Compte Rendu : Bomberman

Titouan Rannou

17 mai 2016

Table des matières

1 Cahier des Charges

1.1 Présentation

Le but de ce projet est de créer un clone du jeu vidéo *Bomberman*. Il s'agit d'un jeu multijoueur qui se joue à 2 joueurs. Voici les régles :

1.1.1 Règles du jeu

Imaginez une arène (vue de dessus) composée de blocs incassables, de blocs cassables (briques) et de chemins praticables. Le jeu se joue à 2 joueurs, ce sont tous les 2 des Bombermans. Il peut se déplacer et poser des bombes. Après quelques secondes elles explosent et détruisent les blocs cassables proches. Les régles sont simples : il ne doit en rester qu'un. Il faut utiliser toute sa ruse, récupérer des items bonus et poser plusieurs bombes afin de venir à bout de votre adversaire.

1.1.2 Environnement

Nous décrivons dans cette section l'arène du jeu : La construction de l'arène se fait à l'aide des images suivantes :

— Des blocs *incassables* 

— Des blocs *cassables* 

— Des 2 personnages 

Il y aura aussi les items suivants dans l'arène :

— Les 2 bombes pour chaque personnage : 

— Le bonus de portée de la bombe 

— Le bonus qui donne un point de vie 

— Le bonus qui augmente le nombre maximal de bombes posés en même temps 

Chaque joueur commence à des coins opposés de l'arène. Ils peuvent se déplacer dans 4 directions, Nord, Sud, Est, Ouest. Ils ont chacun 3 vies en début de partie. Ils en perdent une s'ils sont dans la portée de la bombe. Si un joueur n'a plus de vie il a perdu.

1.1.3 Ce que le joueur peut ou doit faire

— Ne rien faire
— Se déplacer dans l'arène sur les cases vides
— Pouvoir récupérer des bonus
— Aller sur la même case qu'un joueur, une bombe ou un bonus
— Poser des bombes

1.1.4 Ce que le joueur ne peut pas faire

— Sortir de l'arène
— Traverser des obstacles (blocs et briques)
— Jouer une fois mort
— Poser plus de bombes en même temps que son nombre maximal

1.1.5 Ce que les bombes peuvent ou doivent faire

— Exploser au bout de un certain temps secondes
— Être traversées par les joueurs

1.1.6 Ce que les bombes ne peuvent pas faire

— Cassser des blocs (éléments incassables)

1.1.7 Ce que l'explosion peut ou doit faire

— Enlever un point de vie au joueur qui a posé la bombe ou/et au joueur adversaire

— Casser des briques (éléments cassables) correspondant à la portée du joueur
— Faire apparaître différents bonus

1.1.8 Bonus

— Vie
— Augmentation de la portée de la bombe

1.2 Architecture du Programme : Fonctions

Dans cette partie nous présenterons les différentes fonctions qui permettent le bon fonctionnement du jeu :

menu : Fait apparaitre le menu

startgame : Démarre le jeu

perdu : Permet a l'utilisateur de rejouer ou de quitter la partie

dessinermap : Dessine l'arène de jeu

ajouter_brique : Dessine une brique

ajouter_bloc : Dessine un bloc

bombe1 : Pose une bombe qui explose au bout d'un certain temps pour le joueur 1

bombe2 : Pose une bombe qui explose au bout d'un certain temps pour le joueur 2

explosion : Fait exploser la bombe

destruction_animation_explosion : Détruit l'animation de la bombe

bonus_bombe : Pose un bonus qui augmente la portée de la bombe aléatoirement

verif_bonus_bombe : Verifie si le joueur 1 est sur le bonus de portée de bombe

verif_bonus_bombe1 : Verifie si le joueur 2 est sur le bonus de portée de bombe

bonus_vie : Pose un bonus de vie de la bombe aléatoirement

verif_bonus_vie : Vérifie si le joueur 1 est sur le bonus de vie

verif_bonus_vie : Vérifie si le joueur 2 est sur le bonus de vie

bonus_recharge : Pose une bonus qui augmente le nombre maximal de bombes

verif_bonus_recharge : Vérifie si le joueur 1 est sur le bonus de recharge

verif_bonus_recharge1 : Vérifie si le joueur 2 est sur le bonus de recharge

enleve_vie : Enlève un point de vie à un joueur

rajoute_vie : Rajoute un point de vie à un joueur

personnages : Crée les 2 personnages

animdroite : Déplace le joueur 1 vers la droite

animgauche : Déplace le joueur 1 vers la gauche

animbas : Déplace le joueur 1 vers le bas

animhaut : Déplace le joueur 1 vers le haut

animdroite2 : Déplace le joueur 2 vers la droite

animgauche2 : Déplace le joueur 2 vers la gauche

animbas2 : Déplace le joueur 2 vers le bas

animhaut2 : Déplace le joueur 2 vers le haut

2 Compte-Rendu

2.1 Recherche documentaire

Passionnés d'informatique, nous connaissions le site openclassrooms (anciennement site du zéro), qui est un site de cours en ligne gratuit, orienté vers la programmation informatique. Nous avons donc suivi le cours d'initiation à python pour voir des notions que nous n'avions pas vu en cours, ou réviser et approfondir celles que nous avions vu en cours.

2.2 Travail d'équipe

2.2.1 Github

Tout d'abord nous nous sommes heurtés à des difficultés pour travailler ensemble, partager et mettre en communs nos travaux. Nous avons donc cherché une solution : Git. Git est un logiciel de gestion de versions, c'est à dire qu'il permet de stocker nos fichiers en conservants la chronologie de toutes les modifications qui ont été effectuées dessus. Pour un travail d'équipe efficace nous avons utilisé le service web git hub, qui est un réseau social utilisant le système Git. Git Hub nous a permis de nous organiser, ne pas perdre la trâce de nos anciennes modifications. Par exemple, lorsque l'utilisateur fait une modification dans son code, et le code ne marche plus, on peut revenir à une version antérieur ou le code marchait pour pouvoir identifier le problème et le régler.

2.3 Répartition du travail

Dans notre groupe, nous avons essayé au maximum de répartir le travail de façon équitable, afin d'être le plus efficace possible. Ainsi, nous avons réalisé, chacun de notre côté, des fonctions,que nous avons ensuite rassemblé, au fur et à mesure de l'élaboration de notre projet.

Voici les fonctions que nous avons donc réalisé, ainsi que leur répartition entre chaque membre du groupe :

2.3.1 Clément Guin

- menu
- startgame
- perdu
- animdroite
- animgauche
- animbas
- animhaut
- animdroite2
- animgauche2
- animbas2
- animhaut2
- personnages
- enleve_vie

2.3.2 Titouan Rannou

- bonus_bombe
- verif_bonus_bombe

— verif_bonus_bombe1
— bonus_vie
— verif_bonus_vie
— verif_bonus_vie1
— bonus_recharge
— verif_bonus_recharge
— verif_bonus_recharge1
— dessiner_map
— ajouter_brique
— ajouter_bloc

2.3.3 Tanguy Thomas

— bombe1
— bombe2
— explosion
— destruction_animation_explosion
— rajoute_vie

2.4 Mon rôle dans le projet

Mon rôle dans l'équipe a été de créer l'arène sur laquelle on se déplace et joue ainsi que tous les bonus récupérables en détruisant des briques, c'est-à-dire de créer les fonctions faisant apparaître les bonus et de les utiliser dans les fonctions principales comme dans la fonction explosion et les fonctions de déplacements.

Pour créer la carte c'est-à-dire poser tous les blocs (incassables) et les briques (cassables) il m'a fallu utiliser deux fonctions auxiliaires (ajouter_brique et ajouter_bloc) qui crée seulement l'image d'une brique ou d'un bloc. La fonction dessiner_map() est un enchainement de double boucles permettant de représenter les deux axes quadrillant le Canvas (soit par exemple le for i représentant l'axe x et le for j l'axe y) et ainsi éviter de poser les blocs et les briques un par un.

Et c'est à partir de là que mon travail s'est compliqué, Il fallait qu'à chaque fois qu'une brique était détruite, qu'un bonus apparaisse avec une certaine probabilité.
Pour chaque bonus j'ai eu besoin de 3 fonctions :

— bonus_bombe, bonus_vie, bonus_recharge créant l'image du bonus avec une certaine probabilité.

Grâce à un a=randint(0,100) j'ai facilement pu exprimer la probabilité d'apparition du bonus en utilisant un if a<= au pourcentage voulu et ainsi importer l'image du bonus.
Ces fonctions sont utilisés dans la fonction explosion() qui l'exécute à chaque brique cassée

— verif_bonus_bombe, verif_bonus_vie, verif_bonus_recharge permettant de savoir si le joueur 1 récupère le bonus et ainsi récupérer les avantages du bonus
— verif_bonus_bombe1, verif_bonus_vie1, verif_bonus_recharge1 permettant de savoir si le joueur 2 récupère le bonus et ainsi récupérer les avantages du bonus

Les fonctions de vérifications utilisent les coordonnées du joueur ainsi que les coordonnées du bonus (retrouvées grâce au tags) ces deux coordonnées sont rentrées dans deux variables et si ces variables sont égales (le joueur est sur le bonus), on ajoute le bonus dans une liste vide (destroy=[] dans les fonctions) pour ainsi le supprimer. On ajoute ainsi

le bonus en question dans une variable globale (range_bombe, nb_bombes) utilisé dans d'autres fontions. Dans le cas de la fonction pour le bonus de vie on execute la fonction rajoute_vie(1 ou 2) (selon le joueur) qui ajoute une vie au joueur et va afficher le nombre de point de vie du joueur dans un autre canvas en dessous de la carte de jeu. Pour la fonction bombe j'incrémente de 1 ma variable et ainsi j'affiche cette variable dans le canvas en dessous du jeu pour que le joueur puisse connaître la portée de ses bombes. Pour la fonction recharge je rajoute une image de bombe dans le canvas en dessous du jeu pour que le joueur puisse savoir le nombre de bombes qu'il peut poser en même temps.

Ces fonctions sont utilisées dans les fonctions de déplacements des joueurs (animhaut(), animbas() etc..) pour tester à chaque déplacements si le joueur est sur le bonus et ainsi en bénéficier

2.5 Conclusion

Je suis intéressé par l'informatique et l'art de programmer depuis peu de temps et ce projet a vraiment confirmé mon intérêt, et mon goût que j'avais pour l'informatique. Monter un projet en groupe est une très belle expérience enrichissante en connaissance, et en maturité. J'ai beaucoup aimé partager et recevoir des connaissances avec le groupe. Nous nous entendons très bien ce qui a facilité le travail. Le choix du Bomberman était un choix difficile qui nous a posé beaucoup de problèmes mais qui au final nous a beaucoup amusé et enrichi.