**Compte-rendu BOMBEIRB**

**Hoarau Quentin – Fabiani Quentin**

**I-Introduction**

Le but du jeu est d’aller délivrer une princesse retenue prisonnière par des monstres en utilisant des bombes. Le projet est séparé en plusieurs fichiers, nous permettant de gérer chaque objet composant le jeu. Ce projet utilise la bibliothèque SDL pour l’affichage et est actualisé en permanence afin de rafraîchir les parties graphiques et non graphique du jeu.

**II-Gestion des déplacements**

Les déplacement se font selon quatre directions (NORTH,WEST,SOUTH,EAST) et sont géré dans la fonction player\_move(struct \* player player). Cependant nous devons vérifier si le joueur peut bien se déplacer vers la case où il veut se déplacer, par exemple, il doit rester dans la limite de la carte et il ne peut pas se déplacer sur des cases du décors (arbres et rochers). Les différents cas selon la case où le joueur veut aller sont géré dans la fonction player\_move\_aux qui permet donc de savoir si le joueur à le droit de se déplacer ici et si oui, les actions qui devront être exécuter, par exemple, pour déplacer les caisses il faut vérifier que la caisse peut être déplacer dans le sens voulu en vérifiant qu’il n’y ait rien sur le chemin.

**III-Gestion des cartes**

Les cartes sont dans le dossier map, la première carte est chargée dans la fonction game\_ini via la fonction load\_map qui lit les fichiers map et renvoie une structure map qui sera ensuite affiché par la fonction display\_map. Dans la fonction load\_map, nous utilisons la fonction scanf qui permet de lire chaque cellule du fichier que l’on concatène dans grid de la strcuture map. Cette fonction est appelée dans game\_ini mais aussi à chaque fois que le joueur passe une porte.

**IV-Gestion des portes**

Les portes sont peuvent être ouvertes ou fermées, ce qui est codéesur le dernier bit et mènent à des niveaux, codé par les 3 avant derniers bits. Ainsi, dans player\_move\_aux dans me cas porte, nous regardons si elle est fermée et si le joueur a une clé ou si la porte, si oui le joueur passe au niveau codé dans les 3 avant derniers bits grâce à la fonction load\_map et nous augmentons où diminuons le niveau de la carte selon si le joueur va au niveau supérieur ou inférieur, sinon le joueur ne peut pas se déplacer sur la case porte.

**V-Gestion du panneau d’information**

Pour afficher les informations du panneau d’information, nous utilisons la fonction game\_banner\_display() puis window\_display\_image pour afficher les informations comme la vie, le nombre de bombe et le portée du joueur. Nous avons également rajouté une case clé qui apparaîtra quand le joueur aura une clé.

**VI-Gestion des bombes**

Pour les bombes, nous utilisons une liste chaînée via une structure dans laquelle l’on stock les informations comme la position de la bombe, la portée, le temps. Nous avons en effet dû gérer le temps pour les explosions grâce à la fonction SDL\_GetTicks(). Lorsque le joueur veut poser une bombe, on vérifie qu’il a au moins une bombe sur lui, on l’initialise puis on ajoute cette bombe à cette liste chaînée. Ensuite la fonction update\_bomb appelée dans game\_display elle-même appelé dans la boucle de la fonction main, va regarder depuis quand la bombe a été posé puis afficher les les bonnes images de la bombe pour la mèche pour créer une animation.

Pour l’explosion, nous utilisons une boucle for qui va dans toutes les directions (NORTH,WEST,EAST,SOUTH) de la bombe en respectant sa portée et regarde les différent cas, par exemple si c’est un élément du décor, l’explosion s’arrête là, si c’est une caisse, la détruit et regarde ce qu’il se passe à l’intérieur, ou si c’est un joueur où un monstre, lui inflige des dégâts. Tout ceci est fait dans les fonctions start\_fire et check\_fire. Puis finalement, pour arrêter l’explosion nous regardons toutes les cases en feu et nous les mettons vides.

**VII-Gestion des bonus et malus**

Lorsque que le joueur se déplace sur une case bonus, on regarde de quel bonus il s’agit via la fonction map\_get\_bonus\_type puis on ajoite le bonus au joueur. Un bonus sera détruit par une explosion, ce qui est gérer dans la fonction start\_fire, les bonus apparîssent lorsque qu’une caisse en contenant un explose, gérer dans la fonction check\_fire.

**VIII-Gestion des vies**

Le joueur perd une vie s’il est sur le chemin d’une explosion ou s’il est touché par un monstre, ce qui est géré dans la fonction player\_move\_aux, monster\_mov\_aux et update\_bomb. De plus nous avons rajouté un timer pour que le joueur ne puis pas prendre plus d’un dégât par deux secondes.

**IX-Gestion des monstres**

Les monstres sont également gérés par une liste chaînée via une structure, les monstres peuvent apparître après l’explosion d’une caisse ou être présent sur la map des le début, ce qui est fait par la fonction spawn\_new\_monster et spawn\_monster\_map, les monstres bougent grâce à la fonction monster\_move et monster move\_aux qui fonctionne sur le même principe que les fonctions player\_move et player\_move\_aux. Les monstres sont blessés lorsque qu’ils se retrouvent sur le chemin d’une explosion, géré dans la fonction start\_fire et monster\_move\_aux.

**X-Pause**

Lorsque que le joueur appuie sur p, le jeu se met en pause, on regarde à combien de temps nous nous somme arrêtés pour ensuite pouvoir donner le nouveau timer aux bombes eu aux monstres.