Document de Conception

Knil Dungeon

Type	Cahier des Charges
Nom du projet	Knil Dungeon
Auteur	Biannic Romain / Nash-Braguier Paul

1 Rappel du cahier des charges

1.1 Contraintes techniques

-Le jeu devra fonctionner sur les machines de TP de l'ENIB.

-Le développement sera en python

1.2 Fonctionnalités

F1 : Afficher le menu principal
- F1.1 : Reprendre partie
- F1.2 : Recommencer partie
- F1.3 : Quitter Jeu

F2 : Jouer au jeu
- F2.1 : Afficher jeu
  - carte
  - monstre
  - PNJ
  - personnage
  - dialogue
- F2.2 : Interagir avec carte
- F2.3 : Interagir avec PNJ
- F2.4 : Afficher Inventaire
  - F2.4.1 : Se déplacer dans l'inventaire
  - F2.4.2 : Sélectionner inventaire
- F2.5 : Afficher journal des quêtes

F3 : Finir jeu
- F3.1 : Afficher résultat
- F3.2 : Revenir au menu principal

1.3 Prototype 1

Ce prototype porte sur la création de la carte et sur l'interaction
Mise en oeuvre des fonctionnalités : F1.1, F1.2, F1.3, F2.1, F2.2, F2.3
Livré dans une archive au format .zip ou .tgz
Contient un manuel d'utilisation dans le fichier readme.txt

1.4 Prototype 2

Ce prototype réalise toutes les fonctionnalités.
Ajout au prototype 1 des fonctionnalités F2.4.1, F2.4.2, F2.5, F3.1, F3.2
Livré dans une archive au format .zip ou .tgz
Contient un manuel d'utilisation dans le fichier readme.txt

2 Principes des solutions techniques

2.1 Langage

Conformément aux contraintes énoncées dans le cahier des charges, le codage est réalisé avec 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 donnée 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 et les événements clavier.

2.3.2 Affichage

L'affichage se fait 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 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.

2.4 Stockage du jeu

Toutes les informations nécessaires au fonctionnement du jeu seront stockées dans un même dictionnaire composé de plusieurs sous-dictionnaires appelés « game ».

2.4.1 «Background»

Contient les informations concernant l'affichage du jeu (origine et taille de la fenêtre utilisée sur la console) et la carte.

2.4.1.1 «map »

Pour modéliser la carte, nous utiliserons une liste de liste de dictionnaires contenant les caractères utilisés pour représenter la case, la couleur, la collision, etc.

2.4.1 « knil »

Knil sera le personnage jouable, le dictionnaire « knil » contiendra les caractères utilisés pour l'affichage, son inventaire, son orientation et sa position relative à l'écran.

2.4.2 « PNJ » (personnage non jouable)

Ce dictionnaire aura comme clés les noms des personnages et comme valeurs les dictionnaires correspondants à ces derniers (couleur, position, etc..)

2.4.3 « quest »

Ce dictionnaire aura comme clés les noms de la quête et comme valeurs les dictionnaires correspondants à ces derniers (les actions à faire, l'étape de la quête et les récompenses).

2.4.4 « Dialog_box »

Ce dictionnaire indique la forme, la taille et l'origine de la boîte de dialogue

3 Analyse

3.1 Analyse Noms/Verbes

- Verbes :

Afficher, Interagir, Utiliser, Se déplacer, Sélectionner, Accomplir, Finir, Revenir

-Noms :

Menu Principal, Partie, Jeu, Carte, Monstre, PNJ, Personnage, Dialogue, Objet, Inventaire, Quêtes, Résultats

3.2 Types de données

type : Game = struct
    background : Background
    knil : Knil
    pnjs : struct de PNJ
    monster : struct de Monster
    dialog_box : Dialog_box
    quest : struct de Quest
    COLS : entier
    ROWS : entier
fstruct

type : Background = struct
    COLS : entier
    ROWS : entier
    x : entier
    y : entier
    anchor : chaîne
    map : grille de Case
fstruct

type : Case = struct
	fontcolor : tuple(entier, entier, entier)
	bgcolor : tuple(entier, entier, entier)
	crossable : entier
	c : tuple(caractères, caractères)
	special : aucun ou Special
fstruct

type : Special = struct
	fontcolor : tuple(entier, entier, entier)
	bgcolor : tuple(entier, entier, entier)
	crossable : entier
	c : tuple(caractères, caractères)
fstruct

type : Knil = struct
	x : entier
	y : entier
	front : entier
	graphism : tuple(caractère, caractère, caractère, caractère)
	c : caractère
	inventory : Inventory
fstruct

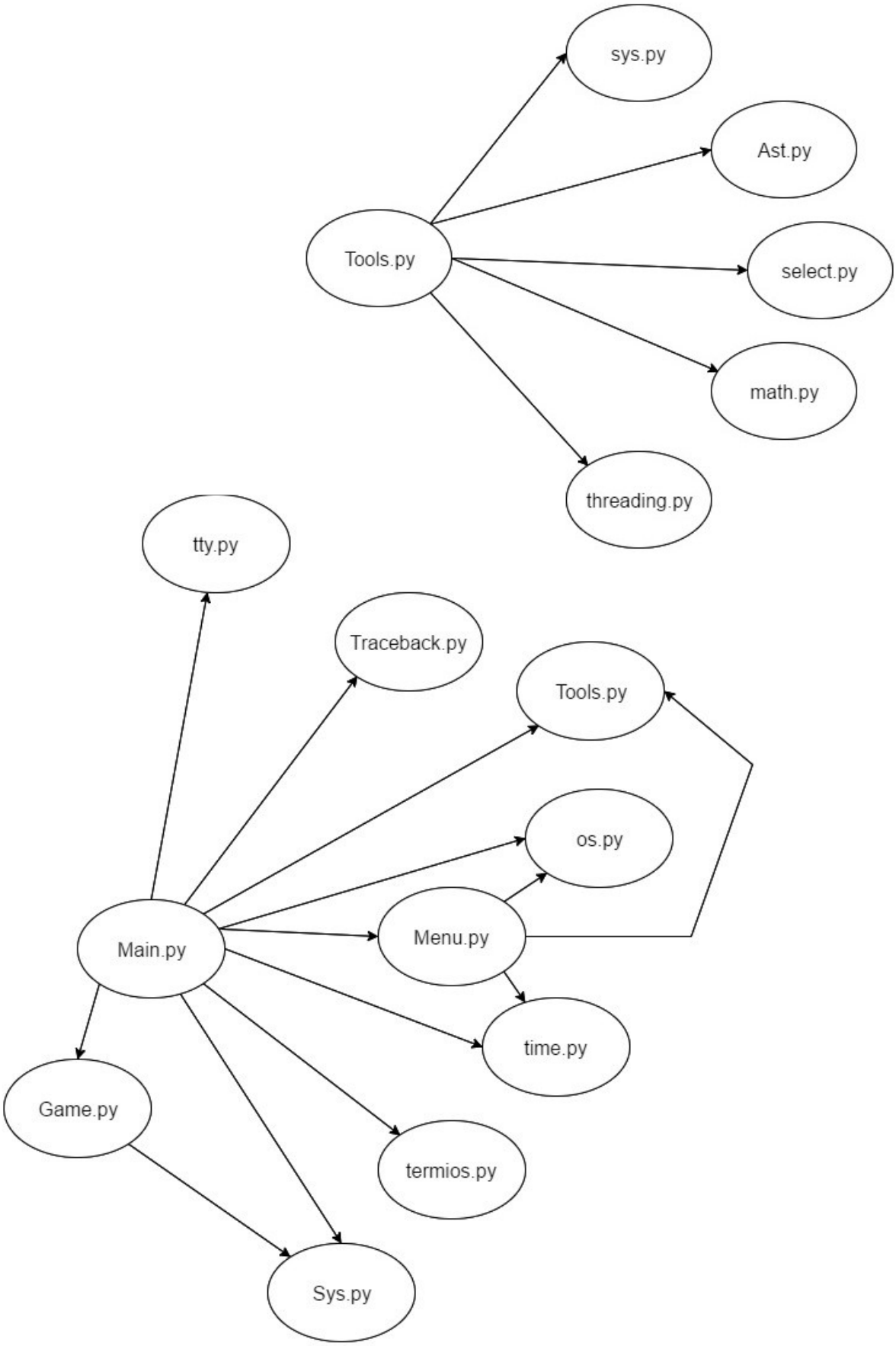
type : Inventory = struct
	name : (chaîne, fonction)
fstruct

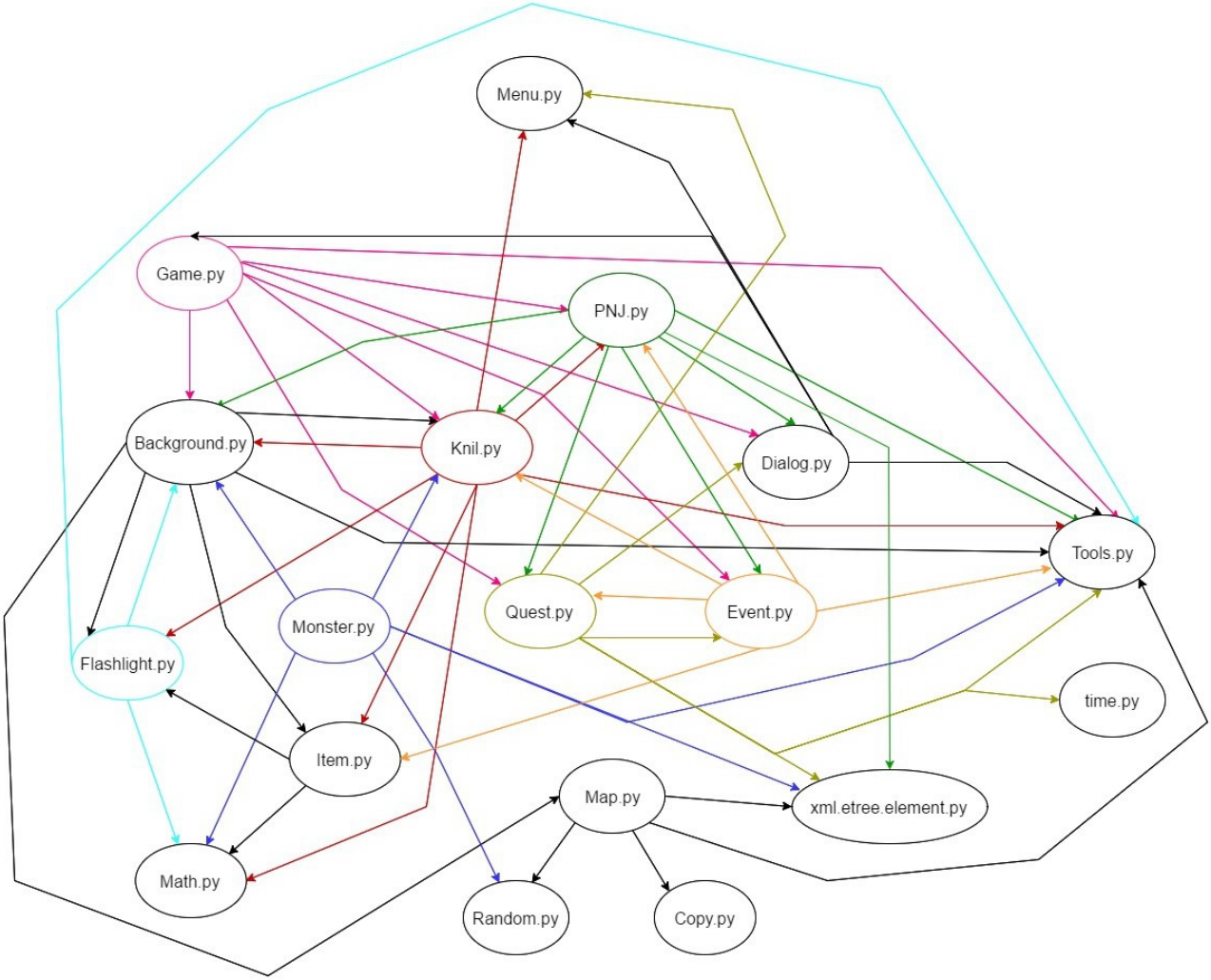
type : Flashlight = struct
	name : chaîne
	alpha : float
	delta : float
	scope : int
	activated : Booléen
fstruct

type : PNJ = struct
	position : tuple(entier, entier)
	c : caractères
	color : entier ou tuple d'entier ou chaîne
fstruct

```
type : Monster = struct
        initial_position : tuple(entier, entier)
        position : tuple(entier, entier)
        c : caractères
        color :
        sight : entier
fstruct

type : Dialog_box = struct
        ylen : entier
        xlen : entier
        COLS : entier
        ROWS : entier
        x : entier
        y : entier
fstruct

type : Quest = struct
        step_lvl : entier
        step_way : Step_way
        details : chaîne
        state : chaîne
        auto_text :  Auto_text
fstruct

type : Auto_text = struct
        chaîne: entier
fstruct

type : Step_way = struct
        entier : tuple (chaîne, liste de chaîne)
fstruct
```

3.3 Dépendance entre modules




Menu.py
Game.py
PNJ.py
Background.py
Knil.py
Dialog.py
Tools.py
Quest.py
Event.py
Monster.py
Flashlight.py
time.py
Item.py
Map.py
xml.etree.element.py
Math.py
Random.py
Copy.py

3.4.1 Arbre principal :

```
Main.main()
+------- Main.init()
|        +------- Game.create()
|        |        +------- Background.create()
|        |        |        + ------  Map.create()
|        |        |
|        |        +------- Knil.create()
|        |        +------- Dialog.dialog_box_create()
|        |        +------- Event.create()
|        |        +------- PNJ.create()
|        |        +------- Quest.create()
|        |        +------- Monster.create()
|        |        +------- Object.create()
|        |
|        +------- Menu.create()
|
+------- Main.run()
         +------- Menu.run()
         +------- Game.isrunning()
         +------- Tools.interact()
         +------- Game.display()
```

3.4.2 Arbre affichage (monde):

```
Game.display()
+------- Game.getDimension()
+------- Background.show()
|        +------- Background.getOrigin()
|        +------- Knil.getPos()
|        +------- Background.getCase()
|        +------- Background.getfontcolor()
|        +------- Background.getBgcolor()
|        |        +------- Knil.is_in_inventory()
|        |        +------- Knil.getPos()
|        |        +------- Background.getOrigin()
|        |        +------- Background.getMap()
|        |        +------- Flashlight.get_highlighted_case_pos()
|        |
|        +------- Tools.color()
|
+------- Knil.show()
|        +------- Background.getOrigin()
|        +------- Background.getCase()
|        +------- Tools.color()
|
+------- PNJ.getPos()
+------- Background.getOrigin()
+------- PNJ.display()
|        +------- Background.getOrigin()
|        +------- Background.is_case_special()
|        +------- Background.is_case_special_activated()
|        +------- Background.getCase()
|
+------- Monster.getPos()
+------- Monster.display()
         +------- Knil.is_in_dungeon()
         +------- Background.getOrigin()
         +------- Background.getCase()
```

3.4.3 Arbre affichage (menu) :

```
Menu.showall()
+------- Menu.display_frame()
+------- Menu.show_cursor_framing()
+------- Menu.run()
        +------- Menu.clearzone()
        +------- Menu.showall()
        +------- Tools.interact()
        +------- Menu.erase_case()
        |       +------- Menu.display_frame()
        +------- Menu.move()
        +------- Menu.show_cursor_framing()
        +------- Menu.validate()
```

3.3.4 Arbre interaction :

```
Main.run()
+------- Tools.interact()
+------- Game.move()
|       +------- Knil.point()
|       +------- Background.canSlide()
|       +------- Knil.collide()
|       +------- Knil.move()
|       +------- Background.slide()
|
+------- Game.exe()
|       +------- Background.getOrigin()
|       +------- Knil.getFrontPos()
|       +------- PNJ.getPos()
|       +------- PNJ.exe()
|               +------- Event.generate()
|               |       +------- Quest.check_quest()
|               |               +------- Event.remove()
|               |               +------- Event.generate()
|               |                       +------- Quest.finish_quest()
|               |               +------- Event.add()
|               +------- Dialog.run_dialog()
+------- Menu.run()
        +------- Menu.clearzone()
        +------- Menu.showall()
        +------- Tools.interact()
        +------- Menu.erase_case()
        |       +------- Menu.display_frame()
        +------- Menu.move()
        +------- Menu.show_cursor_framing()
        +------- Menu.validate()
```

4 Description des fonctions

4.1 Main.py

-Main.init()
-Main.run()
-Main.exception()
-Main.quit()

Main.init() → rien
Description : Initialisation du jeu
Paramètres : aucun
Valeur de retour : aucune

Main.run() → rien
Description : Boucle de simulation, si une erreur survient le try/except permet d'éviter que la console se casse
Paramètres : aucun
Valeur de retour : aucune

Main.exception() → rien
Description : Permet d'afficher l'erreur, dans le cas ou le run la releverait
Paramètres : aucun
Valeur de retour : aucune

Main.quit() → rien
Description : Arrête le jeu et remet les paramètres de base à la console
Paramètres : aucun
Valeur de retour : aucune

4.2 Game.py

Game.create(ROWS, COLS) → dict
Description : Crée une partie
Paramètres :
ROWS : entier
COLS : entier
Valeur de retour : dictionnaire contenant les informations du jeu

Game.move(g, steer) → Booléen
Description : Déplace les différents éléments du jeu si c'est possible (Background, Knil etc..)
Paramètres :
g: Game
steer: chaîne
Valeur de retour : True s'il y a une modification de l'affichage sinon False

Game.display(g) → rien

Description : affiche le jeu

Paramètres :

g: Game

Valeur de retour : aucune

Game.exe(g) → Booléen

Description : Vérifie si il y a une interaction possible si oui, appelle la fonction associée

Paramètres :

g: Game

Valeur de retour : True

Game.isRunning(g) → Booléen

Description : Indique si le jeu est en marche ou non

Paramètres :

g: Game

Valeur de retour : True si le jeu est en marche sinon False

Game.setRunning(g, v) → rien

Description : Permet de la lancer ou d'arrêter le jeu

Paramètres :

g: Game

v : Booléen

Valeur de retour : aucun

Game.getDimension(g) → entier, entier

Description : Renvoie la taille de la fenêtre affichée sur la console

Paramètres :

g: Game

Valeur de retour : taille de l'affichage

4.3 Knil.py

Knil.create(x, y) → dict

Description : Crée le personnage jouable

Paramètres :

x : entier

y : entier

Valeur de retour : Knil

Knil.point(k, steer) → Booléen ou rien

Description : Permet de changer l'orientation du personnage et met à jour les éléments dépendant de son orientation

Paramètres :

k : Knil

steer : chaîne

Valeur de retour : True si le caractère a été modifié sinon aucun

Knil.move(k, steer) → rien

Description : Déplace le personnage

Paramètres :

k : Knil

steer : chaîne

Valeur de retour : aucun

Knil.collide(k, steer, b, pnjs) → Booléen

Description : Vérifie si il y a une case empêchant le déplacement sur le chemin du personnage

Paramètres :

k : Knil

steer : chaîne

b : Background

pnjs : PNJs

Valeur de retour : True si il y a collision sinon False

Knil.show(k, b) → rien

Description : Affiche le personnage

Paramètres :

k : Knil

b : Background

Valeur de retour : aucun

Knil.getPos(k) → entier, entier

Description : Renvoie la position (relative à l'écran) du personnage

Paramètres :

k : Knil

Valeur de retour : coordonnées du personnage

Knil.getFrontPos(k) → entier, entier

Description : Donne les coordonnées de la case directement devant le personnage

Paramètres :

k : Knil

Valeur de retour : coordonnées de la case

Knil.open_inventory(k) → rien

Description : Affiche l'inventaire du joueur

Paramètres :

k : Knil

Valeur de retour : aucun

Knil.add_inventory(k, item) → rien

Description : Ajoute un objet à l'inventaire du joueur

Paramètres :

k : Knil

item : Item

Valeur de retour : aucun

Knil.is_in_inventory(k, item_name) → Booléen

Description : Vérifie si tel objet est dans l'inventaire

Paramètres :

k : Knil

item_name : chaîne

Valeur de retour : True si l'objet y est sinon False

4.4 PNJ.py

PNJ.create(xml_file_path) → dict

Description :Crée un dictionnaire contenant les informations des PNJs

Paramètres :

xml_file_path : chaîne

Valeur de retour : PNJ

PNJ.display(pnj, b ,k) → rien

Description :Affiche les pnjs à l'écran sauf s'ils sont sous un toit

Paramètres :

pnj:PNJ

b : Background

k : Knil

Valeur de retour : aucun

PNJ.setColor(pnj, color) → rien

Description : Change la couleur d'un pnj

Paramètres :

pnj:PNJ

color : entier

Valeur de retour : aucun

PNJ.exe(pnj, q ,d) → rien

Description : genère l’evenement associé à l’execution, et débute le dialogue associé

Paramètres :

pnj:PNJ

q : Quest

d : Dialog_box

Valeur de retour : aucune

PNJ.getPos(pnj) → tuple(entier, entier)

Description : Renvoie les coordonnées du pnj associé

Paramètres :

pnj:PNJ

Valeur de retour : coordonnées du pnj

4.5 Background.py

Background.create(ROWS, COLS, x, y, anchor="nw") → dict

Description : Crée le dictionnaire contenant les informations sur la carte

Paramètres :

ROWS : entier

COLS : entier

x : entier

y : entier

anchor : chaîne

Valeur de retour : Background

Background.slide(b, steer) → rien

Description : Déplace la carte

Paramètres :

b : Background

steer : chaîne

Valeur de retour : aucune

Background.canSlide(b, steer) → Booléen

Description : Vérifie si l'ont peut déplacer la carte

Paramètres :

b : Background

steer : chaîne

Valeur de retour : True si possible sinon False

Background.show(b, k) → rien

Description : Affiche la carte en tenant compte de l'emplacement de Knil (dans une maison etc..)

Paramètres :

b : Background

k : Knil

Valeur de retour : aucune

Background.getOrigin(b) → entier, entier

Description : Renvoie l'origine à partir duquel l'écran s'affiche

Paramètres :

b : Background

Valeur de retour : Coordonnées de l'origine

Background.is_case_special(b, location) → Booléen

Description : Indique si la case a un attribut 'special'

Paramètres :

b : Background

location : entier, entier

Valeur de retour : True si la case est spéciale sinon False

Background.is_case_special_activated(b, location, k) → Booléen

Description : Indique si Knil se trouve sur une case 'special'

Paramètres :

b : Background

location : entier, entier

k : Knil

Valeur de retour : True si la case est spéciale sinon False

Background.getCase(b, x, y) → dict

Description : Renvoie le dictionnaire associé à la case

Paramètres :

b : Background

x : entier

y : entier

Valeur de retour : Case

Background.getFontcolor(case, with_brightness=True) → dict

Description : Renvoie la couleur de la police sur la case

Paramètres :

case : Case

with_brightness : Booléen

Valeur de retour : Case

Background.getBgcolor(case, b=None, with_brightness=True) → dict

Description : Renvoie la couleur de fond de la case

Paramètres :

case : Case

b : Background

with_brightness : Booléen

Valeur de retour : Case

4.6 Map.py

Map.create(xml_file_path) → dict

Description : Crée la grille de dictionnaire à partir d'informations dans un fichier xml, chaque type de bâtiment seront ajoutés à l'aide de fonctions extérieurs.

Paramètres :

xml_file_path : chaîne

Valeur de retour : Dictionnaire associé à la case

Map.case_to_dict(c) → dict

Description : Crée une case

Paramètres :

c : element

Valeur de retour : Case

Map.getCase(grid, x, y) → dict

Description : Renvoie la case associée

Paramètres :

grid : map

x : entier

y : entier

Valeur de retour : Case

Map.setCase(grid, x, y, case_value, force=False) → grid

Description : Applique une case aux coordonnées associées dans la grille en tenant compte du type de case (traversable ou non) si précisé

Paramètres :

grid : map

x : entier

y : entier

case_value : Case

Valeur de retour : grid

Map.grid_append_road(grid, r, values) → grid

Description : Ajoute une structure de type route à la grille

Paramètres :

grid : map

r : element

values : element

Valeur de retour : grid

Map.grid_append_house(grid, h, *args) → grid

Description : Ajoute une structure de type maison à la grille

Paramètres :

grid : map

r : element

values : element

Valeur de retour : grid

Map.grid_append_room(grid, room, origin, nc, wc) → grid

Description : Ajoute recursivement une salle dans la salle supérieure, les coordonnées de la salle sont relatives à celles supérieures (ou à la maison si primaires)

Paramètres :

grid : map
room : element
origin : entier, entier
nc : Case
wc : Case

Valeur de retour : grid

Map.grid_append_tree(grid, t, values) → grid

Description : Ajoute une structure de type arbre à la grille

Paramètres :

grid : map
t : element
values : element

Valeur de retour : grid

Map.grid_append_area(grid, a) → grid

Description : Change une zone de la grille avec un même type de Case

Paramètres :

grid : map
a : element

Valeur de retour : grid

Map.grid_append_rect(grid, origin, size, walls, normal_case, wall_case=None, force_normal_value=False, force_wall_value=False) → grid

Description : Construit un rectangle de Case en prenant en compte les murs

Paramètres :

grid : map
origin : entier, entier
size : entier, entier
walls : entier, entier, entier, entier
normal_case : Case
wall_case : Case
force_normal_value : Booléen
force_wall_value : Booléen

Valeur de retour : grid

4.7 Menu.py

Menu.create(grid, x0, y0, case_xmax, case_ymax, dx=0, dy=0, framing=True, auto_clear="zone", text_align="center") → grid

Description : Crée un menu

Paramètres :

- grid : grille de tuple(chaîne, fonction)
- x0 : entier
- y0 : entier
- case_xmax : entier
- case_ymax : entier
- dx : entier
- dy : entier
- framing : Booléen
- auto_clear : chaîne
- text_align : chaîne

Valeur de retour : Menu

Menu.move(m, steer) → rien

Description : Déplace le curseur dans le menu

Paramètres :

- m : Menu
- steer : chaîne

Valeur de retour : aucun

Menu.show_all(m) → rien

Description : Affiche le menu en entier

Paramètres :

- m : Menu

Valeur de retour : aucun

Menu.show_cursor_framing(m, *args) → rien

Description : Affiche la case sélectionnée par le curseur

Paramètres :

- m : Menu
- args : (argument qui sert à stocker les arguments inutiles)

Valeur de retour : aucun

Menu.display_frame(x, y, cx, cy, ftype) → rien

Description : Affiche une case du menu

Paramètres :

- x : entier
- y : entier
- cx : entier
- cy : entier
- ftype : liste de caractères

Valeur de retour : aucun

Menu.erase_case(m, *args) → rien

Description : Réaffiche le contour de base d'une case.

Paramètres :

m : Menu

args : (argument qui sert à stocker les arguments inutiles)

Valeur de retour : aucun

Menu.validate(m) → rien

Description : Appelle la fonction associée à la case

Paramètres :

m : Menu

Valeur de retour : aucun

Menu.clear_zone(m) → rien

Description : Efface une zone

Paramètres :

m : Menu

Valeur de retour : aucun

Menu.run(m) → rien

Description : Lance la boucle de simulation du menu

Paramètres :

m : Menu

Valeur de retour : aucun

4.8 Item.py

Item.summon_item(item_name) → Item

Description : crée un dictionnaire correspondant à un item

Paramètres :

item_name : chaîne

Valeur de retour : Item

Item.exe(item) → rien

Description : Exécute la fonction associée à l'objet

Paramètres :

item : Item

Valeur de retour : aucune

Item.exist(item_name) → Booléen

Description : Vérifie si l'objet existe

Paramètres :

item_name : chaîne

Valeur de retour : True si l'objet existe sinon False

Item.getName(item) → chaîne

Description : Renvoie le nom de l'objet

Paramètres :

item : Item

Valeur de retour : Nom de l'objet

Item.getItem(item_name) → Item

Description : Retrouve l'objet à partir du nom

Paramètres :

item_name : chaîne

Valeur de retour : Item

Item.getShape(item) → chaîne

Description : Renvoie le dessin de l'objet (stocké dans un fichier texte)

Paramètres :

item : Item

Valeur de retour : caractères utilisé pour l'affichage

Il y aura une fonction associé à l'exécution de chaque objet

4.9 Quest.py

Quest.init(xml_file_path) → Quest

Description : crée le dictionnaire contenant les quêtes

Paramètres :

xml_file_path : chaîne

Valeur de retour : Quest

Quest.check_quest(q, event, k) → chaîne, chaîne

Description : Vérifie si une quête est en attente de l'événement si oui, avance la quête

Paramètres :

q : struct de Quest

event : chaîne

k : Knil

Valeur de retour : tuple(chaîne qui contient des informations associées à la quête qui attendait l'événement , résultat de l'exécution de l'événement)

Quest.auto_txt(q, pnj) → chaîne, entier

Description : Retrouve le texte que le pnj doit dire si il n'est associé à aucune quête

Paramètres :

q : struct de Quest

pnj : chaîne

Valeur de retour : tuple(nom de la quête, numéro associé à l'étape de la quête)

Quest.details(q, quest, d) → rien

Description : Lance une boîte de dialogue affichant les détails de la quête

Paramètres :

q : struct de Quest

quest : chaîne

d : Dialog_box

Valeur de retour : aucun

Quest.run_quest_menu(q, d) → rien

Description : Affiche le journal des quêtes

Paramètres :

q : struct de Quest

d : Dialog_box

Valeur de retour : aucun

Quest.run_quest_log(q, d, detail=true) → rien

Description : Crée la grille correspondant au menu du journal des quêtes

Paramètres :

q : struct de Quest

d : Dialog_box

detail : Booléen

Valeur de retour : aucun

Quest.isStarted(q, questname) → Booléen

Description : Vérifie si la quête est lancée

Paramètres :

q : struct de Quest

quest_name : chaîne

Valeur de retour : Booléen

Quest.isFinished(q, quest_name) → Booléen

Description : Vérifie si la quête est finie

Paramètres :

q : struct de Quest

quest_name : chaîne

Valeur de retour : Booléen

Quest.finish_quest(q, quest_name) → rien

Description : Change l'état de la quête à « finished »

Paramètres :

q : struct de Quest

quest_name : chaîne

Valeur de retour : aucun

Quest.getDetails(q, quest_name) → chaîne

Description : Donne les détails concernants la quête

Paramètres :

q : struct de Quest

quest_name : chaîne

Valeur de retour : détails concernants la quête

4.10 Event.py

Event.create() → rien

Description : crée le dictionnaire contenant les événements

Paramètres :

Valeur de retour : aucun

Event.add(event, dest) → rien

Description : Ajoute un destinataire en l'attente d'un événement à cette événement

Paramètres :

event : chaîne

dest : chaîne

Valeur de retour : aucun

Event.remove(event, dest) → rien

Description : Enlève un destinataire d'un événement

Paramètres :

event : chaîne

dest : chaîne

Valeur de retour : aucun

Event.generate(event, q, k) → chaîne ou rien

Description : génère un événement

Paramètres :

event : chaîne

q : struct de Quest

k : Knil

Valeur de retour :chaîne qui contient des informations associées à la quête qui attendait l'événement ou ne retourne rien si ça ne correspond à aucun événement

Event.execute(execution, k) → rien

Description : Exécute la fonction associée à la résolution d'un événement

Paramètres :

execution : chaîne

k : Knil

Valeur de retour : aucun

4.11 Flashlight.py

Flashlight.create(scope, alpha, delta) → Struct

Description : Crée le dictionnaire associé à la lampe

Paramètres :

scope : entier

alpha : float

delta : float

Valeur de retour : Flashlight

Flashlight.exe(flashlight) → rien

Description : Change l'état de la lampe

Paramètres :

flashlight : Flashlight

Valeur de retour : aucun

Flashlight.get_highlighted_case_pos(flashlight, grid, x, y) → set

Description : Crée le set des coordonnées des cases étant éclairées

Paramètres :

flashlight : Flashlight

grid : grille de Case

x : entier

y : entier

Valeur de retour : set des coordonnées

4.12 Monster.py

Monster.create(xml_file_path) → dict

Description : Crée le dictionnaire contenant les différents monstres.

Paramètres :

xml_file_path : chaîne

Valeur de retour : dict de Monster

Monster.is_Knil_in_sight(monster, k, map) → Booléen

Description : indique si le monstre a une vision sur Knil

Paramètres :

monster : Monster

k : Knil

map : grille de Case

Valeur de retour : True si Knil est en vue sinon False

Monster.move(m, k, b) → rien

Description : gère le mouvement du monstre si Knil n'est pas en vue, le monstre èrre dans le donjon

Paramètres :

m: Monster

k : Knil

b: Background

Valeur de retour : aucune

Monster.kill_Knil(k, d) → rien

Description : Fonction qui « tue » Knil, le joueur reviendra au village

Paramètres :

k : Knil

d : Dialog_box

Valeur de retour : aucune

Monster.show(monsters, k ,b) → rien

Description : Affiche les monstres

Paramètres :

monsters : dict de Monster

k : Knil

b: Background

Valeur de retour : aucune

Monster.wander(m, b) → rien

Description : gère l'errance des monstres

Paramètres :

m : Monster

b: Background

Valeur de retour : aucune

Monster.is_Knil_in_room(m, k ,b) → Booléen

Description : Indique si Knil est dans la salle avec le monstre

Paramètres :

m: Monster

k : Knil

b: Background

Valeur de retour : True si Knil est dans la même salle sinon False

4.13 Tools.py

Tools.Read est une classe (un objet) qui permet d'effectuer la lecture des entrées clavier, dans l'entrée standard, en parallèle du jeu. Son utilisation est indispensable ici en raison de l'accumulation des entrées avec la méthode du select.select() lors d'un appui prolongé sur une touche.
Nous avons donc trois fonctions associées à cette classe :

Read.run(self) → rien
- Description : Boucle de simulation de Read, lis les entrées dans l'entrée standard
- Paramètres :
  - self : Read
- Valeur de retour :aucun

Read.stop(self) → rien
- Description : Arrête la boucle de lecture
- Paramètres :
  - self : Read
- Valeur de retour :aucun

Read.read(self) → caractères
- Description : retour la dernière entrée
- Paramètres :
  - self : Read
- Valeur de retour :entrée clavier

Ainsi que 3 fonctions permettant d'y acceder :

Tools.startReadingData() → rien
- Description : Appelle Read.run en parallèle du jeu par la méthode Thread.start()
- Paramètres :
- Valeur de retour :aucun

Tools.stopReadingData() → rien
- Description : Appelle Read.stop
- Paramètres :
- Valeur de retour : aucun

Tools.readData() → caractères
- Description : Renvoie la
- Paramètre :
- Valeur de retour : entrée clavier

Tools.interact(dest, c, *arg_func, **func) → fonction

Description : Exécute les fonctions respectivement associées aux touches. Les fonctions, l'association et les touches sont donnés en paramètres via les arguments *arg_func **func

Paramètres :

dest : dict

c : caractères

arg_func : liste de tuple de (tuple de chaîne, fonction ou un tuple de fonctions)

func : dict dont les clés sont les touches et dont les valeurs sont les fonctions à exécuté correspondantes aux touches

Valeur de retour : valeur de retour de la fonction exécutées ou si plusieurs fonctions exécutées, liste de valeurs de retour des fonctions exécutées

Tools.xpath(element, path) → élément

Description : Permet d'accéder récursivement à un élément en indiquant son chemin à partir d'un élément

Paramètres :

element : élément

path : chaîne

Valeur de retour : élément

Tools.xml_to_dict(element) → dict

Description :Crée récursivement un dictionnaire à partir d'un fichier xml

Paramètres :

element : element

Valeur de retour : dict organisé comme dans le fichier xml

Tools.str_to_tuple(string, max_len_str=None) → tuple

Description : à partir d'un chaîne un tuple d'élément où chaque élément (qu'on détermine par un séparation avec la virgule)sont évalué. Si il n'y qu'un seule élément alors retourne directement l'élément.

Si l'évaluation est impossible on garde l'élément sous la forme d'une chaîne de caractères d'une taille déterminé.

Paramètres :

string : chaîne

max_len_str : entier

Valeur de retour : tuple