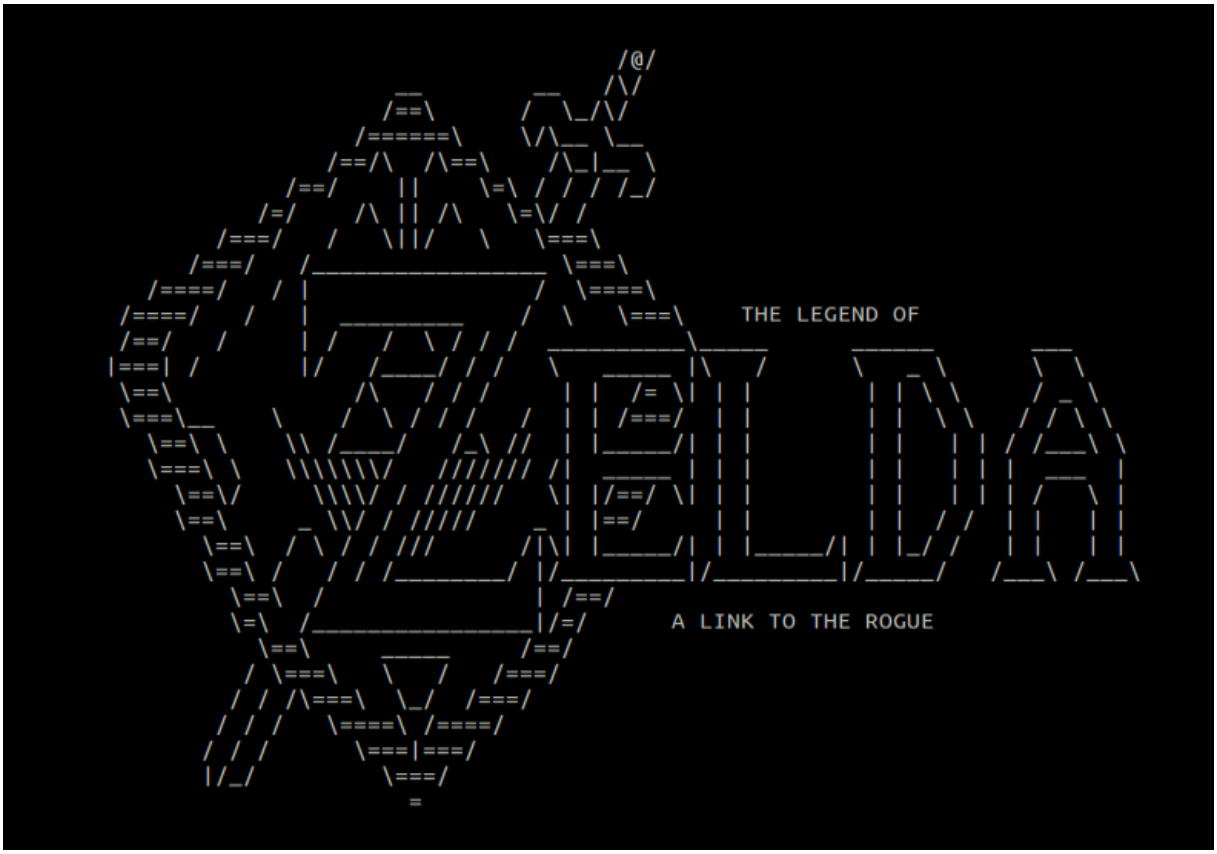
CONCEPTION



Type	Conception
Nom du projet	The Legends of Zelda - A Link to the Rogue
Commentaire	Projet cours de MDD, S2, ENIB
Auteur	LEVEQUE Dorian & ROUE Evan
Version	Prototype 1
Date	20/05/2016

Table des matières

Table des matières

1. Rappel du cahier des charges

1.1. Contraintes techniques

- Le Logiciel est associé à un cours, il doit fonctionner sur les machine de TP de l'ENIB pour que les élèves puissent le tester.
- Le langage utilisé en cours est Python. Le développement devra être donc se faire en python.
- Les notions de programmation orientée objet n'ayant pas encore été abordées, le programme devra essentiellement s'appuyer sur le paradigme de la programmation procédurale.
- Le logiciel devra être réalisé en conformité avec les pratiques préconisées en cours de MDD : barrière d'abstraction, modularité, unicode, etc…
- L'interface sera réalisée en mode texte dans un terminal

1.2. Fonctionnalités

- F1 : Faire un menu fonctionnel, interactif avec une interface simple et claire
  - F1.1 : Jouer une partie
    - F1.1.1 : Jouer une manche
      - F1.1.1.1 : Voir le Donjon :
        - Salle du donjon
        - Personnage
        - Score
        - Monstre
        - Carte
        - Coffre
        - Voir le menu pause
      - F1.1.1.2 : Se déplacer dans le donjon
      - F1.1.1.3 : Attaquer des monstres
      - F1.1.1.4 : Ouvrir des coffres avec des loots variés
      - F1.1.1.5 : Vaincre le Boss
      - F1.1.1.6 : Générer le donjon
    - F1.1.2 : Finir la partie :
      - F1.1.2.1 : Afficher le résultat et les caractéristiques du joueur
      - F1.1.2.2 : Afficher le menu
  - F1.2 : Pouvoir assigner les différentes actions du personnage aux touches du clavier de notre choix

1.3. P1 : Prototype P1

Ce prototype porte essentiellement sur la création des donjons et de l'affichage.
Mise en œuvre des fonctionnalités : F1.1, F1.1.1, F1.1.1.1, F1.1.1.2, F1.1.1.6.
Livré dans une archive au format .zip,.tgz ou .xz.
Contient un manuel d'utilisation dans le fichier readme.txt.

1.4. P2 : Prototype P2

Ce prototype réalise toutes les fonctionnalités obligatoires.
Ajout à P1 des fonctionnalités : F1, F1.1.1.3, F1.1.1.4, F1.1.1.5, F1.1.2, F1.1.2.2
Livré dans une archive au format .zip,.tgz ou .xz.
Contient un manuel d'utilisation dans le fichier readme.txt.

2. Principes des solutions techniques adoptées

2.1. Langage

Conformément aux contraintes énoncées dans le cahier des charges, le codage est réalisé avec le langage python. Nous choisissons la version 2.7.5.

2.2. Architecture du logiciel

Nous mettons en œuvre le principe de la barrière d'abstraction. Chaque module correspond à un type de donné et fournit toutes les opérations permettant de le manipuler de manière abstraite.

2.3. Interface utilisateur

L'interface utilisateur se fera via un terminal de type Linux. Nous reprenons la solution donnée en cours de MDD en utilisant les modules : termios, sys, select.

2.3.1. Boucle de simulation

Le programme mettra en œuvre une boucle de simulation qui gérera l'affichage du menu et du jeu, les déplacements des monstres, et du personnage, ainsi que les évènements clavier.

2.3.2. Affichage

L'affichage se fera en communicant directement avec le terminal en envoyant des chaînes de caractères sur la sortie standard de l'application via notre module « Utils.py ».

2.3.3. Gestion du clavier

L'entrée standard est utilisée pour détecter les actions de l'utilisateur. Le module tty permet de rediriger les évènements clavier sur l'entrée standard. Pour connaitre les actions de l'utilisateur, il suffit de lire sur l'entrée standard.

2.3.4. Image ascii-art

Pour nous faciliter la tâche sur la partie de l'interface, nous utiliserons des « images ascii ». Dans le but de séparer le code et les données, les différentes images ASCII seront stockées dans des fichiers xml rangés dans le répertoire « assets ».

Ces images nous permettront de dessiner dans le terminal, d'abord l'interface de l'utilisateur en jeu, les différentes fenêtres du menu, ainsi que les différentes salles qui constitueront le donjon lors de la génération aléatoire.

2.4. Génération aléatoire des donjons

Pour s'inspirer des jeux tel que « Rogue Legacy » où le donjon à la particularité de ne plus ressembler au donjon que l'on vient de faire suite à une mort. En réalité ce sont les pièces du donjon qui bouge, le donjon se régénère en entier au début de chaque nouvelle session.

Pour pouvoir réaliser cela, nous avons décidé d'utiliser plusieurs images ASCII au format .xml qui représenterons chacune une salle du donjon. Lors du lancement de la session, une

fonction permettra de générer le donjon en utilisant aléatoirement un fichier .xml contenant la salle.

2.5. Les armes :

Dans ces types de jeu, de nombreux items sont proposés et ont chacun leurs propres caractéristiques. Ils sont obtenable que dans des coffres ou en tuant des monstres. Seulement, nous n'avons pas le temps de coder différentes armes, ainsi celle-ci seront remplacer par des capacités que le joueur acquerra à chaque ouverture de coffre, ou monstres tués.

3. Analyse de conception

3.1. Analyse noms/verbes

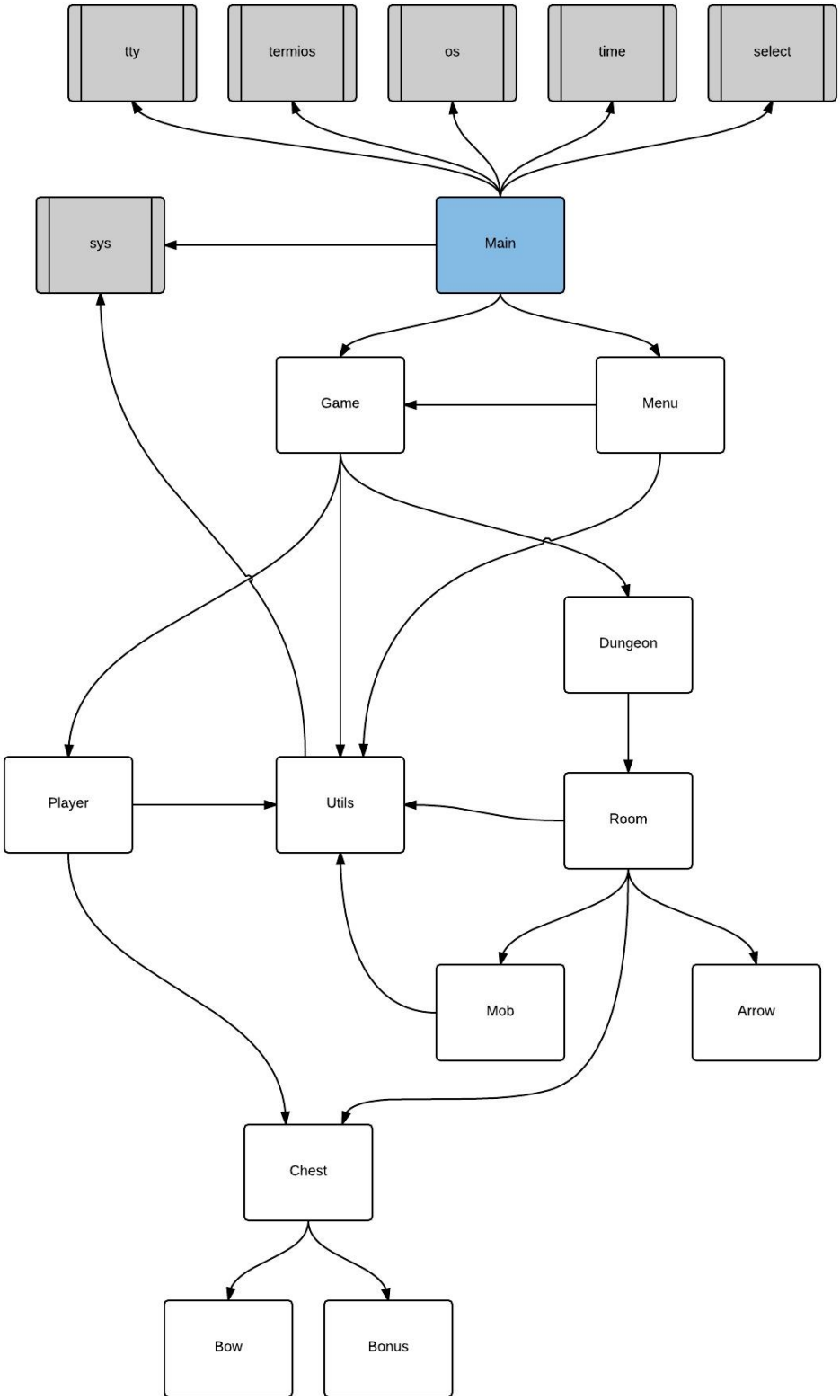
- Verbes :

  Afficher, sélectionner, jouer, générer, déplacer, attaquer, finir, recommencer ou quitter

- Nom :

  Joueur, salle, score, caractéristique, partie, résultat

3.2. Dépendance entre modules



3.3. Analyse descendante

3.3.1. Arbre principal

```
Main.main()
+--     Main.init()
| +--   Menu.create()
| +--   Game.create()
|       +--   Dungeon.create()
|       |     +--   Room.create()
|       +--   Player.create()
|             +--   Chest.create()
|
+--     Main.run()
  +--   Main.show()
  +--   Main.interact()
```

3.3.2. Arbre affichage

```
Main.show()
+--     Menu.show()
+--     Game.show()
  +--   Dungeon.show()
  |     +--   Room.show()
  |           +--   Chest.show()
  |           +--   Mob.show()
  |           +--   Arrow.show()
  +--   Player.show()
```

3.3.3. Arbre interaction

```
Main.interact()
+--   Menu.interact()
|     +--   Menu.getCurrentWindow()
|     +--   Menu.setCurrentWindow()
|     +--   Game.create()
|
+--   Game.interact()
|     +--   Game.move()
|     |     +--   Player.move()
|     |     |     +--   Player.getPosition()
|     |     |     +--   Player.setPosition()
|     |     +--   Dungeon.getCurrentRoom()
|     |           +--   Room.getMobByPosition()
|     |           +--   Mob.move()
|     |                 +--   Mob.getPosition()
|     |                 +--   Mob.setPostion()
|     +--   Game.switchRoom()
|           +--   Player.getPosition()
|           +--   Dungeon.getCurrentRoom()
|           +--   Dungeon.setCurrentRoom()
|           +--   Player.setPosition()
+--   Main.quit()
```

4. Description des fonctions

4.1. Programme Principal : Main.py

- `Main.`**`main()`**
- `Main.`**`init()`**
- `Main.`**`run()`**
- `Main.`**`show()`**
- `Main.`**`interact()`**
- `Main.`**`quit()`**

`Main.`**`main()`** → `rien`

Description : fonction principale du jeu
Paramètres : aucun
Valeur de retour : aucune

`Main.`**`init()`** → `rien`

Description : initialisation du jeu
Paramètres : aucun
Valeur de retour : aucune

`Main.`**`run()`** → `rien`

Description : boucle de simulation
Paramètres : aucun
Valeur de retour : aucune

`Main.`**`show()`** → `rien`

Description : fonction principale du jeu pour l'affichage
Paramètres : aucun
Valeur de retour : aucune

`Main.`**`interact()`** → `rien`

Description : gère les évènements clavier
Paramètres : aucun
Valeur de retour : aucune

`Main.`**`quit()`** → `rien`

Description : permet de quitter le jeu
Paramètres : aucun
Valeur de retour : aucune

4.2. Game.py

- `Game.create()`
- `Game.show(g)`
- `Game.interact(g, key)`
- `Game.switchRoom(g)`

`Game.create()` → `Game`

Description : permet de créer une nouvelle session de jeu

Paramètres : aucun

Valeur de retour : dictionnaire game

`Game.show(g)` → `rien`

Description : affiche l'interface, le joueur, les mobs, et le donjon

Paramètres :

g : Game

Valeur de retour : aucune

`Game.interact(g, key)`

Description : gère et affecte à chaque interaction clavier, une action de déplacement au joueur en fonction des obstacles présents dans la pièce

Paramètres :

g : Game

key : chaine de caractère

Valeur de retour : aucune

`Game.switchRoom(g)`

Description : Détecte lorsque le joueur est sur le bord de la fenêtre de jeu, et change la salle à afficher

Paramètres :

g : Game

Valeur de retour : aucune

4.3. Player.py

- player.**create()**
- player.**getPosition(p)**
- player.**setPosition(p, x, y)**
- player.**move(p, direction)**
- player.**show(p)**
- player.**getHealth(p)**
- player.**setHealth(p, health)**
- player.**getXp(p)**
- player.**setXp(p, xp)**
- player.**getStrength(p)**
- player.**setStrength(p, strength)**
- player.**getResistance(p)**
- player.**setResistance(p, resistance)**
- player.**getDamage(p)**
- player.**setDamage(p, damage)**
- player.**getSelectedBow(p)**
- player.**getInventory(p)**
- **player.getSprite(p)**
- **player.setSprite(p, sprite)**

player.**create()** → Player

Description : initialiser toutes les variables du joueur

Paramètre : rien

Valeur de retour : variables du joueur réinitialisé

player.**getPosition(p)**

Description : renvoie la position du joueur

Paramètre : p : Player

Valeur de retour : tuple (x, y) avec x : entier et y : entier

player.**setPosition(p, x, y)**

Description : définit la position du joueur

Paramètres :

p : Player

x : entier

y : entier

Valeur de retour : rien

player.**move(p, direction)**

Description : incrémente la position du joueur

Paramètres :

p : Player

direction : chaine de caractère

Valeur de retour : rien

`player.show(p)`

Description : affiche le joueur

Paramètres :

p : Player

Valeur de retour : rien

`player.getHealth(p)`

Description : renvoie le nombre de point de vie du joueur

Paramètre : p : Player

Valeur de retour : nombre de point

`player.setHealth(p, health)`

Description : définit le nombre de point de vie du joueur

Paramètres :

p : Player

health : entier

Valeur de retour : rien

`player.getXp(p)`

Description : renvoie le nombre de point d'expérience du joueur

Paramètre : p : Player

Valeur de retour : nombre de point d'expérience

`player.setXp(p, xp)`

Description : définit le nombre de point d'expérience du joueur

Paramètres :

p : Player

xp : entier

Valeur de retour : rien

`player.getStrength(p)`

Description : renvoie la valeur de la force du joueur

Paramètre : p : Player

Valeur de retour : valeur de force du joueur

`player.setStrength(p, strength)`

Description : définit la valeur de la force du joueur

Paramètres :

p : Player

strength: flottant

Valeur de retour : rien

`player.getResistance(p)`

Description : renvoie la valeur de la résistance du joueur

Paramètre : p : Player

Valeur de retour : valeur de la résistance du joueur

`player.setResistance(p, resistance)`

Description : définit la valeur de la résistance du joueur

Paramètres :

p : Player

resistance : flottant

Valeur de retour : rien

`player.getDamage(p)`

Description : renvoie la valeur des dégâts du joueur

Paramètre : p : Player

Valeur de retour : valeur des dégâts du joueur

`player.setDamage(p, damage)`

Description : définit la valeur des dégâts du joueur

Paramètres :

p : Player

damage : flottant

Valeur de retour : rien

`player.getSelectedBow(p)` → booléen

Description : teste si le joueur a sélectionné un arc

Paramètre : p : Player

Valeur de retour : 1 si l'arc est sélectionné, sinon 0

`player.getInventory(p)` → booléen

Description : teste si le joueur possède sur lui l'item

Paramètres :

p : Player

item : liste

Valeur de retour : 1 si le joueur possède l'item, sinon 0

`player.getSprite(p)`

Description : Renvoie le caractère symbolisant le joueur

Paramètres :

p : Player

Valeur de retour : chaine de caractère

player**.setSprite(p, sprite)**

Description : Affecte le caractère symbolisant le joueur

Paramètres :

p : Player

sprite : chaine de caractère

Valeur de retour : rien

4.4. Dungeon.py

- dungeon**.create(name)**
- dungeon**.generate(d)**
- dungeon**.recursiveGeneration(d, remainingRooms, currentRoom, roomLevel, roomsTotalNumber)**
- dungeon**.pickUpRandomRoom(remainingRooms)**
- dungeon**.show(d)**
- dungeon**.getCurrentRoom(d)**
- dungeon**.setCurrentRoom(d, currentRoom)**
- dungeon .**getName(d)**
- dungeon.**setName(d, name)**

dungeon**.create(name)** → dungeon

Description : créer le donjon

Paramètres :

name : chaîne de caractère

Valeur de retour : dictionnaire donjon

dungeon**.generate(d)**

Description : Permet de générer le donjon

Paramètres :

d : Dungeon

Valeur de retour : rien

dungeon**.recursiveGeneration(d, remainingRooms, currentRoom, roomLevel, roomsTotalNumber)**

Description : génère les salles adjacentes à une case donnée. Elle est, par construction, récursive. Elle étudie pour chaque direction, s'il faut mettre (ou non), une salle aléatoire prise dans la liste des salles restantes.
Si oui, alors la fonction se rappelle-elle même pour définir les salles adjacentes de la salle venant d'être placé.

Paramètres :

d : Dungeon

remainingRooms : liste

currentRoom : chaine de caractère

roomLevel : entier

roomsTotalNumber : entier

Valeur de retour : rien

dungeon**.pickUpRandomRoom(remainingRooms)**

Description : Renvoie une salle prise aléatoirement dans la liste des salles restantes et l'enlève de la liste

Paramètres :

remainingRooms: liste

Valeur de retour : room

dungeon**.show(dungeon)**

Description : afficher la salle courante du donjon

Paramètres :

dungeon : Dungeon

Valeur de retour : rien

dungeon**.getCurrentRoom(d)**

Description : Renvoie la salle courante

Paramètres :

dungeon : Dungeon

Valeur de retour : chaine de caractère

dungeon**.setCurrentRoom(d, currentRoom)**

Description : Affecte la salle courante à afficher

Paramètres :

dungeon : Dungeon

currentRoom : chaine de caractère

Valeur de retour : rien

dungeon .**getName(d)**

Description : Renvoie une liste de nom de donjon

Paramètres :

dungeon : Dungeon

Valeur de retour : liste

dungeon.**setName(d, name)**

Description : Attribut un nom de salle

Paramètres :

dungeon : Dungeon

name : chaine de caractère

Valeur de retour : rien

4.5. Room.py

- room.**create(dungeonName, roomName)**
- room.**show(r)**
- room.**drawDoors(r)**
- room.**drawDoor(r, x, y, w, h)**
- room.**getChestByPosition(r, x, y)**
- room.**getMobByPosition(r, x, y)**
- room.**getUpRoom(r)**
- room.**setUpRoom(r, up_room)**
- room.**getDownRoom(r)**
- room.**setDownRoom(r, down_room)**
- room.**getLeftRoom(r)**
- room.**setLeftRoom(r, left_room)**
- room.**getRightRoom(r)**
- room.**setRightRoom(r, right_room)**

room.**create(dungeonName, roomName)**

Description : créer une salle à l'aide des infos du fichier xml

Paramètres :

dungeonName : chaîne de caractère

roomName : chaîne de caractère

Valeur de retour : rien

room.**show(r)**

Description : afficher une salle, avec ses coffres, ses monstres et ses flèches

Paramètre : r : room

Valeur de retour : rien

room.**drawDoors(r)**

Description : affiche les portes du donjon actuellement affiché

Paramètre : r : room

Valeur de retour : rien

room.**drawDoor(r, x, y, w, h)**

Description : Définie la taille et le positionnement des portes du Donjon et remplace dans le background de la salle, des caractères de bordure par des espaces.

Paramètre : r : room

Valeur de retour : rien

room.**getChestByPosition(r, x, y)**

Description : renvoie la valeur de la position du coffre

Paramètres :

r : Room

x : entier

y : entier

Valeur de retour : rien

room.**getMobByPosition(r, x, y)**

Description : renvoie la valeur de la position du monstre

Paramètres :

r : Room

x : entier

y : entier

Valeur de retour : rien

room.**getUpRoom(r)**

Description : renvoie le dictionnaire de la salle situé au-dessus de la salle courante

Paramètre : r : Room

Valeur de retour : Room

room.**setUpRoom(r, up_room)**

Description : définit la salle qui sera au-dessus de la salle courante

Paramètres :

r : Room

up_room : Room

Valeur de retour : rien

4.6. Chest.py

- chest.**create(x, y, donnee)**
- chest.**addItem(c, i)**
- chest.**getPosition(c)**
- chest.**setPosition(c, x, y)**
- chest.**getContent(c)**
- chest.**isInside(c, i)**

chest**.create(x, y, donnee)**

Description : créer un coffre

Paramètres :

x : entier

y : entier

donnee : liste

Valeur de retour : rien

chest**.addItem(c, item)**

Description : Ajouter des items au coffre

Paramètres :

c : Chest

item : Bow ou Bonus

Valeur de retour : rien

chest**.getPosition(c)**

Description : renvoie la position du coffre

Paramètre : c : Chest

Valeur de retour : tuple (x, y)

chest**.setPosition(c, x, y)**

Description : définit la position du coffre

Paramètres :

c : Chest

x : entier

y : entier

Valeur de retour : rien

chest**.getContent(c)**

Description : renvoie le contenu du coffre

Paramètre : c : Chest

Valeur de retour : content

chest**.isInside(c, item)**

Description : teste si l'item est dans le coffre

Paramètres :

c : Chest

item: Bow ou Bonus

Valeur de retour : booléen (true si c'est dedans, sinon false)

4.7. Bow.py

- Bow.**create(b, name)**
- Bow.**getName(b)**
- Bow.**setName(b, name)**
- Bow.**getDamage(b)**
- Bow.**setDamage(b, name)**
- Bow.**getSprite(b)**
- Bow.**setSprite(b, sprite)**

Bow.**create(b, name)**

Description : Créer un arc

Paramètres :

b : Bow

name : chaîne de caractère

Valeur de retour : rien

Bow.**getName(b)**

Description : renvoi le nom de l'arc

Paramètre : b : Bow

Valeur de retour : name (chaîne de caractère)

Bow.**setName(b, name)**

Description : définit le nom de l'arc

Paramètres :

b : Bow

name : chaîne de caractère

Valeur de retour : rien

Bow.**getDamage(b)**

Description : renvoi la valeur de dégât qu'inflige l'arme

Paramètre : b : Bow

Valeur de retour : damage (entier)

Bow.**setDamage(b, name)**

Description : définit la valeur de dégât qu'inflige l'arme

Paramètres :

b : Bow

name : chaîne de caractère

Valeur de retour : rien

Bow.**getSprite(b)**

Description : renvoi le caractère symbolisant la flèche

Paramètre : b : Bow

Valeur de retour : sprite (chaine de caractère)

Bow.**setSprite(b, sprite)**

Description : définit le caractère symbolisant la flèche

Paramètres :

b : Bow

sprite: chaîne de caractère

Valeur de retour : rien

4.8. Arrow.py

- Arrow.**create()**
- Arrow.**getPosition(a)**
- Arrow.**setPosition(a, x, y)**
- Arrow.**getSpeed(a)**
- Arrow.**setSpeed(a, speed)**
- Arrow.**getDamage(a)**
- Arrow.**setDamage(a, damage)**

Arrow.**create()**

Description : créer des flèches

Paramètres : aucune

Valeur de retour : aucune

Arrow.**getPosition(a)**

Description : renvoi la position de la flèche

Paramètres : a : arrow

Valeur de retour : tuple(x,y)

Arrow.**setPosition(a, x, y)**

Description : définir la position de la flèche

Paramètres :

a : arrow

x : entier

y : entier

Valeur de retour : rien

Arrow.**getSpeed(a)**

Description : renvoi la vitesse de la flèche

Paramètres : a : arrow

Valeur de retour : speed (flottant)

Arrow.**setSpeed(a, speed)**

Description : définir la vitesse de la flèche

Paramètres :

a : arrow

speed : flottant

Valeur de retour : rien

Arrow.**getDamage(a)**

Description : renvoi les dégâts de la flèche

Paramètres : a : arrow

Valeur de retour : damage (flottant)

Arrow.**setDamage(a, damage)**

Description : définir les dégâts de la flèche

Paramètres :

a : arrow

damage : flottant

Valeur de retour : rien

4.9. Bonus.py

- Bonus.**create()**
- Bonus.**getName(b)**
- Bonus.**setName(b, name)**
- Bonus.**getAmount(b)**
- Bonus.**setAmount(b, amount)**

Bonus.**create() → Chest**

Description : créer des bonus en fonction d'une liste de bonus

Paramètre : rien

Valeur de retour : item (liste)

Bonus.**getName(b) → Bonus**

Description : renvoi le nom du bonus

Paramètre : b : Bonus

Valeur de retour : name (chaîne de caractère)

Bonus.**setName(b, name) → Bonus**

Description : définit le nom du bonus

Paramètres :

b : Bonus

name : chaîne de caractère

Valeur de retour : rien

Bonus.**getAmount(b) → Bonus**

Description : renvoi la quantité de bonus

Paramètre : b : Bonus

Valeur de retour : amount (entier)

Bonus.**setAmount(b, amount) → Bonus**

Description : définit la quantité de bonus

Paramètres :

b : Bonus

amount : entier

Valeur de retour : rien

4.10. Mob.py

- Mob.**create()**
- Mob.**move(m)**
- Mob.**show()**
- Mob.**getType(m)**
- Mob.**setType(m, type)**
- Mob.**getPosition(m)**
- Mob. **setPosition(m, x, y)**
- Mob.**getHealth(m)**
- Mob.**setHealth(m, health)**
- Mob.**getStrength(m)**
- Mob.**setStrength(m, strength)**
- Mob.**getResistance(m, resistance)**
- Mob.**setResistance(m)**
- Mob.**getDamage(m)**
- Mob.**setDamage(m, damage)**
- Mob.**getSprite(m)**
- Mob.**setSprite(m, sprite)**

Mob.**create():**

Description : créer un monstre

Paramètres : aucun

Valeur de retour : rien

Mob.**move(m):**

Description : permet de déplacer le monstre

Paramètres : m : Mob

Valeur de retour : rien

Mob.**getType(m):**

Description : renvoi le type de monstre

Paramètres : m : Mob

Valeur de retour : type (chaîne de caractère)

Mob.**setType(m, type):**

Description : définit le type de monstre

Paramètres :

m : Mob

type : chaîne de caratère

Valeur de retour : rien

Mob.**getPosition(m):**

Description : renvoi la position du monstre

Paramètres : m : Mob

Valeur de retour : tuple (x, y)

Mob. **setPosition(m, x, y):**

Description : définit la position du monstre

Paramètres :

m : Mob

x : entier

y : entier

Valeur de retour : rien

Mob.**getHealth(m):**

Description : renvoi la valeur des points de vie restante du monstre

Paramètre : m : Mob

Valeur de retour : health (entier)

Mob.**setHealth(m, health):**

Description : définit la valeur des points de vie restante du monstre

Paramètre :

m : Mob

health : entier

Valeur de retour : rien

Mob.**getStrength(m):**

Description : renvoi la valeur des points de force du monstre

Paramètre : m : Mob

Valeur de retour : strength (flottant)

Mob.**setStrength(m, strength):**

Description : définit la valeur des points de force du monstre

Paramètre :

m : Mob

strength : flottant

Valeur de retour : rien

Mob.**getResistance(m):**

Description : renvoi la valeur des points de resistance du monstre

Paramètre : m : Mob

Valeur de retour : resistance (flottant)

Mob.**setResistance(m, resistance):**

Description : définit la valeur des points de resistance du monstre

Paramètre :

m : Mob

resistance : flottant

Valeur de retour : rien

Mob.**getDamage(m):**

Description : renvoi la valeur des points de dégât du monstre

Paramètre : m : Mob

Valeur de retour : damage (entier)

Mob.**setDamage(m, damage):**

Description : définit la valeur des points de dégât du monstre

Paramètre :

m : Mob

damage : entier

Valeur de retour : rien

Mob**.getSprite(m)**

Description : renvoi le caractère symbolisant le monstre

Paramètre : m : Mob

Valeur de retour : sprite (chaine de caractère)

Mob**.setSprite(m, sprite)**

Description : définit le caractère symbolisant le monstre

Paramètres :

m : Mob

sprite: chaîne de caractère

Valeur de retour : rien

4.11. Utils.py

- Utils.**goto(x,y)**
- Utils.**write(text, color, backgroundColor, textForm)**

Utils.**goto(x,y)**

Description : permet de placer le curseur aux coordonnées x, y

Paramètres :

x : entier

y: entier

Valeur de retour : rien

Utils.**write(text, color, backgroundColor, textForm)**

Description : permet d'afficher, de mettre en forme un texte (couleur du texte, couleur du fond de texte), ainsi que de le mettre en gras ou souligné

Paramètres :

text :chaine de caractère

color : chaine de caractère

backgroundColor: chaine de caractère

textForm: chaine de caractère

Valeur de retour : rien

4.12. Menu.py

- Menu.**create()**
- Menu.**setCurrentWindow(menu, windowName)**
- Menu.**getCurrentWindowName(menu)**
- Menu.**gameWindow(menu)**
- Menu.**getFrame(menu)**
- Menu.**getBackground(menu)**
- Menu.**getBackgroundColor(menu)**
- Menu.**show(menu)**
- Menu.**getButtonList(menu)**
- Menu.**getButtonSelected(menu)**
- Menu.**setButtonSelected(menu, buttonName)**
- Menu.**getTexts(menu)**
- Menu.**interact(menu, key)**
- Menu.**changeSelectedButton(menu, action)**
- Menu.**addPreviousWindow(menu, previousWindow)**
- Menu.**getPreviousWindow(menu)**
- Menu.**changeCurrentWindow(menu, buttonTargetList)**

Menu**.create() → Menu**

Description : Créer un menu à l'aide des informations disponible dans un fichier xml. La fonction create() permet de créer un graph d'état stocké dans un dictionnaire. Ainsi chaque fenêtre contient ses propres textes, boutons et fond.

Paramètres : rien

Valeur de retour : dictionnaire menu()

Menu**.setCurrentWindow(menu, windowName)**

Description : Attribue le nom et les données de la fenêtre à afficher dans la fenêtre courante

Paramètres :

windowName : chaine de caractère

Valeur de retour : rien

Menu**.getCurrentWindowName(menu)**

Description : Renvoie le nom de la fenêtre courante

Paramètres : rien

Valeur de retour : chaine de caractère

Menu**.gameWindow(menu) → booléen**

Description : Permet de savoir si on se trouve dans la fenêtre de jeu

Paramètres : rien

Valeur de retour : True si la fenetre courante est « game » sinon False

Menu**.getFrame(menu)**

Description : Renvoie le cadre du jeu

Paramètres : rien

Valeur de retour : liste de chaine de caractère

Menu**.getBackground(menu)**

Description : Renvoie le fond de la fenêtre courante

Paramètres : rien

Valeur de retour : liste de chaine de caractère

Menu**.getBackgroundColor(menu)**

Description : Renvoie la couleur du fond de la fenêtre courante

Paramètres : rien

Valeur de retour : tupple (couleur de texte, couleur de fond de texte)

Menu**.show(menu)**

Description : Affiche les éléments de la fenêtre courante (cadre du jeu, fond, boutons et textes de la fenêtre courante)

Paramètres : rien

Valeur de retour : rien

Menu.**getButtonList(menu)**

Description : Renvoie la liste des boutons de la fenêtre courante

Paramètres : rien

Valeur de retour : liste

Menu.**getButtonSelected(menu)**

Description : Renvoie le nom du bouton sélectionné de la fenêtre courante

Paramètres : rien

Valeur de retour : chaine de caractère

Menu.**setButtonSelected(menu, buttonName)**

Description : Définie le bouton sélectionné de la fenêtre courante

Paramètres :

buttonName : chaine de caractère

Valeur de retour : rien

Menu.**getTexts(menu)**

Description : Renvoie la liste des textes de la fenêtre courante

Paramètres : rien

Valeur de retour : liste

Menu.**interact(menu, key)**

Description : Attribue des actions en fonction des évènements claviers

Paramètres : rien

Valeur de retour : rien

Menu.**changeSelectedButton(menu, action)**

Description : en fonction des actions des évènements claviers, cette fonction permet de changer le bouton sélectionné de la fenêtre courante parmi la liste de bouton que possède la fenêtre.

Paramètres :

Action : chaine de caractère

Valeur de retour : rien

Menu.**addPreviousWindow(menu, previousWindow)**

Description : Permet d'ajouter le nom de la fenêtre précédente dans la liste de transition, afin de savoir d'où on vient.

Paramètres :

previousWindow: chaine de caractère

Valeur de retour : rien

Menu.**getPreviousWindow(menu)**

Description : Renvoie le nom de la fenêtre précédente de la liste de transition

Paramètres : rien

Valeur de retour : chaine de caractère

Menu**.changeCurrentWindow(menu, buttonTargetList)**

Description : Permet de changer de fenêtre courante.

Paramètres :

buttonTargetList: liste

Valeur de retour : rien

5. Calendrier et suivi de développement

5.1. P1 :

5.1.1. Fonction à développer

Fonctions	Codées	Testées	Fait par
Main.**main()**	X	X	Dorian
Main.**init()**	X	X	Dorian
Main.**run()**	X	X	Dorian
Main.**show()**	X	X	Dorian
Main.**interact()**	X	X	Dorian
Main.**quit()**	X	X	Dorian
Game.**create()**	X	X	Evan
Game.**show(g)**	X	X	Evan
Game.**interact(g, key)**	X	X	Evan
Game**.switchRoom(g)**	X	X	Evan
Player.**create()**	X	X	Evan
Player**.getPosition(p)**	X	X	Dorian
Player**.setPosition(p, x, y)**	X	X	Dorian
Player.**move(p, direction)**	X	X	Dorian
Player.**show(p)**	X	X	Evan
Player**.getSprite(p)**	X	X	Evan
Player**.setSprite(p, sprite)**	X	X	Evan
Dungeon**.create(name)**	X	X	Evan
Dungeon**.generate(d)**	X	X	Evan
Dungeon**.recursiveGeneration(d, remainingRooms, currentRoom, roomLevel, roomsTotalNumber)**	X	X	Evan
Dungeon**.pickUpRandomRoom(remainingRooms)**	X	X	Evan
Dungeon**.show(dungeon)**	X	X	Evan
Dungeon.**getCurrentRoom(d)**	X	X	Evan
Dungeon.**setCurrentRoom(d, currentRoom)**	X	X	Evan
Dungeon.**getName(d)**	X	X	Evan
Room**.create(dungeonName, roomName)**	X	X	Evan
Room**.show(r)**	X	X	Evan
Room.**drawDoors(r)**	X	X	Evan
Room.**drawDoor(r, x, y, w, h)**	X	X	Evan
Room**.getChestByPosition(r, x, y)**	X	X	Evan
Room**.getMobByPosition(r, x, y)**	X	X	Evan
Room**.getUpRoom(r)**	X	X	Evan
Room**.setUpRoom(r, up_room)**	X	X	Evan

Room.**getDownRoom(r)**	X	X	Evan
Room.**setDownRoom(r, down_room)**	X	X	Evan
Room.**getLeftRoom(r)**	X	X	Evan
Room.**setLeftRoom(r, left_room)**	X	X	Evan
Room.**getRightRoom(r)**	X	X	Evan
Room.**setRightRoom(r, right_room)**	X	X	Evan
Utils.**goto(x, y)**	X	X	Evan
Utils.**write(text, color, backgroundColor, textForm)**	X	X	Dorian
Menu.create()	X	X	Dorian
Menu.**setCurrentWindow(menu, windowName)**	X	X	Dorian
Menu.**getCurrentWindowName(menu)**	X	X	Dorian
Menu.**gameWindow(menu)**	X	X	Dorian
Menu.**getFrame(menu)**	X	X	Dorian
Menu.**getBackground(menu)**	X	X	Dorian
Menu.**getBackgroundColor(menu)**	X	X	Dorian
Menu.**show(menu)**	X	X	Dorian
Menu.**getButtonList(menu)**	X	X	Dorian
Menu.**getButtonSelected(menu)**	X	X	Dorian
Menu.**setButtonSelected(menu, buttonName)**	X	X	Dorian
Menu.**getTexts(menu)**	X	X	Dorian
Menu.**interact(menu, key)**	X	X	Dorian
Menu.**changeSelectedButton(menu, action)**	X	X	Dorian
Menu.**addPreviousWindow(menu, previousWindow)**	X	X	Dorian
Menu.**getPreviousWindow(menu)**	X	X	Dorian
Menu.**changeCurrentWindow(menu, buttonTargetList)**	X	X	Dorian

5.1.2. Fichier XML :

Nom de l'image	Créer par	Commentaires
Menu.xml	Dorian	Contient toutes les fenêtres du menu
forest_1.xml	Evan	Fichier XML pour générer les salles du donjon forêt.
forest_2.xml	Evan	
forest_3.xml	Evan	

5.2. P2 :

5.2.1. Fonctions à développer

Fonctions	Codées	Testées	Commentaires
player.**getHealth(p)**			
player.**setHealth(p, health)**			
player.**getXp(p)**			
player.**setXp(p, xp)**			
player. **getStrength(p)**			
player.**setStrength(p, strength)**			

player.**getResistance(p)**			
Player.**getDamage(p)**			
Player.**setDamage(p, damage)**			
Player.**getSelectedBow(p)**			
Player.**getInventory(p)**			
chest**.create(x, y, donnee)**			
chest**.addItem(c, i)**			
chest**.getPosition(c)**			
chest**.setPosition(c, x, y)**			
chest**.getContent(c)**			
chest**.isInside(c, i)**			
Bow.**create(b, name)**			
Bow.**getName(b)**			
Bow.**setName(b, name)**			
Bow.**getDamage(b)**			
Bow.**setDamage(b, name)**			
Bow.**getSprite(b)**			
Bow.**setSprite(b, sprite)**			
Arrow.**create()**			
Arrow.**getPosition(a)**			
Arrow.**setPosition(a, x, y)**			
Arrow.**getSpeed(a)**			
Arrow.**setSpeed(a, speed)**			
Arrow.**getDamage(a)**			
Arrow.**setDamage(a, damage)**			
Bonus.**create()**			
Bonus.**getName(b)**			
Bonus.**setName(b, name)**			
Bonus.**getAmount(b)**			
Bonus.**setAmount(b, amount)**			
Mob.**create()**			
Mob.**move(m)**			
Mob.**show(m)**			
Mob.**getType(m)**			
Mob.**setType(m, type)**			
Mob.**getPosition(m)**			
Mob. **setPosition(m, x, y)**			
Mob.**getHealth(m)**			
Mob.**setHealth(m, health)**			
Mob.**getStrength(m)**			
Mob.**setStrength(m, strength)**			
Mob.**getResistance(m, resistance)**			
Mob.**setResistance(m)**			
Mob.**getDamage(m)**			
Mob.**setDamage(m, damage)**			
Mob.**getSprite(m)**			
Mob.**setSprite(m, sprite)**			