Rapport de projet : Bubble-Bobble

IHDC B132

Cédric Evrard

1 Introduction

L'object de ce projet est de réaliser notre propre version du jeu video Bubble-Bobble [1]. Ce jeu video sera réalisé à l'aide du langage de programmation C ainsi qu'avec la bibliothèque graphique OpenGL et de la bibliothèque GLUT.

La réalisation du jeu est divisé en 2 étapes, une premire étape sera de réaliser le coeur du jeu, c'est à dire, un personne fonctionnel, la présence d'ennemis le tout dans un seul niveau. La deuxième partie sera libre et l'objectif de celle-ci sera d'améliorer de la meilleur des faons possible le jeu via, par exemple, l'ajout d'une intelligence artificielle pour les ennemis, la création de plusieurs ennemis, de plusieurs niveaux, ...

2 Projet

2.1 Présentation des écrans

2.1.1 Accueil

L'écran sera affiché à l'ouveture du programme. Sur cet écran (Figure 1), il y aura le titre du jeu ainsi qu'un menu de sélection permettant d'accéder aux autres écrans de l'application.

Figure 1: Écran d'accueil

2.1.2 Contrôles

Cet écran (Figure 2) sera affiché avant chaque partie afin de présenter au joueur les contrôles du jeu afin qu'il puisse facilement savoir comment jouer au jeu.

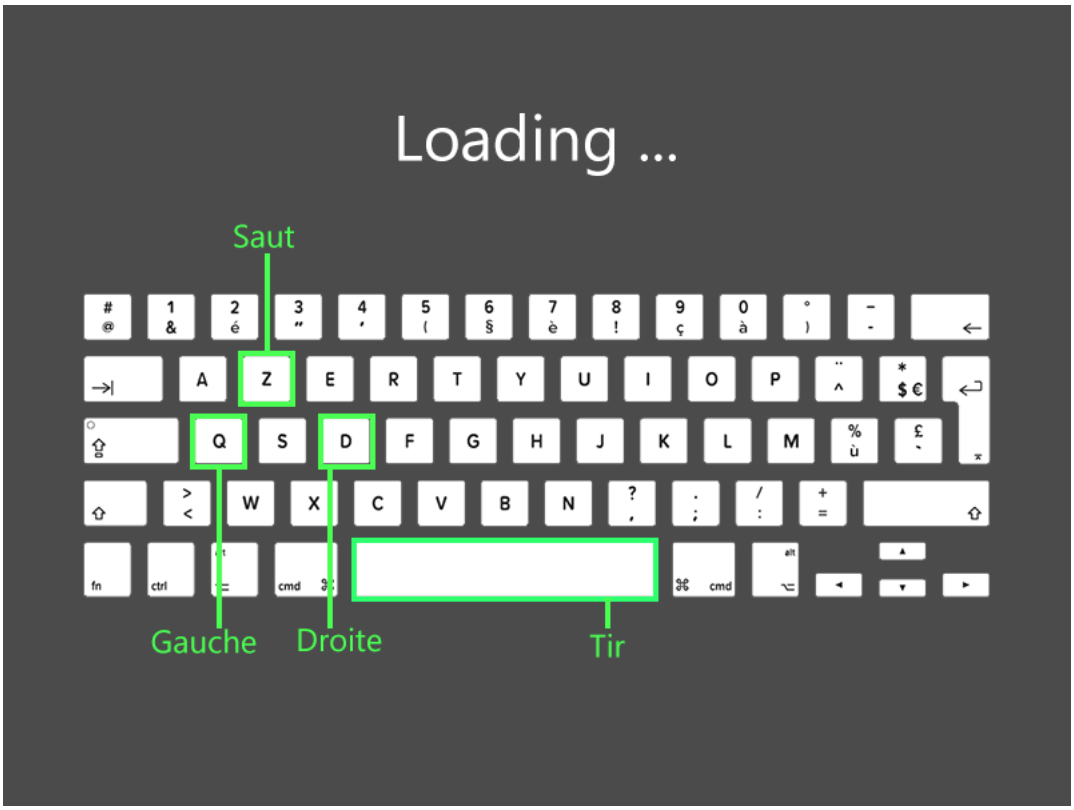

Figure 2: Écran des contrôles affichés au chargement

2.1.3 Jeu

Écran principal de l'application (Figure 3). Il présentera le niveau ainsi que certaines informations au joueur. Le score du joueur sera affiché en haut à gauche, le niveau en haut à droite et le nombre de vie, en bas à gauche.

2.2 Fonctionnement du coeur du jeu

Cette section décrira les diffrents éléments pour le fonctionnement du jeu. Comment le monde est représenté ou comment la détection entre deux éléments du jeu sera détecté.

2.2.1 Représentation de la carte

La carte sera divisé en une grille de 32 cellules de large sur 25 cellules de haut. Chaque cellule de cette grille sera un mur ou une zone vide. Ensuite, chaque bloc aura une taille de 32 pixels de large sur 32 pixels de haut et sera rempli avec une texture.

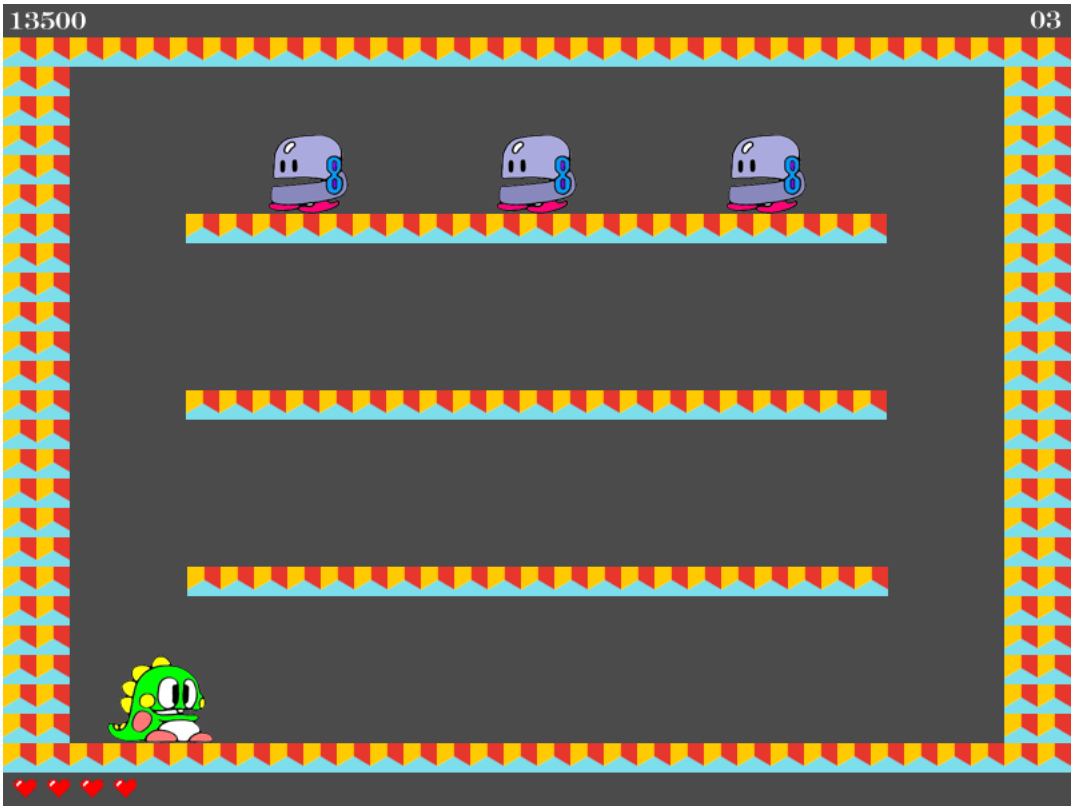

Figure 3: Écran de jeu

2.2.2 Représentation des personnages

Les personnages du jeu, que ça soit pour le joueur ou pour les ennemis, seront représentés à l'aide d'une image qui sera positionnée sur un axe x-y. Le personnage sera entouré d'une "hitbox" [2]. La hitbox sera un rectangle qui entour le personnage (voir exemple figure 4). Ce rectangle permettra de savoir quand deux éléments entre en collision. En effet, lorsque le rectangle d'une "hitbox" rencontre le rectangle d'une autre "hitbox", cela signifie que deux éléments du jeu se sont rencontrés.

Figure 4: La "hitbox" est représentée en rouge

Le même système sera utilisé pour les bulles lancées par le joueur. Même si les bulles sont représentés via une image ronde, les "hitboxs" de celles-ci seront aussi des rectangles afin de faciliter les calculs liés à la détection de "hit".

2.2.3 Gestion des mouvements

La gestion des mouvements se fera via les touches 'Q' (mouvement à droite), 'D' (mouvement à gauche) et 'Z' (mouvement de saut). Lorsque le joueur va à gauche ou à droite, le jeu va vérifier que le joueur ne rentre pas en collision avec un mur. Pour cela, on va regarder la position du joueur par rapport au tableau qui représente la carte. Si le joueur entre dans une cellule du tableau représentant un mur, il sera arrêté dans son mouvement.

Lorsque le joueur saute (via la touche 'Espace'), le jeu va détecter si celui-ci était en mouvement lors du saut ou non. Si celui-ci n'tait pas en mouvement, le saut sera réalisé à la vertical par rapport à sa position de départ. Dans le cas d'un saut alors que le joueur est en mouvement, le saut sera réalisé sous la forme d'une hyperbole.

Lors de la chute du joueur (que ça soit lors d'un saut ou d'une chute, le joueur aura la possibilité de choisir la direction de sa descente ainsi que de ralentir la chute en appurant sur la touche de saut.

Enfin, dernier point lié à la fonction de saut. Lors de la période ou le joueur monte, il ne sera pas bloqu par les murs présent dans le niveau (seul les murs qui entourent la carte seront bloquant). Par contre, lors de la chute, les murs bloqueront le joueur.

2.2.4 Élimination des ennemis

Lorsque le joueur lance une bulle sur un ennemi, celui-ci sera directement transformé en bonbon. Le joueur devra ensuite aller récupérer pour gagner des points. Les bonbons resteront à l'écran un temps défini, aprés cela, ils disparaitront de lécran.

2.3 Stuctures utilisées

2.3.1 Structure de "hitbox"

Une "hitbox" sera représenté via une position dans l'écran de jeu ainsi qu'une hauteur et une largueur.

```
typedef struct hitbox {
  int positionX;
  int positionY;
  int hauteur;
  int largueur;
} Hitbox;
```

2.3.2 Structure de personnage

Les personnages, que ça soit le joueur ou les ennemis, seront représentés à l'aide d'une hitbox, un nombre de vie, une chaine de caractère qui donnera le chemin vers la texture du personnage ainsi qu'un booléen indiquant s'il s'agit d'un personnage amical ou non.

```
typedef struct personnage {
  Hitbox hitbox;
  int nombreVie;
  char *image;
  bool amical;
} Personnage;
```

2.4 Stuctures de donnés dynamiques utilisées

2.4.1 Gestion des collisions

Afin de faciliter la recherche de collision entre deux éléments du jeu, une liste chaînée contenant les informations de position et de taille des "hitbox" sera utilisé. Cette liste aura pour but de pouvoir parcourir le plus rapidement possible les "hitbox" et savoir si deux de celle-ci se croisent.

2.4.2 Ennemis

Les ennemis seront eux aussi présent dans une liste chaînée afin de pouvoir facilement ajouter ou retirer des ennemis du jeu.

3 Conclusion

La principale difficulté de se projet va résider dans le fait d'optimiser la gestion des hitbox afin que l'application se soit pas ralentie par la recherche de collision. En effet, à chaque tour de la boucle principale de GLUT, on va devoir vérifier si deux éléments sont entrés en collision, il faut donc optimiser au maximum cette gestion.

La deuxième difficulté sera lié a la gestion des mouvements du joueur et des ennemis. Ceux-ci pouvant se déplacer librement, il faudra faire attention à ne pas bloquer le joueur dans un mur ou alors à l'opposé, ne pas l'autoriser à se déplacer trop librement sur la carte et donc sortir de celle-ci.

4 References

[1] "Bubble Bobble - StrategyWiki," https://strategywiki.org/wiki/Bubble_Bobble.

[2] "Hitbox - VALVe Developer Community," https://developer.valvesoftware.com/wiki/Hitbox.