Compte-rendu

## But visé et moyen utilisé pour y parvenir

# → Séparer le travail

* Nous avons séparé le travail au début du projet. Nous pensions séparer le projet en deux parties distinctes : graphisme et code, que l'on associera au fur et à mesure. Cependant le groupe de graphisme ne possédait pas une partie qui contenait suffisamment de code pour l'épreuve du bac. Nous leurs avons alors donné des parts de code à faire comme \_\_\_.

# → Relier les travaux entre eux

* Afin de relier le travail des différentes personnes, nous nous sommes demandé comment nous pourrions communiquer de manière collective et rapide. Nous avons alors créé un groupe Facebook afin de partager nos idées ainsi qu'informer le groupe du travail fait ou à faire. Ensuite lorsque nous désirions travailler ensemble sur le code afin de s'entraider et donc d'avancer plus rapidement dans le projet, nous avons utilisé les conférences Skype. Enfin pour partager le code réalisé, nous voulions utiliser une *interface* nous permettant de partager ainsi que de fusionner les codes sources travaillés. Nous avons alors pensé à GitHub.

# → Partage et entraide

* Lorsque nous travaillons seule sur le code nous le sauvegardions dans GitHub, et nous expliquions par la suite aux autres membres du groupe ce que nous avons ajouté dans le programme.
* Aussi lorsque nous découvrions certains bugs dus à l'IDE (=environnement de développement) nous en informions les membres du groupe afin de ne pas nous arrêter et de rechercher à résoudre les bugs déjà résolus ultérieurement.

### Démarche suivie au cours du projet.

* Création de la page d’accueil (à développer : Problèmes page d'accueil : **Pygame**, pas de bouton dans pygame. **Tkinter**, affichage image de fond, arrêter la musique lorsque on clique sur la croix, relier page d'accueil et GameWindow, répéter la musique en continue)

INSERER LE DIAGRAMME DES CLASSES DU PROGRAMME

* Création de la classe Matrix :

→ Création de constantes de sorte à rendre le code plus visible : on comprend tout de suite quelle valeur lui a été donnée. On a décidé de les regrouper dans un fichier unique afin de les retrouver facilement.

→ On a décidé d’utiliser un tableau, la grid, car on voulait accéder à une case directement par un numéro de ligne et un numéro de colonne (ce que ne permettait pas une liste simplement). Celle-ci initialise à 1 la valeur des cases blocs.

→ On utilise la fonction random afin de remplir le tableau des cases vides de valeur 0. Au début du projet, nous avions initialisé le tableau avec des cases de valeur 0, puis lorsque nous devions rajouter les bordures du plateau, il nous a paru plus simple de faire le contraire.

→ Matrix charge les images utilisées pour les représentations des différents objets à afficher tant mobiles que statiques. Ensuite, grâce à la fonction PrintImages on affichera alors les images associées aux valeurs du tableau dans la fenêtre ouverte par GameWindow

* On crée la classe MatrixObjects qui est la classe générique d’un objet se plaçant sur le plateau.
* Man hérite de MatrixObjects : on crée des instances de Man (Bomby1 et Bomby2) dans Matrix. On utilisera alors dans Matrix la fonction random afin d’initialiser les positions des instances Bomber1 ou du Bomber2. Pour simplifier, nous n’avons créé que deux instances de Man, alors que la version originale prévoit jusqu’à 10 joueurs. Il aurait fallu plutôt travailler avec une liste d’instances plutôt que des instances nommées.
* Afin de gérer le déplacement des personnages, on teste dans un premier temps si la case ciblée est vide. Pour cela on utilise la fonction IsEmptyCase. Ensuite, on affecte au Bomberman concerné sa nouvelle position.
* On intercepte les évènements clavier afin de pouvoir mettre en œuvre le déplacement des deux personnages.
* La fenêtre prend en compte la réinitialisation du plateau par F5 et la fermeture de la page par ESC.
* La classe Bombe hérite de MatrixObjects : la position d’une bombe s’initialise en fonction de la position du personnage et de son sens au moment où la bombe est déposée. Bien sûr, la case de destination est vide.

→ On instaure une nouvelle fonction dans la classe Bombe qui est Explode, qui va appeler BombeExplodeAt présente dans Matrix. BombeExplodeAt va grâce à un timer fait disparaitre la bombe en réaffichant de nouveau la valeur 0 dans la grid.

* Gestion explosion. A faire