**P-Bulle**

**Thibaud Racine – CID2A**

**ETML Sébeillon**

**2023 - 2024**

**40 P**

**Table des matières**

[1. Spécifications 3](#_Toc150254827)

[1.1 Informations Générales 3](#_Toc150254828)

[1.2 Procédure 3](#_Toc150254829)

[1.3 Titre 3](#_Toc150254830)

[1.4 Sujet 3](#_Toc150254831)

[1.5 Matériel 3](#_Toc150254832)

[2. Cahier des charges 3](#_Toc150254833)

[1.6 Gestion de Projet 3](#_Toc150254834)

[1.7 Qualité 3](#_Toc150254835)

[1.8 Fonctionnalités requises 3](#_Toc150254836)

[3. UX UI – User experience, user interface 3](#_Toc150254837)

[1.9 Objectifs 3](#_Toc150254838)

[1.10 Interface 3](#_Toc150254839)

[1.11 Personas 3](#_Toc150254840)

[1.12 Palette de Couleurs 3](#_Toc150254841)

[1.13 Accecibilité 3](#_Toc150254842)

[1.14 Conclusion UX - UI 4](#_Toc150254843)

[4. POO – Programmation orienté objet 4](#_Toc150254844)

[1.15 Objectifs 4](#_Toc150254845)

[5. DB – Database 4](#_Toc150254846)

[1.16 Comment charger le dump de la DB dans MySQL ? 4](#_Toc150254847)

[1.17 Comment mettre en place les différents types d’users ? 4](#_Toc150254848)

[1.18 Gestion des Utilisateurs : 4](#_Toc150254849)

[1.19 Requêtes : 4](#_Toc150254850)

[Autres 4](#_Toc150254851)

[1.20 Conclusion 4](#_Toc150254852)

[1.21 Webographie 4](#_Toc150254853)

# Introduction

## Projet :

Le Projet P-Bulles nous demande de faire un réplica du célèbre jeu Snake à l’aide de Java Script afin de nous familisariser avec ce language. N’ayant aucune connaissances en Java Script, j’ai décidé de suivre un tutoriel sur Youtube pour ce projet.

# Code Snake

## Variables

|  |
| --- |
|  |

## Const & Let

|  |
| --- |
|  |

# Classes

## Apple.js

Cette classe en JavaScript, nommée "Apple" encapsule les fonctionnalités liées à la gestion de la pomme dans le jeu, comme l'initialisation, la génération aléatoire de position, le draw Fruit, et le log position.

**Constructeur :**

Le Constructeur initialise la taille de la pomme avec **unitSize**, la largeur du jeu avec **gameWidth**, et la hauteur du jeu avec **gameHeight**.  
La position de la pomme est définie de manière aléatoire en appelant la fonction createRandomPosition

**Méthode logPosition :**  
Affiche la position actuelle de la pomme dans la console.

**Méthode createRandomPosition :**  
Génère une position aléatoire pour la pomme à l'intérieur des limites du jeu et Utilise la fonction **randomNumber(min, max)** pour déterminer les coordonnées X et Y de la pomme.

**Méthode draw(ctx, color) :**  
Dessine la pomme graphiquement avec la couleur spécifiée.  
Utilise les coordonnées de la pomme et sa taille pour dessiner un rectangle

qui représente la pomme.  
Appelle la méthode **logPosition** pour afficher la position de la pomme.

**Méthode randomNumber(min, max) :**  
Génère un nombre aléatoire dans la plage spécifiée (min, max).

## Snake.js

Cette classe en JavaScript, appelée "Snake" encapsule les fonctionnalités liées à la gestion du Snake dans le jeu, y compris l'initialisation, le déplacement, le dessin sur le contexte graphique, et éventuellement la logique pour manger la nourriture.

**Constructeur :**  
Taille de chaque carré dans le Snake **unitSize** et crée le corps initial du serpent en appelant la méthode **createInitialSnake(initialLength)**.

**Méthode createInitialSnake(initialLength) :**  
Crée le corps initial du serpent.  
La longueur initiale du serpent est spécifiée par le paramètre **initialLength**.

**Méthode move(xVelocity, yVelocity) :**  
Déplace le serpent en mettant à jour les coordonnées de chaque partie du corps avec **xVelocity** et **yVelocity**.  
Ajoute une nouvelle tête à l'avant du corps du serpent.

**Méthode draw(ctx, snakeColor, snakeBorder) :**  
Dessine le serpent sur le contexte graphique (`ctx`).  
Chaque partie du corps du serpent est dessinée avec une couleur différente basée sur la teinte (`hue`).  
La bordure du serpent est également définie avec une couleur spécifiée (`snakeBorder`).

# Fonctions Principales

|  |
| --- |
| // Function to set the direction of the snake  function setDirection(newXVelocity, newYVelocity) {      xVelocity = newXVelocity;      yVelocity = newYVelocity;  } |

# Modules

## Import :

|  |
| --- |
|  |

## Export :

|  |
| --- |
|  |

# Concept de programation

## Génération aléatoire :

|  |
| --- |
|  |

## Diverses Structures de données :

|  |
| --- |
|  |

## Gestion Mémoire :

|  |
| --- |
|  |

## Commit Logs

|  |
| --- |
| logPosition() {          console.log(`Position de la pomme = X: ${this.position.x} | Y: ${this.position.y}`);      } |

# Autres

## Webographie

Vidéo : <https://www.youtube.com/watch?v=Je0B3nHhKmM>