BEN-SAID Walid
LABAYE Paul

RAPPORT DE PROJET

TEMPEST C++ / SDL 2

Fonctionnement général

Nous basant sur des projets de jeux faisant usage de la SDL2, nous avons donc décidé d'adopter une structure basée sur le fonctionnement suivant :

Une classe mère, la classe Game, qui contient tous les objets qui composent le jeu (le terrain, les ennemis, le joueur, les projectiles). Chaque frame, 3 fonctions de Game sont appelées :

- handleEvents(), qui enregistre les entrées utilisateur

- update(), qui met à jour les informations du jeu

- render(), qui affiche les objets en fonction des nouvelles informations

Les objets instanciables du jeu implémentent tous une fonction update() et render(), qui sont appelées lors de l'update() et du render() de la Game.

Difficultés

Nous avons manqué de temps pour finir ce projet, et la version actuelle n'est évidemment pas celle que nous aurions voulu présenter. Beaucoup de temps a été consommé notemment par le « manque d'automatismes ». En effet, n'ayant jamais avant ce semestre, pratiqué en C++, beaucoup de solutions à nos problèmes ont nécessité un temps de recherche et de réflexion important. Cependant, nous sentons clairement une différence d'aisance entre le début du développement et les dernières semaines.

- [Paul] - Les principales difficultés techniques qui m'ont fait perdre du temps étaient liées au polymorphisme des objets de jeu et à la manière dont gérer leurs informations. Au début, seulement . J'ai aussi eu des problèmes liés à l'instanciation et la gestion mémoire du c++. Par exemple, j'ai implémenté un compteur statique de projectiles dans la classe Bullet dans le but de pouvoir limiter le nombre de balles que le joueur peut tirer. Ce compteur descendait alors que le constructeur qui l'incrémentait fonctionnait bien. Cette erreur m'a fait perdre quelques heures, j'avais oublié d'implémenter le constructeur de copie de Bullet, qui était appelé lorsqu'on stockait cette dernière dans bullets_.

- [Walid] - Pour ma part, ayant déjà utilisé la SDL lors de mes études, je n'ai pas eu de problèmes concernant l'utilisation de la librairie. Cependant, n'ayant jamais fait de C++, j'ai eu beaucoup de difficultés avec la gestion de la mémoire lors de l'utilisation de références, ou bien du polymorphisme qui nécessitait par moment des casts dynamiques en classes enfants.

BEN-SAID Walid
LABAYE Paul

Coding styles

Nous avons décidé d'emprunter la convention suivante :

- Les noms de classes commence par une majuscule suivis de lettres miniscules (sauf pour HUD)
- Les noms des attributs de classes se terminent par un _
- -Les noms de variables et de fonctions sont toutes en snakecase (les mots sont en miniscules, délimités par des)
- Les accolades ouvrantes sont sur la même ligne que la déclaration de fonction/classe/structure de contrôle

Choix d'implémentation

Game :

Comme décrit plus haut, un objet Game est instancié dans le main. C'est la classe principale du jeu, c'est elle qui gère les informations logique et l'affichage de tous les objets. Comme l'écran est divisé en 2 parties (le niveau dans un carré centré sur l'écran et le HUD sur un bandeau en haut), la classe Game calcule le rendu du jeu dans texture_, qu'elle applique ensuite à window_ grâce à renderer_ et dst_. Elle contient également un attribut hud_, dont la classe contient elle même ses éléments de rendu. level_, player_, bullets_ et enemies_ sont les éléments dont il faut calculer les informations et afficher. Ces attributs sont initialisés à l'aide de init().

Object :

Les différents points des objets à afficher sur le level est définie par les attributs lane_id_, depth_, v_template_ et vertices_. C'est vertices_ qui contient les coordonnées finales de l'objet sur l'écran. L'idée est la suivante : chaque sous-classe d'objet contient un « template » (les points qu'il faut dessiner, normalisés entre -1 et 1). Il est initialisé lors de l'appel au constructeur de Object par un vecteur de points codé en dur dans chaque .cpp. Lorsqu'on souhaite apporter des modifications à un objet, faire rotate les flippers, par exemple, on modifie v_template directement dans le update, puis on met à jour vertices_ en appliquant les transformations relatives au numéro de lane et à la profondeur à v_template_.

update() :

Cette fonction s'occupe de mettre à jour toutes les informations du jeu en fonction de son état actuel. C'est elle qui instancie les projectiles, les ennemis, mais surtout elle appelle le update de tous les éléments à afficher : level_, enemies_, bullets_ et player_.

render() :

Dans cette fonction, les renders du level et de tous les objets sont simplement appelés sur la target texture_ de game. Avant ça, hud_ dessine dans sa propre texture_, puis vient copier le

résultat au bon endroit sur window_. Chaque lane du terrain étant un objet qui implémente un render(), appelé par celui de level_. Pour le reste, tous les objets appartiennent à la classe Object, qui implémente son render. Certains objets, comme les Flipper ou les Bullet, le réimplémentent.

NB : Nous aurions voulu *in fine* pouvoir gérer tous les objets de la game avec un seul vecteur d'objets.

Déroulement du développement

La première partie du travail a consisté à faire des recherches pour décider d'une structure de projet. Walid a parcouru quelques projets de jeux basés sur la SDL2. Une fois la structure adoptée, nous avons commencé par afficher le terrain. Walid s'est occupé de la lecture dans le fichier, et moi de l'affichage avec la bonne homothétie. Nous nous sommes ensuite penchés sur les relations que devaient avoir les différents objets avec le terrain. Nous avons commencé par afficher le joueur, puis ses projectiles. Ça n'est qu'après que nous avons commencé à travailler sur la classe Object, qui englobe tous les objets du jeu, y compris le joueur et les projectiles. Par la suite, Walid s'est préoccupé de calculer et d'afficher le HUD, puis de séparer les rendus dans 2 textures différentes. Quant à moi, je me suis concentré sur le rendu des objets, notemment des flippers et des bullets. J'ai passé beaucoup de temps à faire fonctionner le changement de lane des flippers, et on peut encore voir quelque défauts sur certains changements.