

# 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>1</b>
1.1	Contraintes techniques . . . . .	1
1.2	Fonctionnalités . . . . .	1
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 . . . . .	2
2.3.2	Images ASCII-Art . . . . .	2
<b>3</b>	<b>Analyse</b>	<b>2</b>
3.1	Analyse noms/verbes . . . . .	2
3.2	Types de Donnée . . . . .	3
3.3	Dépendance entre modules . . . . .	3
3.4	Analyse descendante . . . . .	3
3.4.1	Arbre Principal . . . . .	3
3.4.2	Arbre Menu . . . . .	4
3.4.3	Arbre affichage . . . . .	4
3.4.4	Arbre interaction . . . . .	4
<b>4</b>	<b>Description des fonctions</b>	<b>5</b>
4.1	Programme principal : Main.py . . . . .	5

## 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** : Afficher les meilleurs scores

**F4** : Jouer un niveau

**F4.1** : Choisir le niveau

**F4.2** : Afficher le Jeu

**F4.3** : Changer de direction

**F4.4** : Ramasser un jeton

**F4.5** : Finir le niveau

**F4.5.1** : Afficher résultat

**F4.5.2** : Consulter les meilleurs scores

**F4.5.3** : afficher menu

### 1.3 Prototype P1

Ce prototype porte sur la création du menu et sur l'affichage du niveau.

Mise en œuvre de fonctionnalités : F1, F2, F3, F4.2.1, F4.2.4.1, F4.2.4.2, F4.2.4.3.

## 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 œuvre 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)
    direction       : tuple (entier, entier)
    speed           : entier
    growRate        : entier
    size            : entier
    color           : indefini

type : Menu = struct
    difficulty      : liste
    name            : chaine
    highScores      : liste de tuples (chaine, entier)

type : Stats = struct
    timeLeft       : entier
    score          : entier

type : Coin = struct
    number         : entier
    position       : tuple(entier, entier)

```

## 3.3 Dépendance entre modules

## 3.4 Analyse descendante

### 3.4.1 Arbre Principal

```

Main.main()
|
+-- Main.init()
|   |
|   +-- Menu.show()
|   +-- Menu.interact()
|
|
+-- Main.run()
    |
    +-- Main.show()
    +-- Main.interact()

```

### 3.4.2 Arbre Menu

```
Menu.interact()  
|  
+-- Menu.setDifficulty()  
|   |  
|   +-- Snake.setSpeed()  
|   +-- Snake.setGrowRate()  
|  
+-- Menu.setName()  
+-- Level.setUpLevel()  
    |  
    +-- Level.getNubmer()  
    +-- Level.create()
```

### 3.4.3 Arbre affichage

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

### 3.4.4 Arbre interaction

```
Main.interact()  
|  
+-- Snake.move()  
|   |  
|   +-- Snake.getPosition()  
|   +-- Snake.setPosition()  
|   +-- Snake.getDirection()  
|   +-- Snake.setDirection()  
|  
+-- Game.play()  
    |  
    +-- Level.play()  
    +-- Game.testVictory()
```

## 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

Parametres : aucun

Valeurs de retour : aucune

`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()` -> rein

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 scores et quitte le jeu

Parametres : aucun

Valeurs de retour : aucune