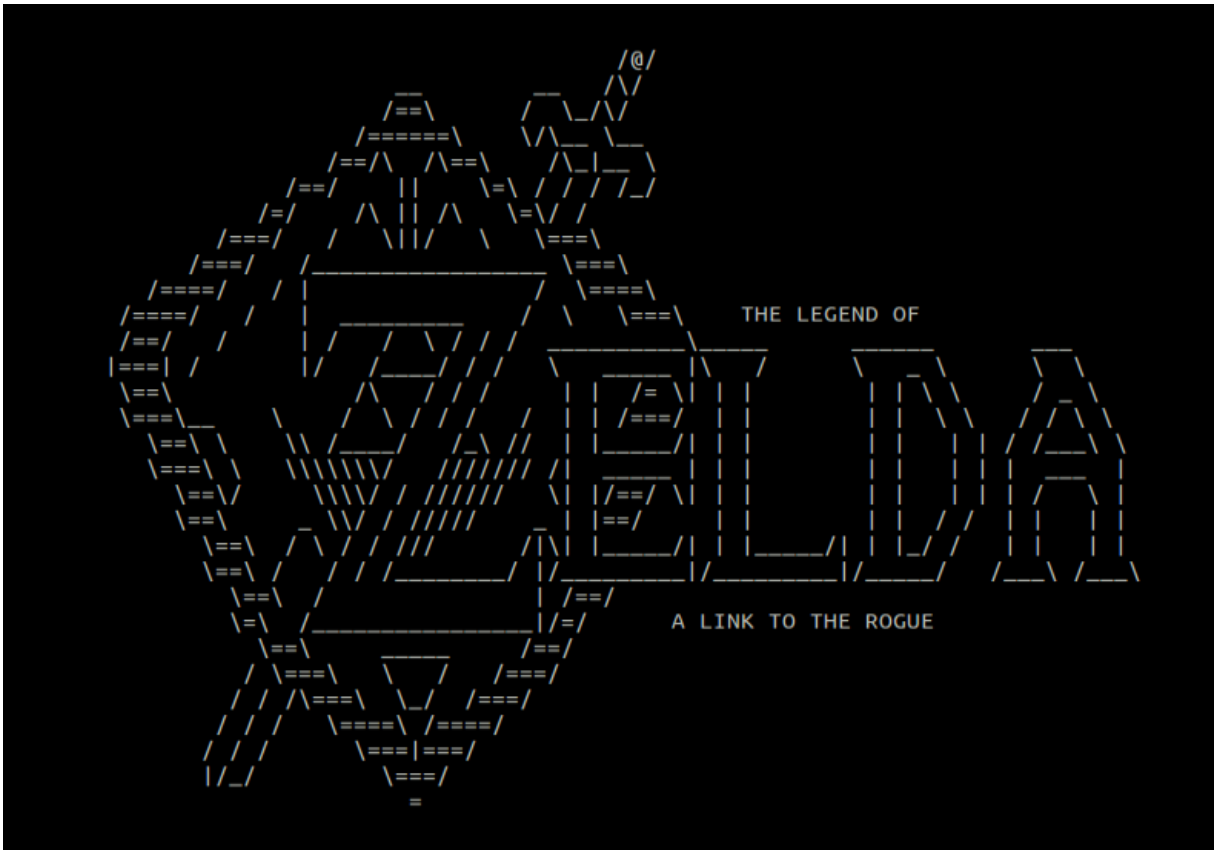
CONCEPTION



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	1.0
Date	01/04/2016

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, 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.

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 textes : background.txt, cadre.txt, victory.txt, defeat.txt, room.txt, doors.txt.

Ces images nous permettront de dessiner dans le terminal, d'abord l'interface de l'utilisateur, mais aussi 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, il y a de nombreux items qui ont chacune leurs propres caractéristiques et qui 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 :

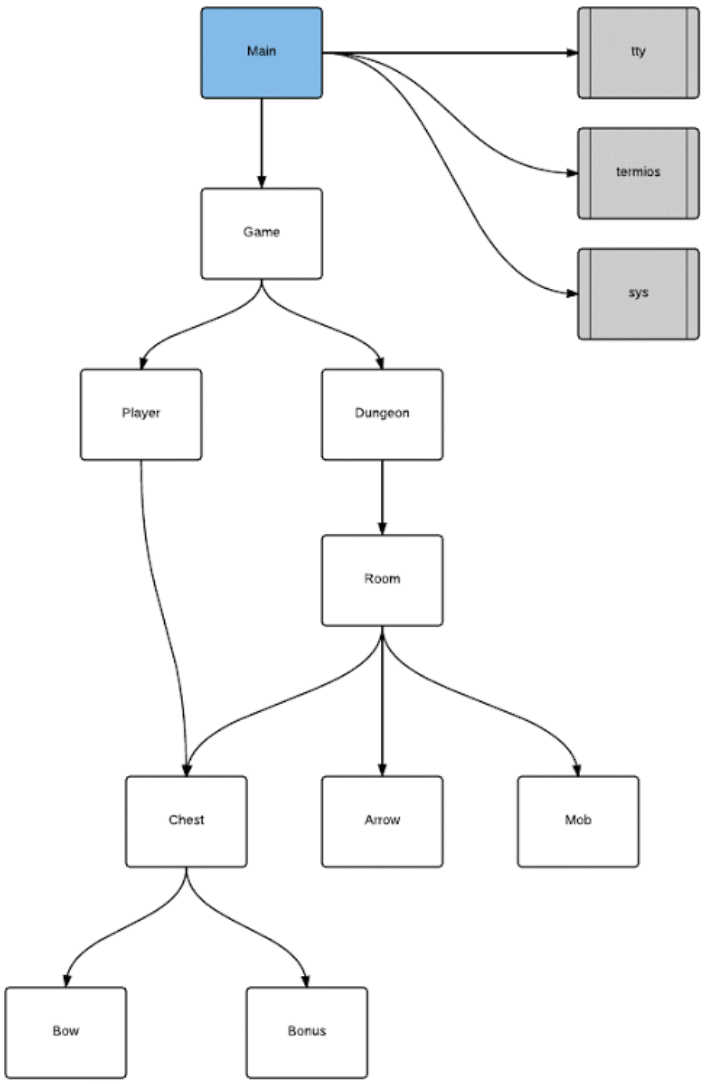
  Afficher, choisir, jouer, générer, déplacer, attaquer, ouvrir, finir, recommencer, quitter

- Nom :

  Joueur, salle, score, caractéristique, partie, résultat,

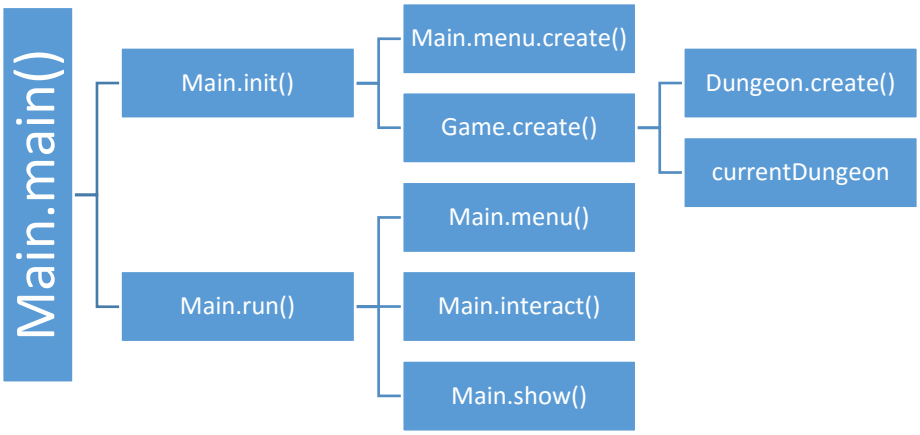
3.2. Types de donnée

3.3. Dépendance entre modules

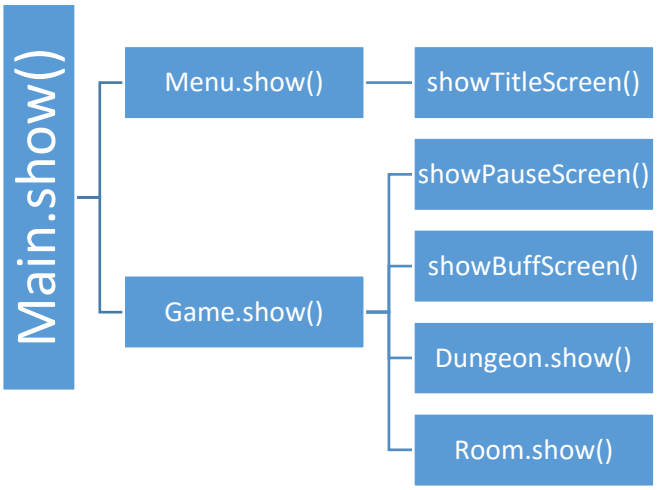


3.4. Analyse descendante

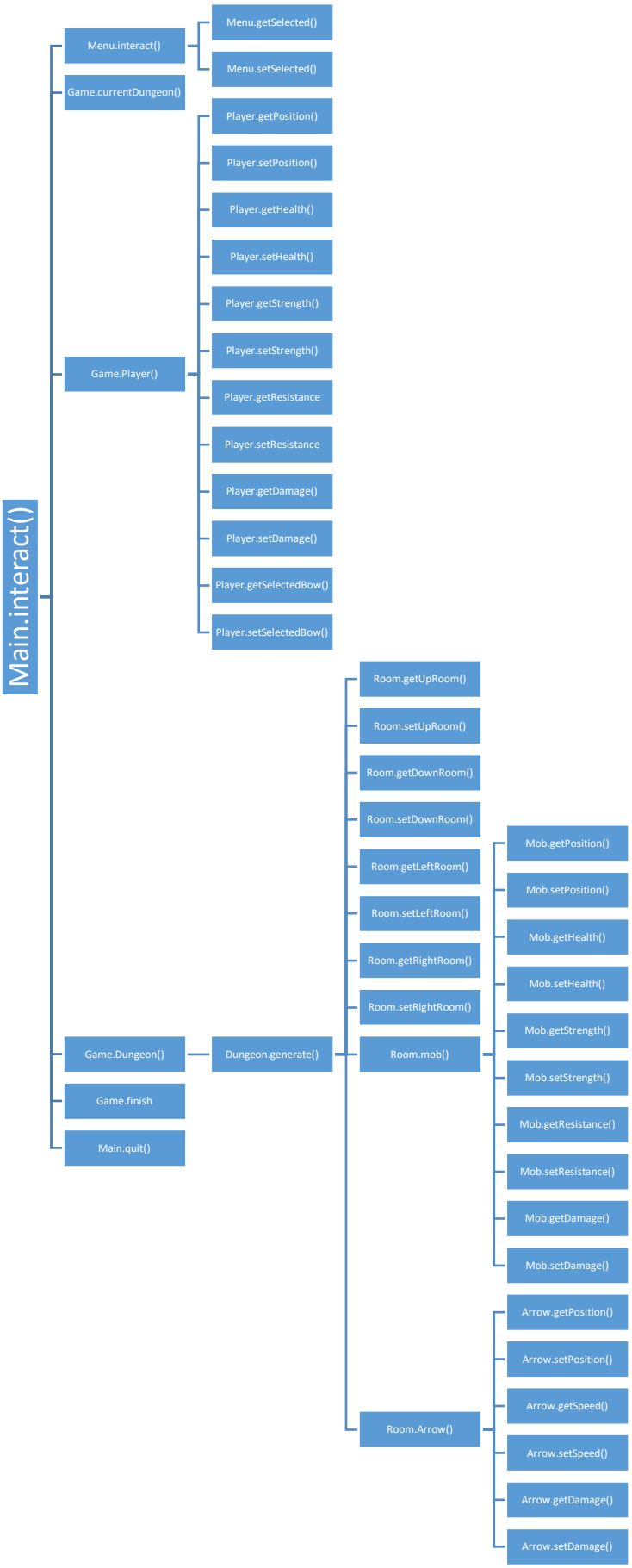
3.4.1. Arbre principal



3.4.2. Arbre affichage



3.4.3. Arbre interaction



4. Description des fonctions

4.1. Programme Principal : Main.py

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

Main.**main() :**

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

Main.**init() :**

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

Main.**run() :**

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

Main.**show() :**

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

Main.**interact() :**

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

Main.**quit() :**

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

4.2. Game.py

- Game.**create() :**
- Game.**player() :**
- Game.**dungeon() :**

Game.create() → Game

Description : permet de créer une session de jeu

Paramètre : aucun

Valeur de retour : aucune

Game.**player()** → Game

Description : fonction principale pour les interactions du joueur

Paramètre : aucun

Valeur de retour : aucune

Game.**dungeon() :**

Description : fonction principale pour générer le donjon et géré les salles

Paramètre : aucun

Valeur de retour : aucune

4.3. Player.py

- player**.create()**
- player**.getPosition(p)**
- player**.setPosition(p, x, y)**
- 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**.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**.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 force du joueur

Paramètre : p : Player

Valeur de retour : valeur de force du joueur

player**.setStrength(p, strength) :**

Description : définit la valeur de 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, item) :** → booléen

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

Paramètres :

p : Player

item : Bow ou Bonus

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

4.4. Dungeon.py

- dungeon**.create(name):**
- dungeon**.generate(d):**
- dungeon**.show(dungeon) :**

dungeon**.create(name):**

Description : créer le donjon

Paramètre :

name : chaîne de caractère

Valeur de retour : donjon vide

dungeon**.generate(d) :**

Description : généré le donjon

Paramètre :

d : Dungeon

Valeur de retour : rien

dungeon**.show(dungeon) :**

Description : afficher les salles du donjon

Paramètre :

dungeon : Dungeon

Valeur de retour : rien

4.5. Room.py

- room**.create(dungeonName, roomName):**
- room**.show(r) :**
- 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

Paramètres :

dungeonName : chaîne de caractère

roomName : chaîne de caractère

Valeur de retour : rien

room**.show(r) :**

Description : afficher une salle

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

room**.getDownRoom(r) :**

Description : renvoie le dictionnaire de la salle situé en-dessous de la salle courante

Paramètre : r : Room

Valeur de retour : Room

room**.setDownRoom(r, down_room) :**

Description : définit la salle qui sera en-dessous de la salle courante

Paramètres :

r : Room

down_room : Room

Valeur de retour : rien

room**.getLeftRoom(r) :**

Description : renvoie le dictionnaire de la salle situé sur le côté gauche de la salle courante

Paramètre : r : Room

Valeur de retour : Room

room**.setLeftRoom(r, left_room):**

Description : définit la salle qui sera sur le côté droit de la salle courante

Paramètre :

r : Room

left_room : Room

Valeur de retour : rien

room**.getRightRoom(r) :**

Description : renvoie le dictionnaire de la salle situé sur le côté droit de la salle courante

Paramètre : r : Room

Valeur de retour : Room

room**.setRightRoom(r, right_room):**

Description : définit la salle qui sera sur le côté droit de la salle courante

Paramètres :

r : Room

right_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.**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

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.**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.**create():**

Description : créer un monstre
Paramètres : aucun
Valeur de retour : rien

Mob.**move(m):**

Description : permet de déplacer le monstre
Paramètres : m : monstre
Valeur de retour : rien

Mob.**getType(m):**

Description : renvoi le type de monstre
Paramètres : m : monstre
Valeur de retour : type (chaîne de caractère)

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

Description : définit le type de monstre
Paramètres :
m : monstre
type : chaîne de caratère
Valeur de retour : rien

Mob.**getPosition(m):**

Description : renvoi la position du monstre
Paramètres : m : monstre
Valeur de retour : tuple (x, y)

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

Description : définit la position du monstre

Paramètres :

m : monstre

x : entier

y : entier

Valeur de retour : rien

Mob.**getHealth(m):**

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

Paramètre : m : monstre

Valeur de retour : health (entier)

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

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

Paramètre :

m : monstre

health : entier

Valeur de retour : rien

Mob.**getStrength(m):**

Description : renvoi la valeur des points de force du monstre

Paramètre : m : monstre

Valeur de retour : strength (flottant)

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

Description : définit la valeur des points de force du monstre

Paramètre :

m : monstre

strength : flottant

Valeur de retour : rien

Mob.**getResistance(m):**

Description : renvoi la valeur des points de resistance du monstre

Paramètre : m : monstre

Valeur de retour : resistance (flottant)

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

Description : définit la valeur des points de resistance du monstre

Paramètre :

m : monstre

resistance : flottant

Valeur de retour : rien

Mob.**getDamage(m):**

Description : renvoi la valeur des points de dégât du monstre

Paramètre : m : monstre

Valeur de retour : damage (entier)

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

Description : définit la valeur des points de dégât du monstre

Paramètre :

m : monstre

damage : entier

Valeur de retour : rien

5. Calendrier et suivi de développement

5.1. P1 :

5.1.1. Fonction à développer

Fonctions	Codées	Testées	Commentaires
Main.**main() :**			
Main.**init() :**			
Main.**run() :**			
Main.**show() :**			
Main.**interact() :**			
Main.**quit() :**			
Game.**create() :**			
Game.**player() :**			
Game.**dungeon() :**			
player**.create()**			
player**.getPosition(p)**			
player**.setPosition(p, x, y)**			
dungeon**.create(name)**			
dungeon**.generate(d)**			
dungeon**.show(dungeon)**			
room**.create(dungeonName, roomName):**			
room**.show(r) :**			
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):**			

5.1.2. Image ASCII :

Nom de l'image	Créer	Commentaires
Titlescreen.txt		
Victory.txt		
Defeat.txt		
Cadre.txt		

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