _ Projet informatique S2

Type Nom du projet	Cahier de conception Robot Rumble
Commentaire	Jeu de tir
Auteur	Carrée Ewan S2P-B
Version	2.2
Date	07/06/2019

Table des matières

1 - Rappel du cahier des charges……3
1.1 Contraintes techniques……3
1.2 Fonctionnalités……3
1.3 Prototype P1……3
1.3.1 Archive……3
1.3.2 Manuel d'utilisation……3
1.4 Prototype P2……3
1.4.1 Archive ……3
1.4.2 Manuel d'utilisation……3

2 - Principes des solutions techniques adoptées……3
2.1 Langage……3
2.2 Architecture du logiciel……3
2.3 Interface utilisateur……3
2.3.1 Boucle de simulation……3
2.3.2 Affichage ……3
2.3.3 Gestion du clavier……3
2.3.4 Image ascii-art……3
2.4 Grille……4

3 - Analyse de conception……4
3.1 Analyse noms/verbes……4
3.2 Types de données……4
3.3 Dépendance entre modules……5
3.4 Analyse descendante……5
3.4.1 Arbre principal……5
3.4.2 Arbre affichage……6
3.4.3 Arbre déplacement……6
3.4.4 Arbre interaction……6

4 - Description des fonctions……6
4.1 Programme principale : Main.py ……7
4.2 Background.py……7
4.3 Animat.py……8
4.4 Robot.py……9
4.5 Shoot.py……10
4.6 Animation.py……10

5 - Calendrier et suivit de développement……11
5.1 P1……11
5.1.1 Fonctions à développer……11
5.1.2 Autre……11
5.2 P2……12
5.2.1 Fonctions à développer……12
5.2.2 Autre……12

1 - Rappel du cahier des charges

Contraintes techniques : Le jeu doit pouvoir être joué dans le terminal Linux. Il doit aussi fonctionner sur les ordinateurs de l'ENIB. Ce jeu doit être codé en langage python 2.7
Enfin, le jeu doit obligatoirement répondre à la condition générale imposée qui dit que le jeu doit être un jeu de tir. Il doit donc obligatoirement comporter une fonction qui permet de tirer.

Fonctionnalités :

F1 : jouer une partie
- F1.1 : afficher le jeu : grille, informations
- F1.2 : se déplacer sur la grille
- F1.3 : tirer
- F1.4 : présence d'un chronomètre
- F1.5 : présence et gestion d'obstacles dans la zone
- F1.6 : évolution du robot + capacités

F2 : finir une partie
- F2.1 : mention de fin de partie

Prototype P1
Archive :
Manuel d'utilisation :Manuel_utilisation.txt
Prototype P2
Archive :
Manuel d'utilisation : Manuel_utilisation.txt

2 - Principes des solutions techniques

Langage :
Conformément aux contraintes énoncées dans le cahier des charges, le codage est réalisé avec le langage python dont la version est 2.7.5
Architecture du logiciel :
Nous mettons en œuvre le principe de la barrière d'abstraction. Chaque module correspond à un type de donnée et fournit toutes les opérations permettant de le manipuler de manière abstraite
Interface utilisateur :
L'interface utilisateur se fera via un terminal de type linux. Nous reprenons la solution donnée en cours d'IPI en utilisant les modules : termios, sys et select.
Boucle de simulation :
Le programme mettra en œuvre une boucle de simulation qui gèrera l'affichage et les événements clavier.
Affichage :
L'affichage se fait en communicant directement avec le terminal en envoyant des chaînes de caractères sur la sortie standard de l'application.
Gestion du clavier :
L'entrée standard est utilisée pour détecter les actions de l'utilisateur. Le module tty permet de rediriger les événements clavier sur l'entrée standard. Pour connaître les actions de l'utilisateur il suffit de lire l'entrée standard
Image ascii-art :
Pour dessiner certaines parties de l'interface nous utilisons des « images ascii ». Dans l'idée de séparer le code et les données, les différentes images ASCII seront stockées dans des fichiers textes : Robot.txt, Background.txt, Animat.txt

Grille :
Pour modéliser le fond du jeu, nous optons pour une image ascii-art que nous découpons en case afin de pouvoir l'utiliser en tant que coordonnées sur un axe orthonormé.

3 - Analyse de conception

Analyse noms/verbes :

- Verbes : jouer, afficher, déplacer,finir,quitter,tirer,soigner
- Nom : joueur,fond,temps,robot,joker,tir,soin

Type de données

Type : Background= struct
    bg : dict
        map : list
    joker : dict
        color : int
        x : int
        y : int
        speed : int

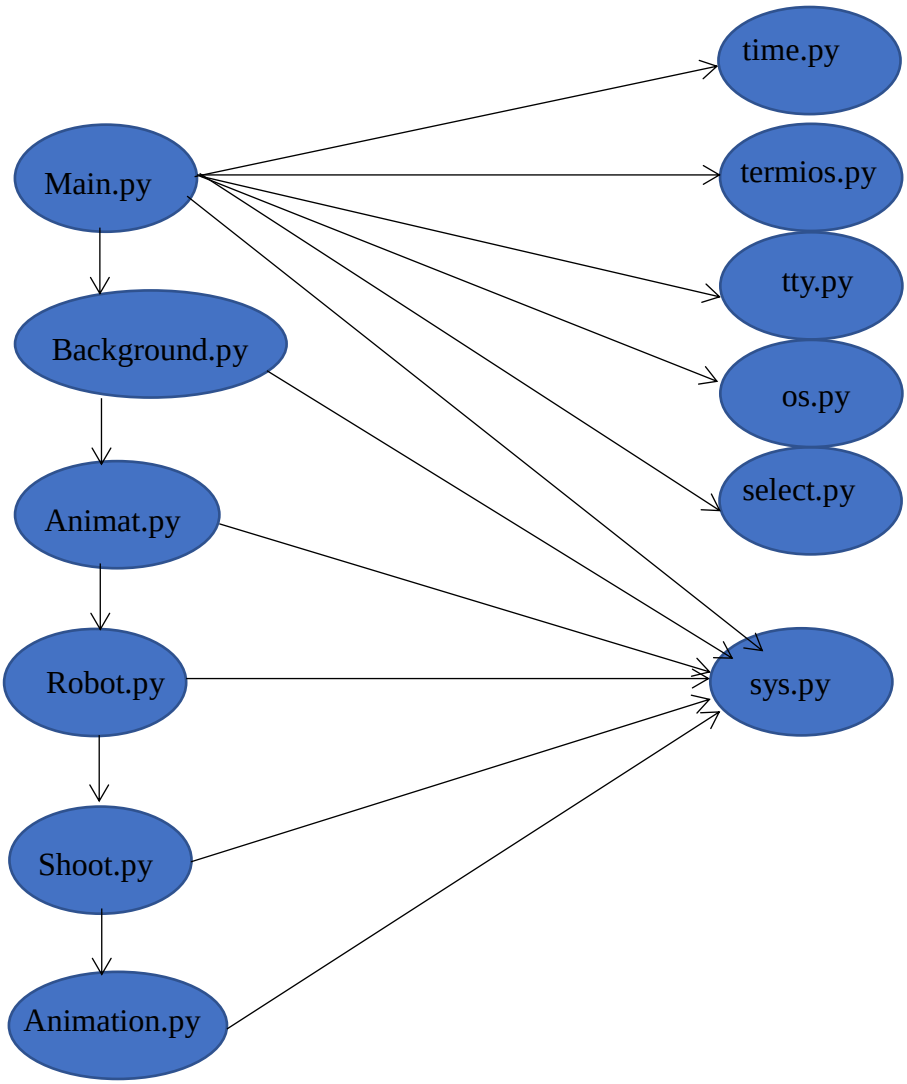
Type : Animat= struct
    animat : dict
        color : int
        x : int
        y : int
        speed : int
        life : int
        shape: list
        width:int
        high:int

Type : Robot= struct
    robot : dict
        color : int
        x : int
        y : int
        speed : int
        life : int
        shape: list
        width:int
        high:int

Type : Shoot= struct
    shot: dict
    x :int
    y :int
    color :int
    decalage :int
    etat :int
    tir :str

Type : Animation= struct
animation: dict
map : list

Dépendance entre modules :



Analyse descendante
Arbre principal :
Main.main()
Main.init()
animat = Animat.create()
background,joker= Background.create()
robot2=Robot2.create()
shot=[]
tirjoueur= Shoot.create()
shot.append(tirjoueur)
roquetterobot=Shoot.create()

```
            shot.append(roquetterobot)
            missilerobot=Shoot.create()
            shot.append(missilerobot)
            missile2robot=Shoot.create()
            shot.append(missilerobot2)
            animation=[]
            win =Animation.create()
            animation.append(win)
            lose =Animation.create()
            animation.append(lose)
            etatRobot= Animation.create()
            animation.append(etatRobot)
        Main.run()
            interact()
            move()
            show()
```

Arbre affichage :

```
Main.show()
    Background.show()
    chrono()
    Shoot.show()
    Robot2.show()
    Animat.show()
```

Arbre déplacement :

```
Main.move()
    Robot2.move()
    Shoot.move()
    Background.move()
```

Arbre interaction :

```
Main.interact()
    c = sys.stdin.read(1)
    if c == '\x1b':       # x1b is ESC
        quitGame()
    elif c== "z":
        Animat.moveH()
    elif c== "q":
        Animat.moveL()
    elif c== "s":
        Animat.moveD()
    elif c== "d":
        Animat.moveR()
    elif c=='6' or c=='0':
        cheat()
```

4 - Description des fonctions

Programme principal : Main.py

def init()
def move()
def interact()
def isData()
def show()
def chrono()
def cheat()
def run()
def end()
def quitGame()

Main.init() → rien
    Description: permet de créer les éléments qui composent le jeu
    Paramètres: aucun
    Valeur de retour : aucune
Main.move() → rien
    Description: permet de déplacer les éléments qui composent le jeu
    Paramètres: aucun
    Valeur de retour : aucune
Main.interact() → rien
    Description: permet d'interagir avec l'utilisateur par le billet d'entrées clavier
    Paramètres: aucun
    Valeur de retour : aucune
Main.isData() → rien
    Description: permet de récupérer les touches clavier que l'utilisateur sélectionne
    Paramètres: aucun
    Valeur de retour : select.select([sys.stdin], [], [], 0) == ([sys.stdin], [], [])
Main.show() → rien
    Description: permet d'afficher les éléments qui composent le jeu
    Paramètres: aucun
    Valeur de retour : aucune
Main.chrono() → int
    Description: fonction chronométrant l'avancée de la partie
    Paramètres: aucun
    Valeur de retour : finalTime
Main.cheat() → rien
    Description: fonction bonus permettant de tricher dans la partie
    Paramètres: aucun
    Valeur de retour : aucune
Main.run() → rien
    Description: fonction qui gère la boucle de simulation qui permet à l'utilisateur de jouer
    Paramètres: aucun
    Valeur de retour : aucune
Main.end()→rien
    Description: fonction qui affiche l'écran de fin de partie
    Paramètres: aucun
    Valeur de retour : aucune
Main.quitGame() → rien

Description: fonction permettant de quitter le jeu à tout moment de la partie
Paramètres: aucun
Valeur de retour : aucune

Background.py

def create(filename)
def show(b,j)
def move(j,a,bg)
def setPosition(e,x,y)
def getPosition(e)
def setColor(e,color)
def getChar()
def setChar()

Background.create(filename) → background
Description: permet de créer le fond du jeu
Paramètres: filename
Valeur de retour : bg
Background.show(bg) → rien
Description: permet d'afficher le fond du jeu
Paramètres: background
Valeur de retour : aucu ne
Background.move(j,a,bg) → rien
Description: permet au joker de se déplacer
Paramètres: j,a,bg
Valeur de retour : aucune
Background.setPosition(e,x,y) → rien
Description: change la position du joker
Paramètres: e,x,y
Valeur de retour : aucune
Background.getPosition(e) → tuple(entier,entier)
Description: renvoie la position du joker
Paramètres: e
Valeur de retour : position du joker
Background.setColor(e,color) → rien
Description: change la couleur du joker
Paramètres: e,color
Valeur de retour : aucune

Background.getChar(bg,x,y) → tuple(entier,entier)
Description: renvoie le contenu d'une case donnée
Paramètres: background,x,y
Valeur de retour : index de la case sélectionnée
Background.setChar(bg,x,y,c) → rien
Description: change le fond d'une case donnée
Paramètres: background,x,y,c
Valeur de retour : aucune

Animat.py

def create(x,y)
def show(a)
def move(a,bg,r2)
def setPosition(e,x,y)
def getPosition(e)
def setColor(e,color)

Animat.create() → animat
    Description: permet de créer le joueur
    Paramètres: aucun
    Valeur de retour : animat
Animat.show(a) → rien
    Description: permet d'afficher le joueur
    Paramètres: a
    Valeur de retour : aucune
Animat.moveH(a,r2,bg) → rien
    Description: permet de déplacer le joueur vers le haut
    Paramètres: a,r2,bg
    Valeur de retour : aucune
Animat.moveD(a,r2,bg) → rien
    Description: permet de déplacer le joueur vers le bas
    Paramètres: a,r2,bg
    Valeur de retour : aucune
Animat.moveL(a,r2,bg) → rien
    Description: permet de déplacer le joueur vers la gauche
    Paramètres: a,r2,bg
    Valeur de retour : aucune
Animat.moveR(a,r2,bg) → rien
    Description: permet de déplacer le joueur vers la droite
    Paramètres: a,r2,bg
    Valeur de retour : aucune
Animat.setPosition(e,x,y) → rien
    Description: change la position du joueur
    Paramètres: e,x,y
    Valeur de retour : aucune
Animat.getPosition(e) → tuple(entier,entier)
    Description: renvoie la position du joueur
    Paramètres: e
    Valeur de retour : position du joueur
Animat.setColor(e,color) → rien
    Description: change la couleur du joueur
    Paramètres: e,color
    Valeur de retour : aucune

Robot.py
def create()
def show(r)
def move(r,a,bg)
def setPosition(e,x,y)

def getPosition(e)
def setColor(e,color)

Robot.create() → robot
    Description: permet de créer le robot
    Paramètres: aucun
    Valeur de retour : robot
Robot.show(r) → rien
    Description: permet d'afficher le robot
    Paramètres: r
    Valeur de retour : aucune
Robot.move(r,a,bg) → rien
    Description: permet de déplacer le robot
    Paramètres: r,a,bg
    Valeur de retour : aucune
Robot.setPosition(e,x,y) → rien
    Description: change la position du robot
    Paramètres: e,x,y
    Valeur de retour : aucune
Robot.getPosition(e) → tuple(entier,entier)
    Description: renvoie la position du robot
    Paramètres: e
    Valeur de retour : position du robot/tir
Robot.setColor(e,color) → rien
    Description: change la couleur du robot
    Paramètres: e,color
    Valeur de retour : aucune

Shoot.py
def create(x,y,color,decalage,etat,tir)
def show(s)
def move(r2,bg,a,s,direction)
def setPosition(e,x,y)
def getPosition(e)
def setColor(e,color)

Shoot.create(x,y,color,decalage,etat,tir) → shot
    Description: permet de créer un tir
    Paramètres: x,y,color,decalage,etat,tir
    Valeur de retour : shot
Shoot.show(s) → rien
    Description: permet d'afficher un tir
    Paramètres: s
    Valeur de retour : aucune
Shoot.move(r2,bg,a,s,direction) → rien
    Description: permet de déplacer un tir
    Paramètres: r2,bg,a,s,direction
    Valeur de retour : aucune
Shoot.setPosition(e,x,y) → rien

Description: change la position d'un tir
Paramètres: e,x,y
Valeur de retour : aucune

Shoot.getPosition(e) → tuple(entier,entier)
Description: renvoie la position d'un tir
Paramètres: e
Valeur de retour : position d'un tir

Shoot.setColor(e,color) → rien
Description: change la couleur d'un tir
Paramètres: e,color
Valeur de retour : aucune

Animation.py

def create(filename)
def show(w,l,r2,a,e)

Animation.create() → animation
Description: permet de créer une animation
Paramètres: aucun
Valeur de retour : animation

Animation.showStart(animation)→str
Description: permet d'afficher l'écran de démarrage
Paramètres: animation
Valeur de retour : name

Animation.showEvent(animation,r2,a,finalTime) → rien
Description: permet d'afficher une animation en fonction des évènements de la partie
Paramètres: animation ,r2,a,finalTime
Valeur de retour : aucune

5 - Calendrier et suivi de développement

P1

Fonctions à développer :

Fonctions codées testées commentées
Main.py def init() def move() def interact() def isData() def show() def run() def quitGame()
Background.py def create(filename):background def show() : background def getChar(bg,x,y) def setChar(bg,x,y)
Animat.py def create(x,y)

def show(a)
def move(a,bg,r2)
def setPosition(e,x,y)
def getPosition(e)
def setColor(e,color)

Robot.py
def create()
def show(r)
def move(r,a)
def setPosition(e,x,y)
def getPosition(e)
def setColor(e,color)

Autre :
Background.txt, Robot.txt, Animat.txt

P2
Fonctions à développer :

Main.py
def chrono()
def cheat()

Background.py
def create(filename) : joker
def show() : joker
def move() : joker

Shoot.py
def create(x,y,color,decalage,etat,tir)
def show(s)
def move(r2,bg,a,s,direction)
def setPosition(e,x,y)
def getPosition(e)
def setColor(e,color)

Animation.py
def create(filename)
def showEvent(animation,r2,a,finalTime)

Autre :
Win.txt, Lose.txt

Version finale

Fonctions à développer:

Animation.py def showStart(animation)
La version finale se veut plus esthétique et plus optimisée.