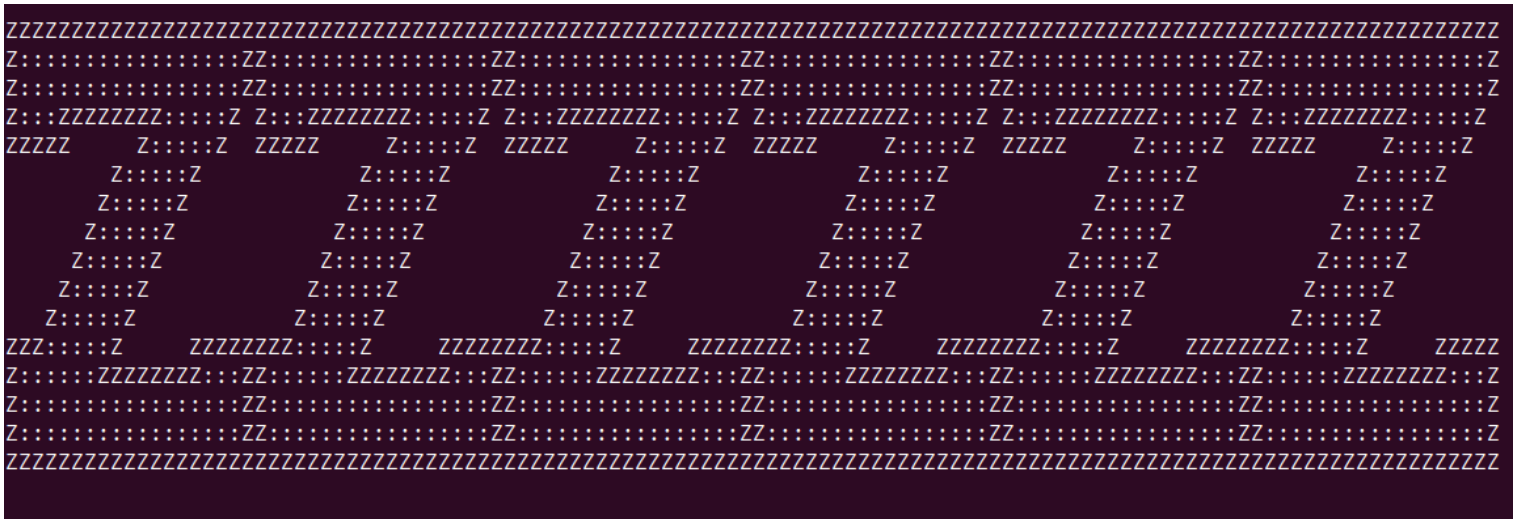
Conception



ZZZZZZ

Leydert Toine
ENIB - S2P

Contraintes techniques

- Le jeu doit pouvoir être lancer sur n'importe quel ordinateur comportant linux ainsi que python et de préférence être ceux de L'ENIB soit des ordinateurs standard tournant sur linux rocky
- Le langage utilisé en cours est Python. Le développement devra donc se faire en python.
- Le logiciel devra être réalisé en conformité avec les pratiques préconisées en cours d'IPI : barrière d'abstraction, modularité, unicode, etc …
- L'interface sera réalisée en mode texte dans un terminal

Fonctionnalités

- F1 : Commencer le jeu
  - F1.1 Afficher l'écran de début
  - F1.2 appuyer sur entrer pour commencer
- F2 : jouer au jeu
  - F2.1 Afficher le jeu
  - F2.2 déplacer le personnage dans un cadre, vers la gauche, droite, vers le haut ou le bas
  - F2.3 Interagir avec la clé
  - F2.4 changer d'écran à la sortie du cadre
- F3 : finir le jeu
  - F3.1 Afficher l'écran de fin de jeu
  - F3.2 Quitter

Langage

Conformément aux contraintes énoncées dans le cahier des charges, le codage est réalisé avec langage python. Nous choisissons la version 3.x du langage

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 de IPI en utilisant les séquence d'échappement ANSI et les modules :

`termios, sys, select.`

Boucle de simulation

Une boucle de une boucle de simulation gèrera l'affichage, les événements clavier et le déplacement de la balle.

Les calculs de physiques devront avoir une fréquence élevé pour améliorer la qualité de la gestion des collisions. L'affichage pourra avoir une fréquence de mise à jour plus faible pour pour ne pas surcharger le terminal.

Affichage

L'affichage se fait en communicant directement avec le terminal en envoyant des chaînes de caractères sur la sortie standard de l'application, en utilisant les séquences d'échappement ANSI.

Gestion du clavier

L'entrée standard est utilisé 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

Les niveaux du jeu seront faits en ascii de manière à ce qu'on puisse créer son propre niveau.

Dans l'idée de séparer le code et les données, une image ASCII représentant le jeu sera stockée dans le fichier texte : `niveau-xx.txt, les x remplacent des numéros.`

Analyse

Analyse noms/verbes :

- Verbes :

  Lancer, Afficher, Appuyer, Jouer, Finir, Se déplacer, Obtenir, Mourir, Changer, Ouvrir

- Nom :

  Personnage, Clef, Porte, Ennemi, Plein, Partiel, Pic

Types de donnée

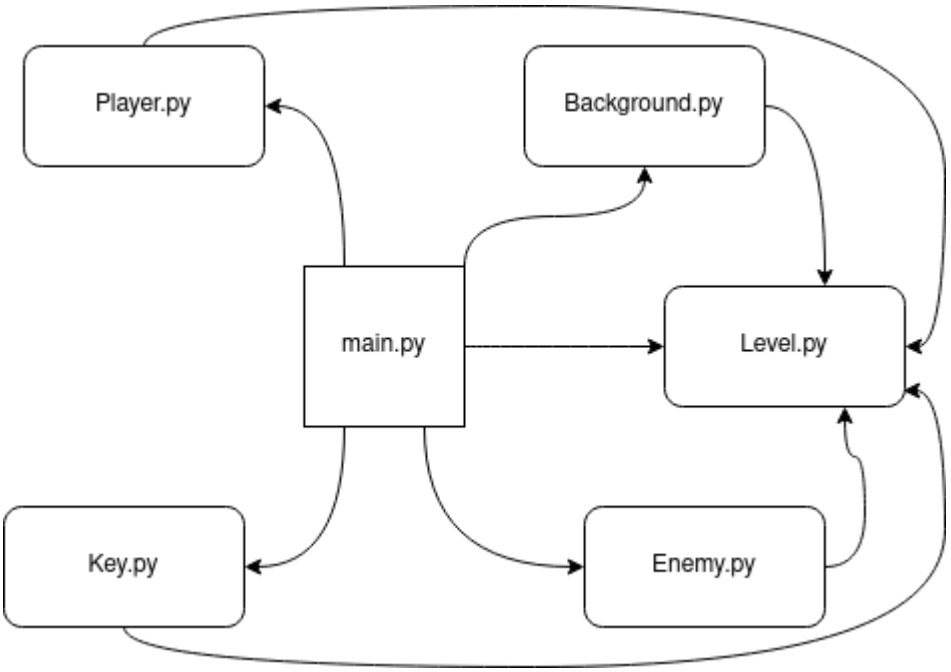
```
type : Background= struct
        fstruct
type : Player= struct
                couleur = réel
                x = entier
                y = entier
                vitesse = reel
        fstruct
type : Level = struct
                grille = liste


type Key = struct
                x = entier
                y = entier
        fstruct

type Enemy = struct
```

x = entier
y = entier
type = entier
state = entier
fstruct

Dépendance entre modules



Analyse descendante

Arbre principal

Main.main()
 +-- Main.init()
 | +-- Background.create()
 | +-- Level.create()
 | +-- Player.create()
 | +-- Enemy.create()
 | +-- Key.create()
 |
 +-- Main.run()
    +-- Main.interact()
    +-- Main.live()
    +-- Main.show()

Arbre interaction

Main.interact()
  +-- Main.is_data()
  +-- Player.move_left()
  +-- Player.move_right()
  +-- Player.gravity_change()
  +-- Player.pick_key()

Arbre simulation

Main.live()
 +-- Player.live()
 | +-- Player.collide()
 | +-- Player.test_collision()
 |
 +-- Enemy.live()
 | +-- Enemy.move()
 |
 +-- Key.live()
 |
 +-- Level.check_exit()
 |
 +-- Level.change()
 |
 +-- Main.game_over()
 +-- Main.win()

Arbre affichage

Main.show()
 +-- Background.show()
 +-- Level.show()
 +-- Player.show()
 +-- Enemy.show()
 +-- Key.show()

Description des fonctions (provisoire, pré-programmation)

Programme Principal: Main.py

data: dict {
  'timeStep': float,
  'show_period': float,
  'show_time': float,

'x_min': float,
'x_max': float,
'y_min': float,
'y_max': float,
'score': int,
'level': int,
'lives': int,
'levels': list Level,
'player': Player,
'enemy': Enemy,
'key': Key
}

- ➢ Main.main()
- ➢ Main.init()
- ➢ Main.run()
- ➢ Main.show()
- ➢ main.interact()
- ➢ Main.is_data()
- ➢ Main.live()
- ➢ Main.loose()
- ➢ Main.win()
- ➢ Main.quit_game()

- **Main.main()** -> rien
  Description: Fonction principale du jeu
  Paramètres: Rien
  Valeur de retour: Aucune

- **Main.init(data)** -> rien
  Description: Initialisation du jeu
  Paramètres: data
  Valeur de retour: Aucune

- **Main.run(data)** -> rien
  Description: Boucle de simulation
  Paramètres: data
  Valeur de retour: Aucune

- **Main.show(data)** -> rien
  Description: Fonction d'affichage du jeu
  Paramètres: data
  Valeur de retour: Aucune

- **Main.interact(data)** -> rien
  Description: Gère les événements clavier
  Paramètres: data
  Valeur de retour: Aucune

- **Main.is_data()** -> bool
  Description: Indique s'il y a des événements en attente
  Valeur de retour: True si oui, False sinon

- **Main.live(data)** -> rien
  Description: Simule l'évolution du jeu sur un pas de temps
  Paramètres: data
  Valeur de retour: Aucune

- **Main.game_over(data)** -> rien
  Description: Termine la partie si le joueur meurt
  Paramètres: data
  Valeur de retour: Aucune

- **Main.win(data)** -> rien
  Description: Termine la partie si le joueur atteint la sortie
  Paramètres: data
  Valeur de retour: Aucune

- **Main.quit_game(data)** -> rien
  Description: Quitte l'application
  Paramètres: data
  Valeur de retour: Aucune

Player.py

- ➢ Player.create(p)
- ➢ Player.set_pos(p)
- ➢ Player.get_pos(p)
- ➢ Player.move_left(p)
- ➢ Player.move_right(p)
- ➢ Player.gravity_change(p)
- ➢ Player.pick_key(p)
- ➢ Player.set_speed(p)

- **Player.create(x, y)** -> Player
  Description: Crée un joueur

Paramètres: x : entier, y : entier (coordonnées initiales)
Valeur de retour: Player

- **Player.set_pos(p,x, y)** -> tuple
  Description: définit la position du joueur
  Paramètres: x : entier, y : entier
  p : Player
  Valeur de retour: Aucun

- **Player.get_pos(p)** -> tuple
  Description: récupère la position du joueur
  Paramètres:
  p : Player
  Valeur de retour: (x : entier, y : entier) : tuple

- **Player.move_left(p)** -> rien
  Description: Déplace le joueur à gauche
  Paramètres:
  p : Player
  Valeur de retour: Aucune

- **Player.move_right(p)** -> rien
  Description: Déplace le joueur à droite
  Paramètres:
  p : Player
  Valeur de retour: Aucune

- **Player.gravity_change(p)** -> rien
  Description: Effectue un changement de gravité
  Paramètres:

p : Player
Valeur de retour: Aucune

- **Player.pick_key(p)** -> rien
  Description: Permet de ramasser la clé
  Paramètres:
  p : Player
  Valeur de retour: Aucune

- **Player.set_speed(p,v)** -> rien
  Description: défini la vitesse du joueur

Paramètres: v : float
p : Player
Valeur de retour: (x : entier, y : entier) : tuple

Enemy.py

- ➢ Enemy.create()
- ➢ Enemy.get_pos(e)
- ➢ Enemy.set_pos(e)
- ➢ Enemy.move(e)
- ➢ Enemy.test_player(e)
- ➢ Enemy.set_speed(e)

- **Enemy.create(x, y)** -> Enemy
  Description: Crée un ennemi
  Paramètres: x : entier, y : entier (coordonnées initiales)
  Valeur de retour: Enemy

- **Enemy.get_pos(e)** -> (x,y)
  Description: récupère les coordonnés
  Paramètres: Aucun
  e : Enemy
  Valeur de retour: (x : entier, y : entier) : tuple

- **Enemy.set_pos(e,x, y)** -> Enemy
  Description: positionne un ennemi
  Paramètres: x : entier, y : entier
  e : Enemy
  Valeur de retour: Enemy

- **Enemy.move(e)** -> rien
  Description: Déplace l'ennemi
  Paramètres:
  e : Enemy
  Valeur de retour: Aucune

- **Enemy.test_player(e)** -> bool
  Description: détecte le joueur dans une zone donné
  Paramètres:
  e : Enemy
  Valeur de retour:

- **Enemy.set_speed(e,v)** -> rien
  Description: défini la vitesse du joueur
  Paramètres: v : float
  e : Enemy
  Valeur de retour: aucune

Key.py

- Key.create()
- Key.get_pos(k)
- Key.set_pos(k)

- **Key.create(x, y)** -> Key
  Description: Crée une clé
  Paramètres: x : entier, y : entier (coordonnées)
  Valeur de retour: Key

- **Key.get_pos()** -> (x,y)
  Description: récupère les coordonnés
  Paramètres: Aucun
  k : Key
  Valeur de retour: (x : entier, y : entier) : tuple

- **Key.set_pos(x, y)** -> key
  Description: positionne une clé
  Paramètres: x : entier, y : entier
  k : Key
  Valeur de retour: Key

Level.py

- Level.create()
- Level.check_exit()
- Level.change()

- **Level.create(filename, offset)** -> Level
  Description: Charge un niveau
  Paramètres: fichier de niveau : chaîne de caractère, offset: entier
  Valeur de retour: Level

- **Level.check_exit(l,player)** -> bool
  Description: Vérifie si le joueur atteint la sortie
  Paramètres: player
  l : Level
  Valeur de retour: True si sorti, False sinon

- **Level.change(l,bool)** -> rien
  Description: Change de niveau si je joueur atteint la sortie
  Paramètres: booléen
  l : Level
  Valeur de retour: Aucune

Calendrier et suivi de développement

fonctions à développer

Fonctions	Date de programmation prévisionnelle	commentaire
main.py	semaine 17	
Main.main()	semaine 20	
Main.init()	semaine 20	
Main.run()	semaine 20	
Main.show()	semaine 20	
main.interact()	semaine 20	
Main.is_data()	semaine 20	
Main.live()	semaine 20	
Main.loose()	semaine 20	
Main.win()	semaine 20	
Main.quit_game()	semaine 20	
Player.create()	semaine 18	
Player.set_pos()	semaine 18	
Player.get_pos()	semaine 18	
Player.move_left	semaine 18	
Player.move_right()	semaine 18	
Player.gravity_change()	semaine 18	

Player.pick_key()	semaine 18	
Player.set_speed()	semaine 18	
Enemy.create()	semaine 19	
Enemy.get_pos()	semaine 19	
Enemy.set_pos()	semaine 19	
Enemy.move()	semaine 19	
Enemy.test_player()	semaine 19	
Enemy.set_speed()	semaine 19	
Key.create()	semaine 19	
Key.get_pos()	semaine 19	
Key.set_pos()	semaine 19	
Level.create()	semaine 19	
Level.check_exit()	semaine 19	
Level.change()	semaine 19	