PROJET ISN

Bomberman

SOMMAIRE :

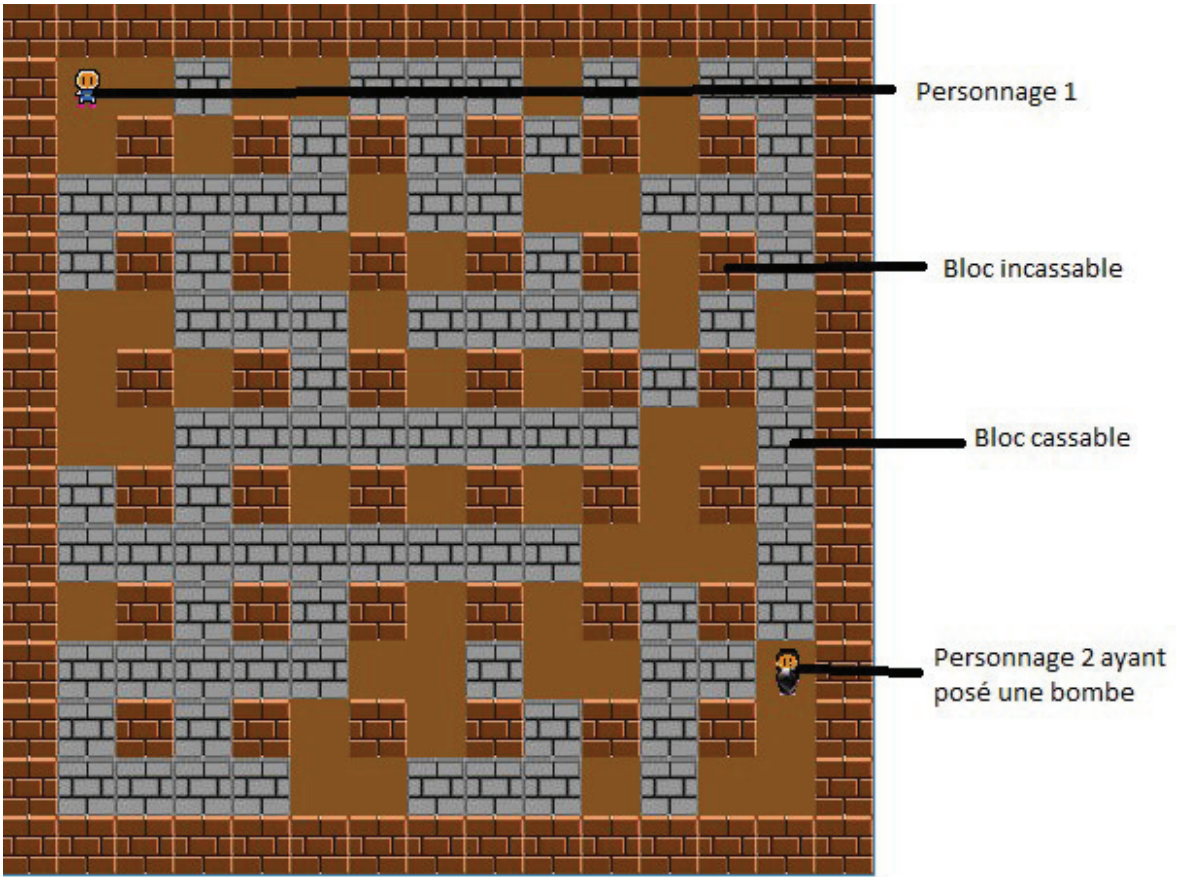
I) Introduction

Lors de notre année de Terminale S, Adrien BOIS, Amélie LEMOINE et moi-même avons programmé un jeu d'arcade/réflexe semblable au fameux « Bomberman » dans le cadre de notre projet d'ISN. Nous avons utilisé le langage de programmation Python et avons utilisé la bibliothèque PyGame et Numpy afin de mener à bien notre projet.
Nous avons rassemblé le programme et les images qu'il utilise dans un dossier appelé « Bomberman ».

II) Mode d'emploi

A) But du jeu

Le but du jeu est de faire exploser son adversaire à l'aide de bombes que l'on peut déposer sur l'emplacement de notre personnage. Cependant, des blocs obstruent le chemin des deux joueurs, leur permettant d'élaborer une stratégie afin de coincer leur opposant.



Il y a deux types de blocs : les blocs cassables et incassables (voir image)
Comme leur nom l'indique, ces blocs cassables peuvent être détruits à l'aide d'une bombe d'un des deux joueurs.

B) Contrôles

Notre jeu se joue à deux joueurs sur le même ordinateur.

- Joueur 1 (en haut à gauche) :

Z	déplacement vers le haut
S	déplacement vers le bas
Q	déplacement vers la gauche
D	déplacement vers la droite
Barre espace	Pose une bombe sur son emplacement

- Joueur 2 (en bas à droite) :

↑	déplacement vers le haut
↓	déplacement vers le bas
←	déplacement vers la gauche
→	déplacement vers la droite
Pavé Numérique 5	Pose une bombe sur son emplacement

III) Structure du projet

A) Répartition des tâches

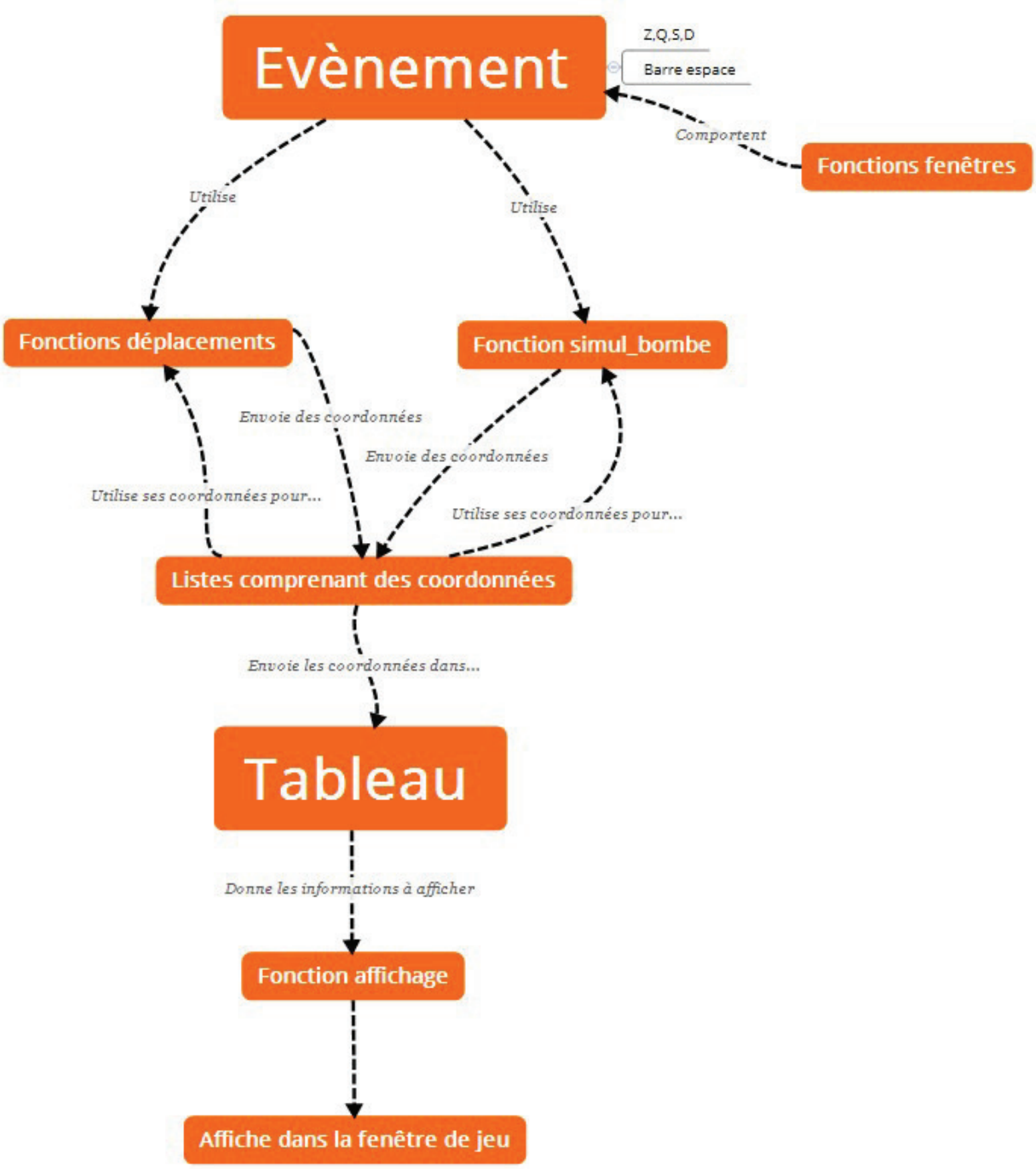
Nous avons, dès le début, réparti nos différentes tâches aux différents membres de groupe. Voici un tableau récapitulatif de la répartition des tâches.

Adrien	Benjamin	Amélie
-gestion des personnages -gestion des bombes	-gestion du temps -gestion des déplacements -gestion du placement des briques extérieures	-création des différentes fenêtres -gestion des briques incassables dans le labyrinthe
	-placement des briques cassables	

B) Structure globale du programme

- Tout d'abord, lorsque l'on exécute le programme, une fenêtre « menu » s'affiche et donne la possibilité à l'utilisateur de commencer à jouer, de lire des instructions ou de prendre connaissance des contrôles du jeu.
- Pour passer d'un menu à l'autre, l'utilisateur doit cliquer sur des boutons mis à disposition.
- Si l'utilisateur clique sur la croix, la fenêtre devrait se fermer. Le programme pourrait être encore amélioré à ce niveau-là. En effet, la fenêtre ne se ferme pas toujours après que l'utilisateur ait cliqué et il faut cliquer un certain nombre de fois avant que la fenêtre se ferme.
- Une fois le jeu commencé, les deux joueurs utilisent les commandes pour lancer des évènements :

  La fonction « d_(direction) » fais bouger le personnage vers un certaine direction (gauche,droite,bas,haut) grâce à un tableau. Par exemple, lorsque le joueur 1 appuie sur la touche « Q », la fonction « d_gauche » fait bouger le personnage vers la gauche dans un tableau qui est ensuite convertis en pixels puis les différents éléments du tableau sont affichés grâce à la fonction « affichage » qui est réutilisée continuellement afin d'actualiser à chaque évènements l'apparence graphique du jeu.
  La fonction « simul_bombe » va faire exploser les bombes posées par les joueurs après un certain temps et mettre fin au jeu si un joueur est touché grâce à un changement de fenêtre indiquant le joueur gagnant et offrant la possibilité à un joueur de relancer une partie en cliquant sur un bouton « rejouer ».
  Dans la page suivante, un schéma récapitulatif de l'organisation du programme :


Evènement
Z,Q,S,D
Barre espace
Comportent
Fonctions fenêtres
Utilise
Utilise
Fonctions déplacements
Fonction simul_bombe
Envoie des coordonnées
Envoie des coordonnées
Utilise ses coordonnées pour...
Utilise ses coordonnées pour...
Listes comprenant des coordonnées
Envoie les coordonnées dans...
Tableau
Donne les informations à afficher
Fonction affichage
Affiche dans la fenêtre de jeu

Vous remarquerez que le schéma ne parle pas des fonctions qui gèrent les fenêtres car elle n'ont pas de lien avec les autres éléments du schéma selon moi. En effet, ces fonctions ont pour but de proposer un certain nombre d'évènements possibles. Par exemple, dans la fenêtre de menu (gérée par la fonction « menu »), le joueur a la possibilité de cliquer sur deux boutons : Jouer (qui renvoie vers la fonction « main » qui lance le jeu) et Instructions (qui renvoie vers la fonction « instruc_fonctionmt » qui lance une fenêtre d'instruction.

C) Stratégie adoptée et description détaillée de la fonction déplacement

Nous avons commencé par importer toutes les bibliothèques dont nous avions besoin : numpy, pygame… etc.
Afin de gérer les différents éléments de notre jeu, nous avons utilisé un tableau (tab) de 15 x 15 cases que nous avons créé avec numpy.
Nous avons ensuité traduit les cases de ce tableau en pixels, étant donné que nous avions choisi une fenêtre de 450 pixels x 450 pixels, nous avons décidé de diviser les 15 x 15 cases du tableau en carrés de 30 pixels sur 30 pixels. De ce fait, chaque case correspondait à un élément du jeu d'une taille de 30 pixels sur 30 pixels. Nous avons donc décidé d'assimiler un chiffre à chaque élément du jeu :

- Le joueur 1 correspond au chiffre 1 du tableau ; le joueur 2 correspond au 2
- Une bombe est symbolisée par le chiffre 5
- Un joueur sur une bombe correspond dans le tableau à l'addition de son chiffre avec celui d'une bombe (par exemple, quand le personnage 1 pose une bombe, la case sur laquelle il se trouve possède la valeur 6)
- Un brique incassable par la fonction « simul_bombe » correspond à la valeur 8
- Une brique cassable par la fonction « simul_bombe » correspond à la valeur 9

Toutes les valeurs de ce tableau vont ensuite devoir être affichées : c'est le rôle de la fonction « affichage ». Cette fonction va « imprimer » (« blitter ») toutes les informations du tableau sur la fenêtre de jeu en associant les images chargées au préalable à une certaine position du tableau qui va ensuite être convertie en position dans la fenêtre. Par exemple, si la case de la 3eme colonne et de la 3eme ligne possède la valeur 1, ses coordonées (3,3) vont être converties en pixels par une fonction « conv » qui les multiplie par 30. Donc les coordonnées en pixels du personnage 1, c'est-à-dire le bord en haut à gauche de l'image du personnage 1, se trouvera à 90pixels verticaux et 90pixels horizontaux du bord en haut à gauche de la fenêtre.
Grâce à ces deux fonctions, toutes les informations changées dans le tableau induiront un changement de l'affichage (on relance la fonction affichage à chaque boucle d'évènements).
On peut donc utiliser le tableau pour gérer tous les évènements :
Par exemple,
La fonction « simul_bombe » :
Elle prend pour paramètre des coordonnées du tableau et remplace le contenu de toutes les cases en forme de croix autour de lui par un 0 sauf celles qui contiennent un 8.

Avant l'explosion de la bombe :

8	8	8
1	5	9
0	0	9

Après l'explosion de la bombe :

8	8	8
0	0	0
0	0	9

Je vais également vous détailler une fonction que j'ai dû développer : les fonctions de déplacements.
Il y en a 4, une pour chaque direction.
Dans chacune de ces fonctions, on utilise trois variables : « x », « y » et « nb ».
Ces trois variables sont situées dans une liste appelé lperso qui est initialisée avec ces valeurs : [(1,1,1),(13,13,2)].
Le premier indice (donc lperso[0]) correspond au personnage 1 et le deuxième au personnage 2. Dans ces deux indices, on retrouve les trois variables utilisées pour la fonction de déplacement. La première correspond à x, la deuxième à y et la troisième à nb.
Par exemple, au départ, on voit que le sur la case de coordonnée (x=1,y=1) se trouve le personnage 1 (nb=1).
La variable « nb » nous a permis de faire 4 fonctions de déplacements au lieu de 8 pour les deux personnages.
La fonction va donc utiliser ces variables pour changer la position du personnage 1 ou 2 (en fonction de la variable « nb ») dans le tableau.
Pour cela, la fonction vérifie que la case vers laquelle le personnage se déplace est bien vide. Par exemple, si il s'agit d'un déplacement vers la droite, il faudra vérifier les coordonnée x,y+1.
Puis on remplace la valeur que possède cette case par un 1 ou un 2.
De plus, les coordonnées du personnage seront modifiées dans la liste. En effet, comme la fonction de déplacement modifie la variable x ou y, puis renvoie les coordonnées à la liste lperso, les coordonnées dans la liste lperso sont changées à chaque déplacement du personnage.
Dans le cas où le personnage vient de poser une bombe (la case sur laquelle il se trouve possède donc la valeur 6 ou 7 selon le personnage), lorsque le joueur se décale, on va remplacer la case où il se situait auparavant par une bombe.
Nous pourrions améliorer le programme en fusionnant ces 4 fonctions en une seule en ajoutant une variable correspondant au type de déplacement (ex : gauche=1, droite=2…)

Schéma d'un déplacement dans le tableau :

8	9	8
8	1	0
8	0	9

Utilisation de la fonction « d_droite » :

8	9	8
8	0	1
8	0	9

Toutes ces fonctions sont utilisées dans les évènements. Lorsque l'utilisateur appuie sur une touche de déplacement, on remplace la valeur des coordonnées dans la liste lperso par ce que renvoie la fonction de déplacement en utilisant les coordonnées de la liste. Par exemple :

```
if event.key==K_a: # lorsque l'on appuie sur la touche Q le perso1 se déplace à gauche
    lperso[0]=d_gauche(lperso[0][0],lperso[0][1],lperso[0][2])
```

IV) Conclusion

Pour conclure, je vais récapituler les problèmes qu'il faudrait encore résoudre :

- Les mouvements des personnages se font blocs par blocs, il est donc facile d'avancer très vite. L'idéal serait de refaire les fonctions de déplacements en faisant bouger les personnages pixels par pixels et faire correspondre leur position en pixels avec une position dans le tableau.
- Il n'y a pas de sons, une musique et des bruitages seraient les bienvenus
- Des améliorations pourraient également être ajoutées : quand on casse un bloc on a un certain pourcentage de chance de faire tomber une amélioration qui s'ajoute aux bombes du personnage qui passe sur la case qui la possède et augmente la portée d'explosion des bombes de ce personnage par exemple.
- Les fenêtres du jeu « s'accumulent », ce qui fait que le jeu a du mal à se fermer
- Les fonctions de déplacements pourraient être fusionnées en une seule fonction.

J'ai beaucoup apprécié faire mon année de Terminale avec la spécialité ISN, j'ai beaucoup appris sur l'informatique et la programmation et je compte approfondir mes connaissances dans ce domaine lors de mes études post-bac. Travailler en équipe pour notre projet fut également un côté positif de cette spécialité, cela nous a permis, à Adrien, à Amélie et à moi d'apprendre à se comprendre et à s'organiser entre nous.