Version du 5 février 2016 à 17:37

PG110 : Projet Bombeirb

2015-2016

L'objectif du projet de programmation est la réalisation d'un jeu 2D en C s'inspirant du jeu *bomberman*. Nous appelons notre jeu `bombeirb`.

1 Principes du jeu

Le joueur doit traverser les cartes composant les différents niveaux du jeu pour aller délivrer la princesse en se servant de ses bombes. Des portes lui permettent de passer d'un niveau à un autre. Les portes peuvent être fermées et nécessitent alors de posséder une clef dans l'inventaire. Une cellule peut contenir :

- le joueur ou la princesse
- des éléments de décors (arbres, pierres...) infranchissables et indestructibles ;
- des caisses destructibles ;
- des portes, ouvertes ou fermées, permettant d'évoluer entre les cartes ;
- des clefs pour débloquer les portes fermées ;
- des bonus ou des malus

2 Travail fourni

2.1 Matériel

Nous vous fournissons une première ébauche du jeu, utilisant les bibliothèques `SDL` (Simple DirectMedia Layer) et son extension `SDL_image` pour l'interface utilisateur et la gestion du 2D. Le programme a été testé sur Linux, MacOS et Windows. Le lancement du jeu fait apparaitre une carte, chargée statiquement en mémoire (Figure 1), dans laquelle le joueur peut se déplacer sans limite dans toutes les directions quelque soit la nature des cellules. L'archive contenant les sources du squelette de jeu ainsi que le guide d'installation se trouvent à l'adresse `http://reveille.vvv.enseirb-matmeca.fr/perso/pg110/`.

2.2 Codage des cartes

Une carte de taille *Largeur* x *Hauteur* est représentée sous la forme d'un tableau à une dimension tel que la cellule de coordonnées (`i,j`) sur la carte corresponde à l'élément d'indice (`i + j*Hauteur`) dans le tableau. La valeur d'un élément du tableau représente le type de la cellule correspondante. Cette valeur est codé par un entier sur 8 bits (`char`). Le codage de chaque cellule est définit de la manière suivante :

b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0

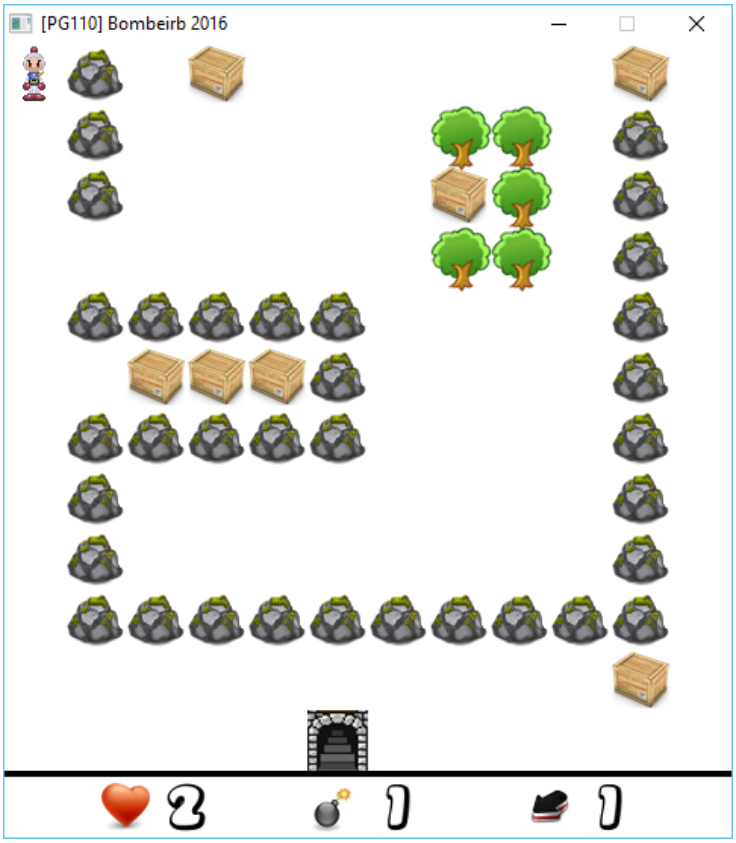

FIGURE 1 – Carte initiale de l'ébauche de jeu fournie

Les 4 bits de poids faibles (b_3 - b_0) représentent le type de la cellule avec le codage suivant :

- **0** : Vide
- **1** : Décor (arbres et pierres)
- **2** : Joueur
- **3** : Princesse
- **4** : Caisse
- **5** : Bonus / Malus
- **6** : Monstre
- **7** : Bombe
- **8** : Clef
- **9** : Porte

Les 4 bits de poids forts (b_7 - b_4) codent le sous-type de la cellule ou des informations complémentaires.

Décor Le bit b_4 code le type de décor : 0 pour un arbre et 1 pour une pierre. Les bits b_7 à b_5 sont inutilisés.

Caisse Les bits b_7 - b_4 codent ce qui se trouve dans la caisse avec le codage suivant

0 : Vide
1 : Diminue la puissance des bombes
2 : Augmente la puissance des bombes
3 : Diminue le nombre de bombes
4 : Augmente le nombre de bombes
5 : Monstre
6 : Vie supplémentaire

Porte Le bit b_7 code l'état de la porte : **0**=fermée, **1**=ouverte. Les bits b_6-b_4 codent le numéro de la carte atteignable en franchissant la porte.

3 Travail à fournir

Il vous est demandé de compléter l'ébauche de jeu fournie, de produire un court rapport d'au plus deux pages, ainsi que de faire une démonstration de votre implantation des fonctionnalités demandées. Le rapport devra contenir, pour chaque fonctionnalité ajoutée, une description de la solution adoptée. Il devra être remis au format `PDF`. Votre code devra être clairement commenté et indenté. Les fonctions seront nommées en anglais. Les fonctionnalités demandées constituent un cadre obligatoire à partir duquel vous êtes libres d'agrémenter le jeu comme vous l'entendez. Nous décrivons ci-après les fonctionnalités attendues du jeu.

3.1 Gestion des déplacements

Les mouvements du joueur sont limités par le cadre de la carte, les éléments de décors et les caisses. Les caisses doivent pouvoir être déplacées par le joueur si rien ne gêne dans le sens de la poussée. Si un bonus se trouve dans la direction déplacement d'une caisse, la caisse reste bloquée. Le joueur ne peux pas déplacer deux caisses à la fois.

3.2 Gestion des mondes

Les cartes sont décrites dans des fichiers dans le dossier `data` et chargées en mémoire par le programme. Nous définissons les conventions suivantes :

— Les cartes sont stockées sous forme de fichiers texte afin de pouvoir les créer et les modifier avec un simple éditeur de texte ;
— Le nom de fichier d'une carte est de la forme `map_N` ou `N` est le numéro de niveau ;
— La première ligne du fichier, de la forme *width:height*, représente la largeur et la hauteur de la carte (valeurs décimales) ;
— La case en haut à gauche de la carte correspond aux coordonnées `(0,0)` ;
— Chaque ligne correspond à une ligne de cellule sur la carte ; Chaque cellule est suivie par un au moins un espace (y compris la dernière cellule d'une ligne) ;
— Chaque cellule de la carte est définie en respectant le codage défini au début du sujet, en format décimal. Ainsi, une porte ouverte donnant accès au niveau 7 donne un codage binaire de `[1][111][1001]` = 249 en décimal.

Comme nous n'utilisons que 3 bits pour coder les différentes carte du jeu, il y a un maximum de 8 niveaux possibles. Le premier niveau est toujours le niveau 0. L'exemple ci-dessous illustre le codage d'une carte de 3 par 2. Le symbole ␣ représente un espace.

```
3:2
2␣0␣␣␣17␣
0␣249␣0␣
```

3.2.1 Chargement des cartes

Écrire les fonctions permettant de charger une carte à l'écran à partir d'un fichier.

3.2.2 Gestion des portes

Lorsque le joueur arrive sur la case d'une porte ouverte, il passe automatiquement au niveau correspondant à cette porte. Si la porte est fermée, le joueur doit ramasser la clé présente sur la carte.

3.2.3 Gestion du panneau d'informations

Le panneau d'information doit afficher le nombre de vies, le nombre de bombes et leur portée, la présence d'une clef dans l'inventaire et le numéro de niveau.

3.3 Gestion des bombes

Lorsque le joueur presse la touche `[ESPACE]`, il dépose une bombe sur la case sur laquelle il se trouve, déclenchant une explosion au bout de 4 secondes. La mèche de la bombe diminue à chaque seconde. La portée de la bombe est par défaut de 1 case, en croix (case du dessus, case du dessous, case de gauche, case de droite). Les éléments de décor stoppent la propagation de l'explosion dans le sens qu'ils obstruent (Fig. 2). Si une caisse est sur le chemin de l'explosion, elle disparait pour laisser apparaitre son contenu. Une explosion ne peut détruire qu'une seule caisse dans une même direction. Si c'est un bonus, un malus ou un monstre, il disparait. Enfin, si le joueur est sur une cellule touchée par une explosion, il perd une vie. Les explosions n'ont aucun effet sur les portes et les clefs. Lorsque une bombe explose, une nouvelle bombe est ajoutée à l'inventaire du joueur.

(a) sans obsacle (b) avec obstacle

FIGURE 2 – Exemples d'explosions d'une bombe de portée 1

3.4 Gestion des bonus et malus

Les bonus et malus peuvent être sur la carte ou apparaitre lors de l'explosion d'une caisse. Il en existe 5 :

- **portée-** / **portée+** : ajoute/retire 1 à la portée des bombes. La portée ne peut pas être nulle. Le changement de portée ne concerne que les bombes qui seront posées plus tard. Les bombes pour lesquelles la mèche est déjà allumée conservent leur portée initiale.
- **bomb-** / **bomb+** : ajoute/enlève une bombe à l'inventaire. Le joueur dispose toujours d'au minimum 1 bombe.
- **vie+** : ajoute une vie

3.5 Gestion des vies

Le joueur dispose de 2 vies au démarrage du jeu. Il peut en perdre s'il se trouve sur une case à portée de l'explosion d'une bombe. Si le joueur n'a plus de vie, la partie se termine.

3.6 Gestion des monstres

Les monstres peuvent être présents dès le chargement de la carte ou apparaitre à l'explosion d'une caisse. Leurs déplacements sont entièrement aléatoires. Une collision avec un monstre déclenche la perte d'une vie. Le joueur bénéficie alors d'une temporisation pendant lequel il est invulnérable.

Commencez par ajouter un seul monstre à la fois, puis augmenter le nombre de monstres. La vitesse de déplacement des monstres doit être faible dans les premiers niveaux et augmenter plus on se rapproche de la princesse. Ceux qui le souhaite peuvent ajouter un module d'intelligence artificielle pour que les monstres se dirigent vers le joueur et non plus aléatoirement.

3.7 Pause

La touche `[P]` met le jeu en pause et permet de reprendre la partie. Si une ou plusieurs bombes ont été posées et n'ont pas encore explosé lors de la pause, le temps avant leur explosion doit être le même lors de la reprise du jeu.

3.8 Sauvegarde / Chargement partie

On souhaite pouvoir enregistrer une partie en cours. Il faut sauvegarder l'état d'avancement du joueur, son nombre de vie et de bombe, sa position sur la carte en cours, la présence ou non d'une clef dans son inventaire, l'état des cartes du niveau (monstres, décor). Une fois que l'enregistrement est effectué, il faut permettre de recharger la partie enregistrée. Le choix du format d'enregistrement est libre.