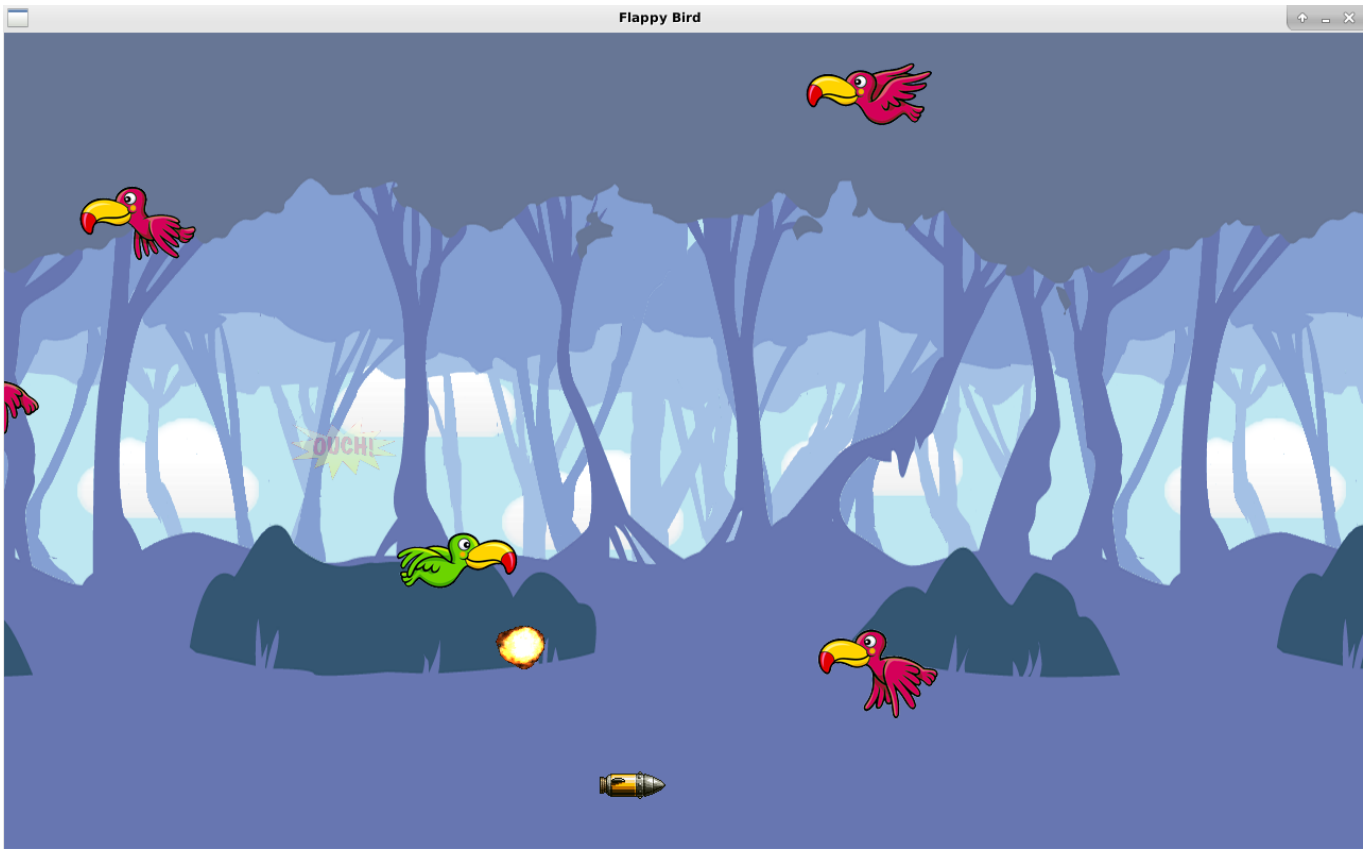
PROJET TECHNOLOGIQUE
L3

Rapport de Projet

Yohan Lematre
Quentin Fergelot



1/ Fonctionnement du jeu

Vous contrôlez un oiseau et pouvez voler en appuyant sur la touche « haut » ou descendre en piqué en appuyant sur la touche « down ». Vous êtes face à des oiseaux ennemis et pouvez les détruire en leur tirant un missile dessus, en appuyant sur la touche « espace ».

Les oiseaux ennemis apparaissent du bord opposé de l'écran, et se rapprochent de vous en suivant une trajectoire définie à l'avance. Si l'un d'entre eux vous touche, il disparaît et vous prenez un dégât, ce qui vous immobilise quelques secondes.

En effet si vous prenez un dégât vous ne pourrez plus interagir jusqu'à ce que votre oiseau touche le sol. Vous allez alors clignoter et un «OUCH » s'affichera pour vous signaler votre bévue.

Vous ne pouvez pas sortir de l'écran, si vous atteignez le plafond, vous prenez également un dégât.

2/ Difficultés rencontrées

Les principales difficultés que nous avons rencontrées concernent l'utilisation du fichier « list.h » donné, qui fut difficile à comprendre de premier abord, tout comme la compréhension du fonctionnement des timers.

Dans le code de notre projet, deux fonctions demandées sont vides à cause de ces incompréhensions : *animation_timer_expired* et *timer_init*. En effet elles ne nous ont pas semblé utiles, ou du moins nous n'avons pas saisi la manière de les utiliser.

Nous nous sommes documentés sur le Web et avons discuté avec un camarade pour mieux comprendre les fonctionnalités des listes et des timers et réussir à s'en servir dans notre projet.

De plus, le paterne d'oscillation des oiseaux ennemi a été compliqué à implémenter. De la même manière des discussions avec les professeurs et des recherches nous ont permis de nous en sortir. Malgré tout notre implémentation n'est toujours pas parfaite et mérite que l'on se penche à nouveau dessus pour l'améliorer.

3/ Problèmes connus et améliorations possibles

Quelques problèmes non résolus persistent dans le code de notre projet, rien de très important cependant.

Premièrement, dans ce projet initial nous n'avons que très peu utilisé le fichier « constantes.h », et la plupart des valeurs absolues ont été écrites en de manière brute dans le code.

Un premier problème auquel nous avons réfléchi est le fait que la fréquence de tirs de missile est dépendante de la fréquence d'affichage de l'écran, et non du temps entre chaque salve. De ce fait plus un utilisateur a une fréquence d'affichage élevée plus il pourra tirer rapidement, avec les conséquences que l'on peut imaginer.

De plus, pendant la génération des oiseaux ennemis, la valeur initiale du positionnement de l'oiseau sur sa courbe sinusoïdale n'est pas complètement aléatoire. En effet l'oiseau ne peut apparaître qu'à deux emplacements différents de sa courbe.

C'est un problème mineur mais qui mérite tout de même d'être corrigé. L'effet est notamment visible lorsque deux oiseaux successifs et ayant la même vitesse apparaissent avec le même point de positionnement et on donc une trajectoire entièrement parallèle.

Le problème le plus dérangeant d'un point de vue technique est un problème au niveau du calcul des collisions. Le calcul des hitboxs des objets n'est pas parfait et ne calcule des collisions qu'avec un seul des quatre cotés de notre oiseau. Ce problème est parfois visible lorsque l'on arrive sur un oiseau ennemi par le dessus.

Aussi, la gestion des divers objets pouvant entrer en collision entre eux se fait au cas par cas et pourrait très certainement être améliorée.

Enfin, le principal problème concerne un élément de jouabilité. En effet les choix que nous avons faits donnent à l'oiseau une impulsion verticale assez élevée lorsque nous appuyons sur la touche « up », ce qui rend compliqué le fait de ne pas prendre de dégâts contre les ennemis ou avec le haut de l'écran.

Nous avons réfléchi à une solution à ce problème et pensons savoir comment la mettre en place mais nous n'avons pas eu le temps de nous y consacrer pleinement, préférant centrer nos efforts sur la suite des instructions données dans les feuilles de TD.

4/ Éléments intéressants de notre projet

À la suite de la découverte du fichier « list.h » et de l'implémentation de la macro *for_all_object* nous nous sommes inspiré de ce travail pour mettre en place une autre macro appelée *for_all_other_object* qui nous permet à partir d'un objet particulier de parcourir tous les autres objets de la liste sauf celui-ci.
Cela nous est utile dans la partie concernant la gestion des collisions et nous permet d'éviter d'un objet vérifie une possible collision avec lui-même.
Nous somme satisfait de ce point, car la compréhension de macros aussi complexes que celles données dans « list.h » à été compliqué pour nous.

Un autre point satisfaisant de notre projet est l'ajout d'un champ *dead_function* dans la structure *object_class* qui permet d'associer à chaque type d'objet une fonction appelée en conséquence d'une collision avec un autre objet.
Cela permet de gérer aisément les tâches à effectuer après une collision pour chaque type d'objet. Pour la plupart des objets il s'agit de leur affecter l'état *OBJECT_STATE_DEAD* (ou *OBJECT_STATE_DESTROYED* pour le missile par exemple), quant à l'oiseau cela permet de gérer son clignotement entre autres.

Pour finir, nous avons rajouté l'affichage d'un texte signalant que notre oiseau à été touché par un ennemi ou qu'il a été en contact avec le haut de l'écran. Une fonctionnalité qui n'était pas demandée dans les feuilles de TD mais qui était présente dans l'animation du rendu.
Ceci nous a mené à rajouter des champs gérant l'opacité et un rapport de taille d'affichage dans notre objet pour contrôler tout ça plus aisément.