

# ÉCOLE NATIONALE DES INGÉNIEURS DE BREST

DOCUMENT DE CONCEPTION

MDD-PROJET

---

## Spazz

---

Noé MAILLARD et Allan DANO

26 mai 2014

Version 1.1

## Table des matières

<b>1</b>	<b>Rappel du cahier des charges</b>	<b>3</b>
1.1	Contraintes techniques . . . . .	3
1.2	Fonctionnalités . . . . .	3
1.3	Prototype P1 . . . . .	3
<b>2</b>	<b>Principe des solutions techniques</b>	<b>3</b>
2.1	Langage . . . . .	3
2.2	Architecture du logiciel . . . . .	3
2.3	Interface utilisateur . . . . .	3
2.3.1	Boucle de simulation . . . . .	4
2.3.2	Images ASCII-Art . . . . .	4
<b>3</b>	<b>Analyse</b>	<b>5</b>
3.1	Analyse noms/verbes . . . . .	5
3.2	Types de Donnée . . . . .	5
3.3	Dépendance entre modules . . . . .	6
3.4	Analyse descendante . . . . .	6
3.4.1	Arbre principal . . . . .	6
3.4.2	Arbre affichage . . . . .	6
3.4.3	Arbre interaction . . . . .	6
<b>4</b>	<b>Description des fonctions</b>	<b>7</b>
4.1	Programme principal : Main.py . . . . .	7
4.2	Module Game.py . . . . .	8
4.3	Module Menu.py . . . . .	11
4.4	Module Level.py . . . . .	12
4.5	Module Snake.py . . . . .	14
4.6	Module HighScores.py . . . . .	15
4.7	Module Editor.py . . . . .	15

# 1 Rappel du cahier des charges

## 1.1 Contraintes techniques

- Le logiciel crée est évalué par les professeurs sur un ordinateur de salle de TP, il faut donc que le jeu s'exécute et soit jouable sur ces machines
- Le cours porte sur le langage Python, il est donc évident que le jeu soit écrit en Python
- Le paradigme utilisé est celui de la programmation procédurale
- L'interface doit être en mode texte dans le terminal

## 1.2 Fonctionnalités

**F1** : Choisir un pseudo

**F2** : Choisir la difficulté

**F3** : Choisir le niveau

**F4** : Jouer un niveau

**F4.1** : Afficher le Jeu

**F4.2** : Changer de direction

**F4.3** : Ramasser un jeton

**F4.4** : Finir le niveau

**F5** : Afficher les meilleurs scores

**F6** : Quitter le jeu

**F7** : Créer des niveaux

## 1.3 Prototype P1

Ce prototype porte sur la création du jeu et la possibilité de changer les paramètres du jeu.

Mise en œuvre de fonctionnalités : F1, F2, F3, F4.1-3, F6

# 2 Principe des solutions techniques

## 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 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.

### 2.3.1 Boucle de simulation

Le programme mettra en oeuvre une boucle de simulation qui gèrera l'affichage et et les événements clavier.

### 2.3.2 Images ASCII-Art

Pour stocker les niveaux du jeu nous utilisons des images ascii stockées dans des fichiers textes

## 3 Analyse

### 3.1 Analyse noms/verbes

Verbes :

nommer, choisir, jouer, afficher, déplacer, finir, quitter

Noms :

joueur, Spazz, pseudo, direction, niveau, score, taille, position

### 3.2 Types de Donnée

```
type : Game = struct
    level          : Level
    spazz          : Snake
    menu           : Menu
    state          : chaine de caracteres
    difficulty     : entier
```

```
type : Level = struct
    allLevels      : liste de chaines de caractere
    level          : liste de liste de chaine de caracteres
    levelNumber    : entier
```

```
type : Menu = struct
    items          : liste de chaine de caracteres
    cursor         : entier
    selectedItem   : entier
```

### 3.3 Dépendance entre modules

### 3.4 Analyse descendante

#### 3.4.1 Arbre principal

```
Main.main()  
  +-- Main.init()  
  |      +-- Level.create()  
  |      +-- Menu.create()  
  |      +-- Game.create()  
  |  
  +-- Main.run()  
      +-- Main.show()  
      +-- Main.interact()
```

#### 3.4.2 Arbre affichage

```
Main.show()  
  +-- Menu.show()  
  +-- Game.show()
```

#### 3.4.3 Arbre interaction

```
Main.interact()  
  +-- Menu.interact()  
  +-- Game.interact()
```

## 4 Description des fonctions

### 4.1 Programme principal : Main.py

- **Main.init()**
- **Main.run(game)**
- **Main.show(game)**
- **Main.interact(game)**

**Main.init()** -> rien

Description : initialisation des paramètres du jeu  
Paramètres : aucun  
Valeurs de retour : aucune

**Main.run(game)** -> rien

Description : boucle de simulation  
Paramètres :  
    game : Game  
Valeurs de retour : aucune

**Main.show(game)** -> rien

Description : affiche le jeu  
Paramètres :  
    game : Game  
Valeurs de retour : aucune

**Main.interact(game)** -> rien

Description : gere les action de l'utilisateur  
Paramètres :  
    game : Game  
Valeurs de retour : aucune

**Main.quit(game)** -> rien

Description : quitte le jeu  
Paramètres :  
    game : Game  
Valeurs de retour : aucune

## 4.2 Module Game.py

- `Game.create(menu, level, win, state, name, difficulty)`
- `Game.show(game)`
- `Game.interact(game)`
- `Game.quitGame(status, game)`
- `Game.getMenu(game)`
- `Game.getLevel(game)`
- `Game.setLevel(level, game)`
- `Game.getSnake(game)`
- `Game.setSnake(snake, game)`
- `Game.showFood(food, win)`
- `Game.getFood(game)`
- `Game.foodEaten(snake, food)`
- `Game.getState(game)`
- `Game.setState(state, game)`
- `Game.getWin(game)`
- `Game.getName(game)`
- `Game.setName(name, game)`
- `Game.askName(game)`
- `Game.getDifficulty(game)`
- `Game.setDifficulty(difficulty, game)`
- `Game.askDifficulty(game)`

`Game.create(menu, level, win, state, name, difficulty)` -> Game

Description : crée une nouvelle partie

Parametres :

menu	: Menu
level	: Level
win	: fenetre curses
state	: chaine de caractères
name	: chaine de caractères
difficulty	: entier

Valeurs de retour : nouvelle partie en fonction des parametres

`Game.show(game)` -> rien

Description : Affiche la partie

Parametres :

game : Game

Valeurs de retour : aucune

`Game.interact(game)` -> rien

Description : Calcule les differentes variables du jeu.

Parametres :

game : Game

Valeurs de retour : aucune



**Game.quitGame(status, game)** ->rien

Description : Change l'état du jeu et donc quitte la partie en cours

Parametres :

status : String

game : Game

Valeurs de retour : aucune

**Game.getMenu(game)** ->menu

Description : Retourne le menu de la variable game

Parametres :

game : Game

Valeurs de retour : returnValue

**Game.getLevel(game)** ->Level

Description : retourne le level contenu dans le game

Parametres :

game : Game

Valeurs de retour : level

**Game.setLevel(level, game)** ->Game

Description : change le level contenu dans le game

Parametres :

level : Level

game : Game

Valeurs de retour : game

**Game.getSnake(game)** ->Snake

Description : retourne le snake stocke dans le game

Parametres :

game : Game

Valeurs de retour : Snake

**Game.setSnake()** ->Game

Description : Change la variable snake stockée dans le game

Parametres :

snake : Snake

game : Game

Valeurs de retour : variable game modifiée

**Game.showFood()** ->rien

Description : affiche le food

Parametres :

food : liste  
win : fenetre curses  
Valeurs de retour : aucune

Game.**getFood()** ->liste

Description : retourne le food stocké dans le game

Parametres :

game : Game

Valeurs de retour : liste des coordonnées du food

Game.**setNewFood()** ->game

Description : change les coordonnées du food aléatoirement

Parametres :

game : Game

Valeurs de retour : variable game modifiée

Game.**foodEaten()** ->booléen

Description : détermine si le snake a ramassé un jeton

Parametres :

snake : Snake

food : liste

Valeurs de retour : booléen

Game.**getState()** ->chaîne de caractère

Description : retourne l'état du jeu

Parametres :

game : Game

Valeurs de retour : variable state contenue dans la variable game

Game.**setState()** ->game

Description : change l'état du jeu

Parametres :

state : String

game : Game

Valeurs de retour : variable game modifiée

Game.**getWin(game)** ->Window

Description : retourne la fenêtre curses du game

Parametres :

game : Game

Valeurs de retour : variable win contenue dans la variable game

**Game.getName(game)** -> chaîne de caractères  
 Description : retourne le nom du joueur  
 Paramètres :  
     game : Game  
 Valeurs de retour : variable name contenue dans la variable game

**Game.setName(name, game)** -> Game  
 Description : change le nom du joueur  
 Paramètres :  
     name : chaîne de caractères  
     game : Game  
 Valeurs de retour : Game

**Game.askName(game)** -> chaîne de caractère  
 Description : demande le nom du joueur à l'utilisateur  
 Paramètres :  
     game : Game  
 Valeurs de retour : chaîne entrée par l'utilisateur

**Game.getDifficulty(game)** -> entier  
 Description : retourne la difficulté  
 Paramètres :  
     game : Game  
 Valeurs de retour : variable difficulté de la variable game

**Game.setDifficulty(difficulty, game)** -> Game  
 Description : change la variable difficulty dans le game  
 Paramètres :  
     difficulty : entier  
     game : Game  
 Valeurs de retour : game

**Game.askDifficulty()** -> entier  
 Description : demande la difficulté à l'utilisateur  
 Paramètres :  
     game : Game  
 Valeurs de retour : difficulté choisie par l'utilisateur

### 4.3 Module Menu.py

- **Menu.create(\*argv)**
- **Menu.show(game)**
- **Menu.interact(game)**
- **Menu.getNumberOfMenuItems(menu)**
- **Menu.quit(game)**

**Menu.create()** -> Menu

Description : crée le menu selon un nombre variable d'argument

Parametres :

argv : liste de chaine de caractères

Valeurs de retour : menu

**Menu.show(game)** -> rien

Description : affiche le menu

Parametres :

game : Game

Valeurs de retour : aucune

**Menu.interact(game)** -> rien

Description : interagit avec l'utilisateur

Parametres :

game : Game

Valeurs de retour : aucune

**Menu.getNumberOfMenuItems(menu)** -> entier

Description : retourne le nombre d'items que le menu inclut

Parametres :

menu : Menu

Valeurs de retour : nombre l'items dans le Menu

**Menu.quit()** -> game

Description : quitte le menu et autorise le début du jeu

Parametres :

game : Game

Valeurs de retour : game dont le state à été changé

## 4.4 Module Level.py

- **Level.create(levelNumber, levelFile)**
- **Level.getMap(level)**
- **Level.getLevelNumber(level)**
- **Level.setLevelNumber(levelNumber, level)**
- **Level.askLevelNumber(game)**
- **Level.getNumberOfLevels(level)**
- **Level.show(level, win)**

**Level.create(levelNumber, levelFile)** -> Level

Description : crée la variable abstraite de type Level

Parametres :  
    levelNumber : entier  
    levelFile : chaine de caractères  
Valeurs de retour : variable de type Level

**Level.getMap(level)** -> liste de liste de caractere  
Description : retourne la carte du niveau contenue dans la variable le  
Parametres : level : Level  
Valeurs de retour : carte

**Level.getLevelNumber(level)** -> entier  
Description : retourne le numero du niveau  
Parametres :  
    level : Level  
Valeurs de retour : variable levelNumber contenue dans level

**Level.setLevelNumber(number, level)** -> Level  
Description : change le numéro du niveau de la variable level  
Parametres :  
    levelNumber : entier  
    level : Level  
Valeurs de retour : level

**Level.askLevelNumber(game)** -> entier  
Description : demande le numéro du niveau à l'utilisateur  
Parametres :  
    game : Game  
Valeurs de retour : numero du niveau demandé par l'utilisateur

**Level.getNumberOfLevels(level)** -> entier  
Description : retourne le nombre total de niveaux  
Parametres :  
    level : Level  
Valeurs de retour : nombre de niveaux dans la variable level

**Level.show(level, win)** -> rien  
Description : affiche la carte du niveau en cours  
Parametres :  
    level : Level  
    win : fenetre curses  
Valeurs de retour : rien

## 4.5 Module Snake.py

- `Snake.create(headXPos, headYPos, headFacing, length)`
- `Snake.show(snake, win)`
- `Snake.computeNextPos(key, snake, win)`
- `Snake.getHeadX(snake)`
- `Snake.getHeadY(snake)`
- `Snake.reset()`

`Snake.create(headXPos, headYPos, headFacing, length) -> Snake`

Description : crée la variable snake

Parametres :

headPosX : entier

headPosY : entier

Valeurs de retour : variable Snake

`Snake.show(snake, win) -> rien`

Description : Affiche le snake dans la fenetre

Parametres :

snake : Snake

win : fenetre curses

Valeurs de retour : aucune

`Snake.computeNextPos(key, snake, win) -> snake`

Description : calcule la prochaine position du snake

Parametres :

key : entier

snake : Snake

food : liste d'entiers

win : fenetre curses

Valeurs de retour : la variable snake modifiée

`Snake.getHeadX() -> entier`

Description : retourne la position en X de la tete du snake

Parametres :

snake : Snake

Valeurs de retour : position en x de la tete

`Snake.reset() -> Snake`

Description : remet la variable snake a son etat au debut du jeu

Parametres :

snake : Snake

Valeurs de retour : nouveau snake

## 4.6 Module HighScores.py

- `HighScores.log(score, name, difficulty)`
- `HighScores.get()`
- `HighScores.show(highScores, win)`

`HighScores.log(score, name, difficulty)` → `highScores`  
Description : sauvegarde les meilleurs scores  
Parametres :  
    score : entier  
    name : chaine de caracteres  
    difficulty : entier  
Valeurs de retour : nouveaux `highScores`

`HighScores.get()` → `highScores`  
Description : charge les `highScores` contenus dans les fichiers `.dat`  
Parametres :  
    aucun  
Valeurs de retour : `highScores`

`HighScores.show(highScores, win)` → rien  
Description : affiche les meilleurs scores  
Parametres :  
    highScores : `highScores`  
    win : fenetre curses  
Valeurs de retour : aucune

## 4.7 Module Editor.py

- `Editor.init()`
- `Editor.loadNew()`
- `Editor.showLevel(win, level)`
- `Editor.action(win, level, key, trace)`
- `Editor.changeAllTiles(win, level)`
- `Editor.saveLevel(level)`
- `Editor.quit(win)`
- `Editor.start()`

`Editor.init()` → fenetre curses  
Description : Initialise une nouvelle fenetre curses pour l'editeur de  
Parametres :  
    aucun  
Valeurs de retour : aucune

`Editor.loadNew()` → liste de liste de caracteres  
Description : cree un niveau avec un cadre

Parametres :  
    aucun  
Valeurs de retour : level

Editor.**showLevel(win, level)**->rien  
Description : Affiche le niveau que l'on edite  
Parametres :  
    win : fenetre curses  
    level : liste de liste de caracteres  
Valeurs de retour : aucune

Editor.**action(win, level, key, trace)**->rien  
Description : deplace le curseur et change le caractetre a l'endroit o  
Parametres :  
    win : fenetre curses  
    level : liste de liste de caracteres  
    key : entier  
    trace : booleen  
Valeurs de retour : aucune

Editor.**changeAllTiles(win, level)**->rien  
Description : met a jour les caracteres du niveau  
Parametres :  
    win : fenetre curses  
    level : liste de liste de caracteres  
Valeurs de retour : aucune

Editor.**saveLevel(level)**->rien  
Description : sauvegarde le niveau a la suite des autres niveaux dans  
Parametres :  
    level : liste de liste de caracteres  
Valeurs de retour : aucune

Editor.**quit(win)**->rien  
Description : quitte l'éditeur  
Parametres :  
    win : fenetre curses  
Valeurs de retour : aucune

Editor.**start()**->rien  
Description : Lance l'editeru de jeu  
Parametres :  
    aucun



Valeurs de retour : aucune