ÉCOLE NATIONALE D'INGÉNIEURS DE BREST

RÉVÉLATEUR D'INGÉNIEURS
DEPUIS 1961

CONCEPTION

The Shades

Type	CONCEPTION
Nom du projet	The Shades
Auteurs	MAHIEUX Pierre (p3mahieu@enib.fr) PEARCE Valentin (v3pearce@enib.fr)
Version	1.0
Date	06/04/2014

Sommaire

1 Rappel du cahier des charges

1.1 Contraintes techniques

-Le logiciel doit fonctionner sur les machines de TP de l'ENIB pour que tout les élèves puissent le tester

-Le langage utilisé doit être le même que celui utilisé en cours. Nous utiliserons donc le langage Python

-Les notions de programmation orientée objet n'ayant pas été abordées en cours nous ne les utiliserons pas pour notre projet

-Le logiciel devra être développer en se basant sur les notions abordées en cours de MDD (barrière d'abstraction ; modularité ...)

-L'interface sera textuelle et affichée dans un terminal

1.2 Fonctionnalités

F1 : Nommer le joueur

F2 : Générer la carte

F3 : Jouer une partie

- F3.1 : Se déplacer
  - -nord
  - -sud
  - -est
  - -ouest
- F3.2 : Observer les alentours
- F3.3 : Ramasser de l'équipement
  - F3.3.1 : Ramasser des armes
  - F3.3.2 : Ramasser des consommables (soin)
- F3.4 : Se battre
- F3.5 : Perdre
  - F3.5.1 : Mourir
  - F3.5.2 : Limite de temps dépassée
- F3.6 : Gagner (sortir du quartier)

F4 : Colorer certaines parties du texte

1.3 P1 : Prototype 1

Ce prototype porte sur les fonctions de début et fin de partie, de génération de carte et de déplacement.

Mise en œuvre des fonctions F1, F2.1, F3.1, F3.2, F3.5.2, F3.6.

1.4 P2 : Prototype 2

Ce prototype ajoute les fonctions de gestion d'inventaire, de combat et de décès.

Ajout des fonctions F3.3, F3.4, F3.5.1.

Ajout éventuel des fonctions non-nécessaires.

2 Principes des solutions techniques

2.1 Langage

Conformément aux contraintes exposées dans le cahier des charges nous utiliserons le langage python dans sa version 2.7.5

2.2 Architecture du logiciel

Nous mettons en œuvre le principe de la barrière d'abstraction. Chaque module traite un type de donnée et fournit toutes les opérations nécessaires a la manipulation abstraite de ses données.

2.3 Interface utilisateur

L'interface utilisateur se fera par le biais d'un terminal type Linux.

Nous utiliserons les modules *termios, sys, select*.

2.3.1 Boucle de simulation

Notre programme mettra en œuvre une boucle de simulation qui gérera la durée de la partie et les événements du clavier.

2.3.2 Affichage

L'affichage se fait par l'intermédiaire du terminal en envoyant le texte directement sur la sortie standard.

2.3.3 Gestion du clavier

Toutes les actions se font par l'intermédiaire de raccourcis clavier. En utilisant le module *tty* on peut rediriger les événements clavier vers l'entrée standard.

2.3.4 Image ascii-art

On utilisera 3 images ascii : *background.txt, victoire.txt, defaite.txt*. Correspondant respectivement au fond d'écran de jeu, message de victoire, message de défaite.

2.4 Gestion de la carte

2.4.1 Récupération de cartes préconçues

Dans un premier temps nous "fabriquerons" des cartes que le jeu ira récupérer aléatoirement.

2.4.2 Génération aléatoire des cartes

Nous proposons d'ajouter une fonction de génération aléatoire de carte à notre programme. Celle-ci générera un labyrinthe à travers le tableau de la zone de jeu.

2.5 Gestion de l'inventaire

A chaque objet du jeu correspond un numéro qui sera recensé dans un dictionnaire. Lors d'un combat le jeu lira quel objet est équipé dans l'inventaire. Dans notre base de données chaque objet est associé à des caractéristiques.

3 Analyse

3.1 Analyse noms/verbes

```
Noms :

    joueur, carte, partie, alentours, équipement, armes,
consommables, temps, limite de temps, texte

Verbes :

    nommer, générer, récupérer, jouer, se déplacer, observer,
ramasser, se battre, perdre, mourir, dépasser, gagner, colorer
```

3.3 Types de Données

type : Game = struct

    player : Player

    map : Map

    items : Items

type : Player = struct

    name : chaîne

    health : entier

power : entier

spentTime : entier

equiped : entier

inventory : liste d'entiers

coordinates : tuple

type : Map = liste de listes

(coord_x,coord_y) : Area

type : Area = struct

items : liste d'entiers

monster : entier

north : entier

south : entier

east : entier

west : entier

type : Items = liste

element = Item

type : Item = struct

name : chaîne de caractères

descript : chaîne de caractères

type : booléen

modifier : réel (entier relatif)

type : Monsters = liste

element = Monster

type : Monster = struct

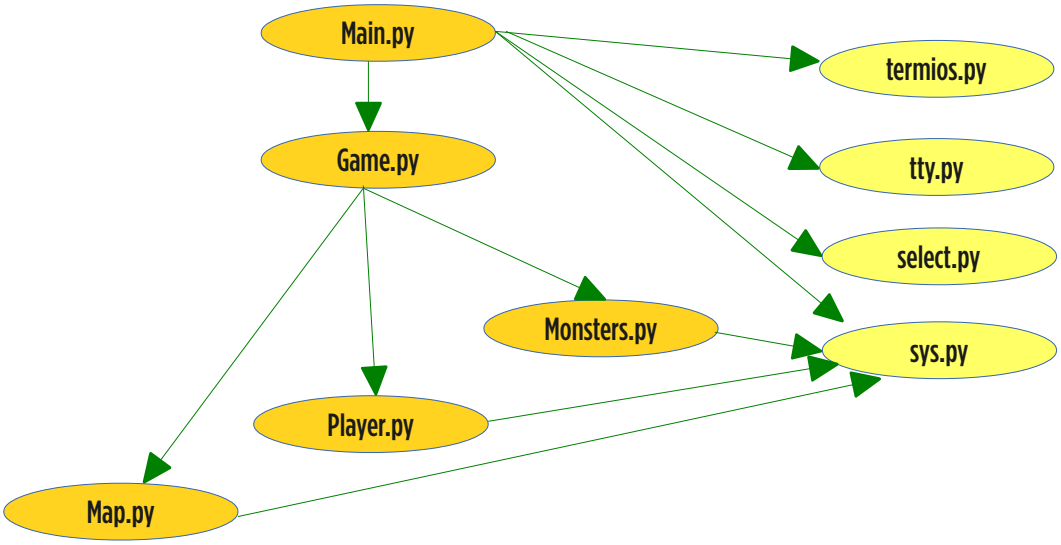
name : chaîne de caractères

descript : chaîne de caractères

health : entier

power : entier

3.3 Dépendance entre Modules



3.4 Analyse Descendante

3.4.1 Arbre Principal

```
Main.main()
+---Main.init()
|    +---Game.init()
|    |    +---Player.setName()
|    |    +---Map.generate()
|    |        +---Map.createEmpty()
|    |        +---Map.createMaze()
|    |        +---Map.addItem()
|    +---Game.descript()
|        +---Player.getPosition()
|        +---Map.getDescript(position)
|             +---Item.getDescript(index)
+---Main.run()
     +---Game.checkHealth()
     |    +---Player.getHealth()
     |    +---Game.lose()
     +---Game.checkTime()
```

```
    |   +---Game.lose()
    +---Main.display()
    +---Main.interact()
```

3.4.2 Arbre d'Affichage

```
Main.display()
+---Game.display(description)
```

3.4.3 Arbre d'Interaction

```
Main.interact()
+---Game.getAction()
    +---Game.move(direction)
    |   +---Player.getPosition()
    |   +---Map.check(position,direction)
    |   +---Player.move(direction)
    |   +---Game.win()
    +---Game.descript()
    |   +---Player.getPosition()
    |   +---Map.getDescript(position)
    +---Game.altDescript()
    |   +---Player.getPosition()
    |   +---Map.getAltDescript(position)
    +---Game.inventory()
    |   +---Player.getInventory()
    +---Game.fight()
    |   +---Player.getPosition()
    |   +---Map.getMonster(position)
    |   +---Player.getPower()
    |   +---Monster.getPower()
    |   +---Player.healthEdit(modifier)
    |   +---Monster.healthEdit(modifier)
    |   +---Map.removeMonster()
```

```
+---Game.use()
|   +---Player.healthEdit(modifier)
|   +---Player.equip(index)
+---Game.take()
|   +---Map.removeItem()
|   +---Player.addItem()
+---Game.drop()
    +---Map.addItem()
    +---Player.removeItem()
```

4 Description des Fonctions

4.1 Programme Principal : Main.py

- Main.main() → rien

  Description : Fonction Principale du programme

  Paramètres : Aucun

  Valeur de Retour : Rien

- Main.init()→ rien

  Description : Initialise le programme

  Paramètres : Aucun

  Valeur de Retour : Rien

- Main.run()→ rien

  Description : Boucle de simulation

  Paramètres : Aucun

  Valeur de Retour : Rien

- Main.display()→ rien

  Description : Affichage du jeu

  Paramètres : Aucun

  Valeur de Retour : Rien

- Main.interact()→ rien

  Description : Gère les événements clavier

Paramètres : Aucun

Valeur de Retour : Rien

4.2 Player.py

- Player.setName()→ rien

  Description : Nomme le joueur

  Paramètres : Aucun

  Valeur de Retour : Rien

- Player.healthEdit(modifier)→ rien

  Description : Modifie la vie du joueur

  Paramètres :

  modifier : reel

  Valeur de Retour : Rien

- Player.equip(index)→ rien

  Description :Équipe un objet au joueur

  Paramètres :

  index : entier

  Valeur de Retour : Rien

- Player.getHealth()→ entier

  Description : Récupère la vie du joueur

  Paramètres : Aucun

  Valeur de Retour : Vie du joueur

- Player.getPosition()→ tuple

  Description : Récupère la position du joueur

  Paramètres : Aucun

  Valeur de Retour : Coordonnées

- Player.getTime()→ entier

  Description : Récupère le temps passé par le joueur

  Paramètres : Aucun

  Valeur de Retour : Temps passé relatif aux actions du joueur

- Player.getInventory()→ liste

  Description : Récupère l'inventaire

  Paramètres : Aucun

  Valeur de Retour : Liste des valeur d'index des objets de l'inventaire du joueur

- Player.getPower()→ entier

  Description : Récupère la force du joueur

  Paramètres : Aucun

  Valeur de Retour : Force du joueur
- Player.move(direction)→ rien

  Description : Change les coordonnées du joueur

  Paramètres :

  direction : entier

  Valeur de Retour : Rien
- Player.removeItem(index)→ rien

  Description : Retire un objet de l'inventaire du joueur

  Paramètres :

  index : entier

  Valeur de Retour :Rien
- Player.addItem(index)→ rien

  Description : Ajoute un objet à l'inventaire du joueur

  Paramètres :

  index : entier

  Valeur de Retour : Rien

4.3 Map.py

- Map.generate()→ rien

  Description : Lance la génération de la carte

  Paramètres : Aucun

  Valeur de Retour : Rien
- Map.createEmpty()→ rien

  Description : Crée un carte vierge

  Paramètres : Aucun

  Valeur de Retour : Rien
- Map.createMaze()→ rien

  Description : Crée un labyrinthe dans la carte vierge

  Paramètres : Aucun

  Valeur de Retour : Rien

- Map.check(position,direction)→entier

  Description : Vérifie la possibilité de passage

  Paramètres :

  position : tuple

  direction : entier

  Valeur de Retour : 0 si impossible, 1 si possible, 2 si sortie et victoire

- Map.getDescript(position)→ chaîne de caractères

  Description : Récupère la description de base de la zone

  Paramètres :

  position : tuple

  Valeur de Retour : Description de base de la zone

- Map.getAltDescript(position)→ chaîne de caractères

  Description : Récupère une description poussée de la zone (présence d'objets)

  Paramètres :

  position : tuple

  Valeur de Retour : Description poussée de la zone

- Map.getMonster(position)→ entier

  Description : Vérifie la présence d'un monstre dans la zone

  Paramètres :

  position : tuple

  Valeur de Retour : Valeur d'index du monstre

- Map.removeMonster(position)→ rien

  Description : Retire le monstre de la zone

  Paramètres :

  position : tuple

  Valeur de Retour : Rien

- Map.removeItem(index,position)→ rien

  Description : Retire unobjet de la zone

  Paramètres :

  index : Valeur d'index de l'objet

  position : tuple

  Valeur de Retour :

- Map.addItem(index,position)→ rien

    Description : Ajoute un objet à la zone

    Paramètres :

        index : Valeur d'index de l'objet

        position : tuple

    Valeur de Retour : Rien

4.4 Game.py

- Game.checkHealth()→ rien

    Description : Vérifie si le joueur est toujours en vie

    Paramètres : Aucun

    Valeur de Retour : Rien

- Game.init()→ rien

    Description : Lance l'initialisation des varibles uniques à la partie

    Paramètres : Aucun

    Valeur de Retour : Rien

- Game.lose()→ rien

    Description : Fonction de fin par défaite, affiche le message de défaite

    Paramètres : Aucun

    Valeur de Retour : Rien

- Game.win()→ rien

    Description : Fonction de fin par victoire, affiche le message de victoire

    Paramètres : Aucun

    Valeur de Retour : Rien

- Game.inventory()→ chaîne de caractères

    Description : Lance la récupération d'inventaire

    Paramètres : Aucun

    Valeur de Retour : Texte pour l'affichage

- Game.display(description)→ rien

    Description : Affiche le texte corresponndant à la denière action faite

    Paramètres :

        description : chaîne

    Valeur de Retour : Rien

- Game.getAction()→ chaîne de caractères

  Description : Attend une frappe du clavier pour choisir l'action correspondante

  Paramètres : Aucun

  Valeur de Retour : Texte correspondant à l'action effectuée

- Game.move(direction)→ chaîne de caractères

  Description : Tente de déplacer le joueur

  Paramètres :

  direction : entier

  Valeur de Retour : Description de la nouvelle zone ou message d'erreur

- Game.descript()→ chaîne de caractères

  Description : Renvoie la description de la zone à l'afficheur

  Paramètres : Aucun

  Valeur de Retour : Description de la zone

- Game.altDescript()→ chaîne de caractères

  Description : Renvoie la description poussée de la zone à l'afficheur

  Paramètres : Aucun

  Valeur de Retour : Description poussée de la zone

- Game.fight()→ chaîne de caractères

  Description : Lance un attaque sur un monstre et calcule le vainqueur

  Paramètres : Aucun

  Valeur de Retour : Description de l'attaque

- Game.use()→ chaîne de caractères

  Description : Propose d'utiliser un objet

  Paramètres : Aucun

  Valeur de Retour : Desription de l'effet obtenu

- Game.take()→ chaîne de caractères

  Description : Propose de ramasser un objet

  Paramètres : Aucun

  Valeur de Retour : Descritpion de l'action

- Game.drop()→ chaîne de caractères

    Description : Propose de se débarraser d'un objet

    Paramètres : Aucun

    Valeur de Retour : Description de l'action

4.5 Monsters.py

- Monsters.getPower(index)→ entier

    Description : Récupère la valeur de force d'un monstre

    Paramètres :

        index : entier

    Valeur de Retour : Valeur de force du monstre

- Monsters.healthEdit(modifier)→ rien

    Description : Modifie la vie du monstre

    Paramètres :

        modifier : reel

    Valeur de Retour : Rien

4.6 Items.py

- Items.getDescript(index)→ chaîne de caractères

    Description : Récupère la description d'un objet

    Paramètres :

        index : entier

    Valeur de Retour : Description de l'objet

- Items.getName(index)→ chaîne de caractères

    Description : Récupère le nom d'un objet

    Paramètres :

        index : entier

    Valeur de Retour : Nom de l'objet

- Items.getModifier(index)→ entier

    Description : Récupère la valeur d'effet d'un objet

    Paramètres :

        index : entier

    Valeur de Retour : Valeur d'effet de l'objet

5 Calendrier et suivi de développement

5.1 P1

5.1.1 Fonctions à développer

Fontion	Codée	Vérifiée	Commentaires
Main.main()			
Main.init()			
Main.run()			
Main.display()			
Main.interact()			
Player.setName()			
Player.healthEdit(modifier)			
Player.getHealth()			
Player.getPosition()			
Player.getTime()			
Player.move(direction)			
Map.generate()			
Map.createEmpty()			
Map.createMaze()			
Map.check(position,direction)			
Map.getDescript(position)			
Map.getAltDescript(position)			
Game.checkHealth()			
Game.init()			

Game.lose()
Game.win()
Game.display(description)
Game.getAction()
Game.move(direction)
Game.descript()
Game.altDescript()

5.1.2 Autres

background.txt, victoire.txt, defaite.txt

5.2 P2

Fonction	Codée	Vérifiée	Commentaires
Player.equip(index)			
Player.getInventory()			
Player.getPower()			
Player.removeItem(index)			
Player.addItem(index)			
Map.getMonster(position)			
Map.removeMonster(position)			
Map.removeItem(index,positio n)			
Map.addItem(index,position)			
Game.inventory()			
Game.fight()			
Game.use()			
Game.take()			
Game.drop()			

Monsters.getPower(index)
Monsters.healthEdit(modifier)
Items.getDescript(index)
Items.getName(index)
Items.getModifier(index)