

# ÉCOLE NATIONALE DES INGÉNIEURS DE BREST

DOCUMENT DE CONCEPTION

MDD-PROJET

---

## Spazz

---

Noé MAILLARD et Allan DANO

Date

Version 1.0

# Table des matières

<b>1</b>	<b>Rappel du cahier des charges</b>	<b>2</b>
1.1	Contraintes techniques . . . . .	2
1.2	Fonctionnalités . . . . .	2
1.3	Prototype P1 . . . . .	2
<b>2</b>	<b>Principe des solutions techniques</b>	<b>2</b>
2.1	Langage . . . . .	2
2.2	Architecture du logiciel . . . . .	2
2.3	Interface utilisateur . . . . .	2
2.3.1	Boucle de simulation . . . . .	3
2.3.2	Images ASCII-Art . . . . .	3
<b>3</b>	<b>Analyse</b>	<b>4</b>
3.1	Analyse noms/verbes . . . . .	4
3.2	Types de Donnée . . . . .	4
3.3	Dépendance entre modules . . . . .	5
3.4	Analyse descendante . . . . .	5
3.4.1	Arbre principal . . . . .	5
3.4.2	Arbre réglages . . . . .	5
3.4.3	Arbre affichage . . . . .	5
3.4.4	Arbre interaction . . . . .	5
<b>4</b>	<b>Description des fonctions</b>	<b>6</b>
4.1	Programme principal : Main.py . . . . .	6
4.2	Module Game.py . . . . .	7
4.3	Module Settings.py . . . . .	8
4.4	Module Level.py . . . . .	8
4.5	Module Coin.py . . . . .	9

# 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** : Jouer un niveau

**F3.1** : Choisir le niveau

**F3.2** : Afficher le Jeu

**F3.3** : Changer de direction

**F3.4** : Ramasser un jeton

**F3.5** : Finir le niveau

**F3.5.1** : Afficher résultat

**F3.5.2** : Afficher les meilleurs scores

**F3.5.3** : quitter le jeu

## 1.3 Prototype P1

Ce prototype porte sur la création et l'affichage du niveau.  
Mise en œuvre de fonctionnalités : F1, F2, F3.2, F3.2

# 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 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
    score           : Stats

type : Level = struct
    levelNumber     : entier
    map             : liste de liste de caractere
    coin            : Coin

type : Snake = struct
    position        : tuple (entier, entier)
    facing          : tuple (entier, entier)
    speed           : entier
    growRate        : entier
    size            : entier
    color           : indefini

type : Settings = struct
    difficulty      : liste
    levelNumber     : entier
    name            : chaine

type : Stats = struct
    timeLeft       : entier
    score          : entier
    highScores     : liste de tuples (chaine, entier)

type : Coin = struct
    numberLeft     : entier
    position       : tuple(entier, entier)
```

### 3.3 Dépendance entre modules

### 3.4 Analyse descendante

#### 3.4.1 Arbre principal

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

#### 3.4.2 Arbre réglages

```
Settings.askSettings()  
  +-- Settings.askName()  
  +-- Settings.askDifficulty()  
  +-- Settings.askLevelNumber()
```

#### 3.4.3 Arbre affichage

```
Main.show()  
  +-- Game.show()  
      +-- Level.show()  
      +-- Stats.show()  
      +-- Snake.show()  
      +-- Coin.show()
```

#### 3.4.4 Arbre interaction

```
Main.interact()  
  +-- Game.play()  
  |      +-- Snake.play()  
  |      +-- Coin.spawn()  
  |      +-- Game.endGame()  
  |  
  +-- Main.quit()  
      +-- Stats.showHighScores()
```

## 4 Description des fonctions

### 4.1 Programme principal : Main.py

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

`Main.init()` -> rien

Description : initialisation

Parametres : aucun

Valeurs de retour : aucune

`Main.run()` -> rien

Description : boucle de simulation

Parametres : aucun

Valeurs de retour : aucune

`Main.show()` -> rien

Description : affiche le jeu

Parametres : aucun

Valeurs de retour : aucune

`Main.interact()` -> rien

Description : gere les action de l'utilisateur

Parametres : aucun

Valeurs de retour : aucune

`Main.quit()` -> rien

Description : sauvegarde les données et quitte le jeu

Parametres : aucun

Valeurs de retour : aucune

## 4.2 Module Game.py

- `Game.create(s)`
- `Game.show(g)`
- `Game.move(g)`
- `Game.play(g)`
- `Game.finishLevel(g)`

`Game.create(s)` -> Game

Description : crée une nouvelle partie

Parametres :

s : Settings

Valeurs de retour : nouvelle partie en fonction des parametres

`Game.show(g)` -> rien

Description : affiche le jeu

Parametres :

g : Game

Valeurs de retour : aucune

`Game.play(g)` -> rien

Description : le spazz se déplace dans le niveau, quand il touche un j

Parametres :

g : Game

Valeurs de retour : aucune

`Game.endGame(g)` -> Game

Description : si le joueur perd ou qu'il gagne, le niveau est terminé

Parametres :

g : Game

Valeurs de retour : le Game à la fin du niveau



### 4.3 Module Settings.py

- `Settings.askSettings()`
- `Settings.askName()`
- `Settings.askDifficulty()`
- `Settings.askLevelNumber()`

`Settings.askSettings()` -> Settings  
Description : initialise les réglages  
Parametres : aucun  
Valeurs de retour :

`Settings.askName()` -> chaîne  
Description : demande le nom de l'utilisateur  
Parametres : aucun  
Valeurs de retour : nom de l'utilisateur

`Settings.askDifficulty()` -> float  
Description : demande la difficulté (vitesse du spazz)  
Parametres : aucun  
Valeurs de retour : valeur de la vitesse

`Settings.askLevelNumber()` -> entier  
Description : demande le niveau à charger  
Parametres : aucun  
Valeurs de retour : numero du niveau à charger

### 4.4 Module Level.py

- `Level.create(s)`
- `Level.show(l)`

`Level.create(s)` -> liste de liste de caracteres  
Description : crée un niveau  
Parametres :  
    s : Settings  
Valeurs de retour : niveau avec les settings

`Level.show(l)` -> rien  
Description : affiche le niveau  
Parametres :  
    l : Level  
Valeurs de retour : aucune

## 4.5 Module Coin.py

- `Coin.spawn(g)`

`Coin.spawn(g)` -> rien

Description : si il n'y a pas de jeton sur le plateau, en créer un

Parametres :

`g` : Game

Valeurs de retour : aucune