1. **Démarche de conception**

Pour bien débuter la conception du projet, nous avons commencé à lister les différentes fonctionnalités demandées. Nous avons écarté huit propriétés demandées :

|  |  |
| --- | --- |
| **1** | Création du schéma relationnel de la base de données |
| **2** | Etude des réponses de l’API Open Food Facts |
| **3** | Création du découpage du programme |
| **4** | Création d’un jeu de donnée à importer dans la base de données local |
| **5** | Création de la base de données |
| **6** | Remplissage de la base avec le jeu de donnée créé |
| **7** | Créer les différentes requêtes sql |
| **8** | Créer l’affichage pour que l’utilisateur puisse interagir avec le programme |

Par la suite, nous avons ouvert le projet dans un répository git pour y mettre les différentes ressources du projet et effectuer les différentes sauvegardes dont nous aurons besoin. Ce repository se trouve via [ce lien](https://github.com/ZasshuNeko/-OC--Projet-n-3).

1. **Etude de l’algorithme**

Nous listons alors les étapes que nous allons devoir exécuter pour réaliser ce projet, nous étudierons ces étapes celons les interactions entre l’utilisateur et le programme :

1. **Initialisation du labyrinthe**
   1. Le labyrinthe dépend du fichier texte carte
   2. Ce fichier, une fois lu, est transféré dans une liste que l’on nommera « liste\_map »
   3. Cette liste sera alors transférée dans un dictionnaire formé ainsi : (tuple) : valeur, le tuple permet de générer l’emplacement de l’image (40px\*40px), nous créons lors de cette initialisation un dictionnaire d’objet que nous nommons « liste\_cle »
2. **Création du labyrinthe**
   1. Initialisation des images du labyrinthe : Sol, mur, porte
   2. Mise en place des images, utilisation de pygame pour mettre en place les images en utilisant la clé du dictionnaire
   3. Initialisation des images objets : *ether, aiguille, tube plastique, seringue*
   4. Mise en place des objets, utilisation de pygame pour mettre en place les images en utilisant la clé du dictionnaire « liste\_cle »
3. **Initialisation du personnage principal**
   1. Permet d’initialiser le symbole du personnage, utilise un module pour déterminer la position du personnage dans le niveau, initialisation de l’image et de sa position
4. **Utilisation des déplacements**
   1. La classe personnage est à l’écoute des évènements du clavier
   2. A chaque pression sur les flèches directionnelles, cela interpellera la classe personnage qui modifiera l’emplacement du personnage puis l’affichage se mettra à jour pour effectuer le déplacement du joueur.
5. **Ramasser un objet**
   1. Quand la position du personnage est identique à la position d’un objet, ce dernier passe d’un dictionnaire représentant les objets à ramasser dans un dictionnaire représentant les objets ramassés
   2. On interpelle la classe objet pour mettre à jour l’inventaire
6. **Terminer la partie**
   1. Quand le dictionnaire des objets ramassés est complet se rendre au niveau de l’ennemi et gagner la partie
   2. Si le joueur se rend vers l’ennemi sans tous les objets, il perd la partie

Cette liste formant l’algorithme du programme montre que nous aurons besoin d’un découpage spécifique :

Un fichier par classe, nous aurons donc « *Class\_Maze* » gérant le niveau, « *Class\_Personnage* » gérant le personnage, « *Class\_items* » gérant les objets. Nous gardons deux fichiers l’un étant le programme à démarrer que nous nommerons « *macgyvers\_main.py* » et le dernier fichier contient les modules spécifiques ne rentrant pas dans une classe que nous nommons « *module\_actions.py* » .

1. **Difficultés & solutions**

Les difficultés rencontrées lors du projet sont dues à la programmation autodidacte qui fut mise en place en premier lieu. La programmation procédurale qui me permit de finaliser l’exercice facilement montrât de nombreuses failles que ce soit dans l’évolution continue du programme, dans la lecture du code et son évolution.

Les difficultés furent donc de corriger ces aspects et de m’approprier un peu plus la programmation orientée objets. Dans un premier temps, il fallut regrouper les trop nombreux modules éparpillés et les fusionner dans des classes regroupant les grandes propriétés du projet (le niveau, le personnage, les objets). Dans un second temps, il fut nécessaire de corriger les éléments permettant une meilleure compréhension du code en retirant les « chiffres magiques » et en modifiant les noms de variables.