Bob l'éponge Game

Rapport de projet C++

Soumaia Touil
Esra Turk

Enseignant : Cécile Braunstein

21 janvier 2019

Table des matières

1 Qu'est que QCM Bob L'éponge ?

Le projet C++ donné cette année a pour thème "Yellow". Le code proposé doit respecter plusieurs contraintes (au minimum) :

— 8 classes ;

— 3 niveaux de hiérarchie : Formulation <- Reponse <- Score ;

— 2 fonctions virtuelles différentes : init() de Formulation et init(Personnage p) de Deplacement ;

— 2 surcharges d'opérateurs : operator() de Question et Reponse ;

— 2 conteneurs STL : vector et map ;

— Commenter le code ;

— Pas d'erreurs avec Valgrind ;

— Rendu par dépôt git

— Pas de méthodes/fonctions de plus de 30 lignes ;

— Makefile.

Afin de respecter le thème, nous nous sommes penchées sur les personnages de couleur jaune. Nous avons choisi de créer un jeu tourné sur le monde de Bob L'éponge.

1.1 Pourquoi Bob L'éponge ?

Bob L'éponge est une série télévisée diffusée pour la première fois en 1999. C'est un des dessins animés les plus populaires en France et dans le monde. Elle passionne les petits comme les grands.
Nous avons choisi cette série car elle est connue de tous et a un univers ludique.

1.2 Le jeu

Nous avons décidé d'implémenter un quizz autour du monde de Bob L'éponge. Nous utilisons SDL2 pour créer une application 2D. Voici le principe du jeu.
Le joueur aura a répondre à 10 questions au total et devra choisir entre 3 réponses : a, b ou c.

Tout d'abord, le joueur choisit un personnage. Il a trois choix possibles :

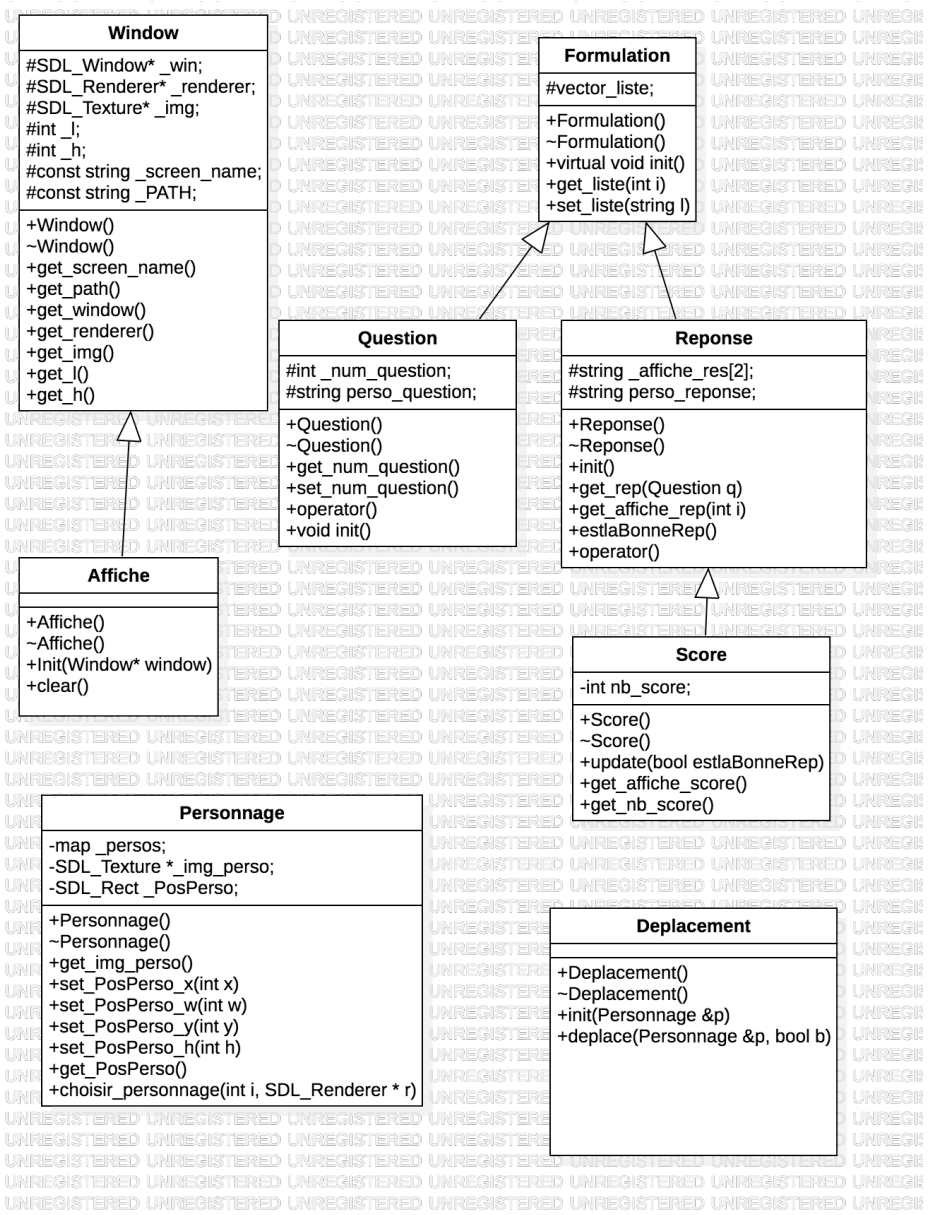
— Bob,

— Patrick,

— Carlo.

Le choix du personnage entraînera son apparition en bas de la page créée. Il permettra aussi d'afficher les questions le concernant. Nous reviendrons sur ce point là plus tard. Ce personnage avancera à chaque bonne réponse et reculera dans le cas contraire.
A la fin du jeu, une image s'affichera témoignant du score total. Elle permettra à l'utilisateur de voir s'il doit mieux se renseigner sur le monde de Bob l'éponge ou si les années passées devant ce dessin animé ont fait du joueur un expert en la matière.

2 Diagramme UML du jeu



3 Les informations utiles à connaître avant de jouer

3.1 Ce qu'il faut télécharger

Pour pouvoir lancer le QCM, il vous faut les bibliothèques suivantes :

1. SDL2
2. SDL2_image
3. SDL2_key

Vous pouvez les télécharger sur le site suivant : https ://www.libsdl.org.

3.2 Les chemins

Plusieurs fichiers contiennent des chemins vers des images. Il faut donc penser à les changer si besoin.

Les fichiers concernés sont les suivants :

1. questions1.txt, questions2.txt et questions3.txt
2. reponses1.txt, reponses2.txt et reponses3.txt
3. Affiche.cc, Window.cc et main.cpp

4 Les classes

Explicitons les classes du programme.

4.1 Questions et Réponses

- **Formulation**

  La classe Formulation est une classe virtuelle et est une classe mère pour Question et Reponse. Elle est composé d'un vector utilisé par les autres classes pour stocker les chemins vers les images.
  Elle est aussi composée d'une fonction virtuelle pure init qui permettra de remplir le vecteur.

- **Question**

  Question hérite de Formulation comme dit précédemment. Elle contient :
  - **init** qui récupère les données d'un fichier donné, perso_question et le copie dans la liste héritée de la classe mère.
  - l'opérateur **()**. Cet opérateur affecte à l'attribut perso_question, le chemin vers le fichier contenant les questions liées au personnage choisi. perso_question.
  - Un setter et un getter

- **Réponse**

  La classe Reponse ressemble beaucoup à celle précédente. On retrouve la fonction init, l'opérateur () (mais avec les fichiers contenant les réponses).
  Nous avons ajouté un booléen qui retourne true si la réponse donnée est bien la vrai false sinon. Elle est utile pour la classe Score et pour afficher l'image vrai ou faux contenu dans l'attribut _affiche_res.

- **Score**

  Score hérite de Reponse. Elle stocke le nombre de bonnes réponses du joueur avec **update**. Elle contient un vecteur, celui hérité de Formulation, composé de trois éléments. A la fin du jeu, si le joueur demande à voir son score (voir section 5), nous affichons une des trois images stockés dans ce fameux vecteur. Cela dépend de la valeur prise par nb_score.

4.2 Affichage de l'écran

- **Window**

  En ce qui concerne l'affichage des fenêtres , la classe Window permet l'initialisation des variables SDL nécessaires à l'affichage d'un écran et des différents composants de cet Écran.

  Cette classe est composé de :

— Variables de type SDL_Window, SDL_Renderer et SDL_Texture. Ces derniers permettent le chargement d'une fenêtre , sur une surface avec un rendu.

— Un constructeur par défaut.

— Un constructeur prenant en entrée les informations nécessaires comme la surface , la fenêtre , le contenu , la largeur, la hauteur, ainsi que le chemin.

— Un destructeur permettant la libération de la mémoire.

Etant donné que tous nos variables sont privé , nous avons mis en place «get_variable» afin de récupérer ces derniers (Exemple : get_l et get_h qui récupère la largeur et la hauteur ...).

Le but principale de la classe Window est de mettre en place les éléments utilisés par la classe Affichage.

- **Affiche**

Le but de la classe Affiche ,qui hérite de la classe Window ,est d'afficher l'écran de la page d'accueil . Cette classe contient des fonctions qui vont permettre l'initialisation et l'affichage d'une fenêtre donné . La fonction principale dans cette classe est «Init», prenant comme paramètre une fenêtre, permet d'initialiser la première page dans lequel s'affiche le menu du jeux. Cette classe comprend :

— Un constructeur par défaut.

— Un constructeur prenant en entrée les informations nécessaires comme la surface , la fenêtre , le contenu , la largeur, la hauteur, ainsi que le chemin.

— Un destructeur permettant la libération de la mémoire.

— Une fonction Init qui initialise la fenêtre, le renderer et l'image ,en récupérant les données d'une fenêtre en paramètre et en utilisant les fonctions IMG_LoadTexture et SDL_QueryTexture

4.3 Le personnage et son déplacement

- **Personnage**

La classe Personnage est celle qui va définir deux points importants du jeu. L'utilisateur doit impérativement choisir un personnage pour pouvoir commencer le QCM. En effet, nous avons créer trois fichiers questions et réponses correspondant aux trois personnages. Personnage est composé principalement :

— D'un constructeur qui initialise l'attribut map avec les chemins vers les images des personnages disponibles. Nous avons Bob l'éponge, son ami Patrick et son voisin Carlo.

— D'une fonction choisir_personnage qui renvoie la texture créée, utilisé pour charger l'image sur l'écran par la suite. Cette fonction apparaît lors de la décision du personnage par le joueur.

— D'un destructeur pour détruire la texture.

— De setter et getter.

- **Déplacement**

La classe Déplacement est en lien avec celle précédente. En effet, elle va permettre au personnage de bouger selon les réponses données. Tout d'abord, nous devons initialiser la position du personnage afin de pouvoir le placer sur la fenêtre. C'est le but de la fonction virtuelle **init**.
La fonction **deplace**, elle, aura pour devoir de faire avancer l'image si la réponse est bonne sinon de reculer. Ainsi, l'utilisateur pourra en déduire son score au fur et à mesure

5 Le traitement des événements

Nous avons décider d'utiliser le clavier de l'ordinateur pour jouer. Nous aurions pu aussi considérer les événements créés par la souris mais nous nous sommes contenter du premier pour plus de facilités. Nous utilisons SDL_Event afin de traiter chaque cas utile et un switch afin de continuer à jouer tant que l'utilisateur ne quitte pas. Le code concernant l'évènement est dans le fichier Events.h. C'est en soi la partie la plus essentielle pour faire fonctionner le programme.

Voici les touches que le joueur peut utiliser :

- SDLK_QUIT SDLK_ESCAPE pour quitter le jeu
- SDLK_UP permet l'affichage de l'image avec les personnages disponibles
- SDLK_1, SDLK_1, SDLK_3 pour choisir un des personnages proposés (Bob, Patrick, Carlo). Ces options sont donc à utiliser après le précédent cas.
- SDLK_RIGHT permet de passer à la question suivante.
- SDLK_a, SDLK_b, SDLK_c pour répondre à la question.
- SDLK_s pour afficher l'image avec le score du joueur. Il ne faut pas oublier de cliquer sur s après avoir répondu à la dernière question.

  Dans main.cpp, nous retrouvons la fonction main composé seulement de l'appel à la fonction Events().

Pour conclure

Enfin tout au long de notre projet, nous avons pu finaliser notre jeu et ajouter petit à petit les éléments conçus au départ. En effet dès le départ de notre projet, nous avons voulu mettre en place un jeu qui prend en compte le déplacement d'un personnage choisi, avec des difficultés sous forme de QCM . Nous avons pu ainsi choisir un animé «Yellow», et choisir les questions en rapport avec le personnage et mettre enfin en place un jeu jouable.

Difficulté rencontrée :

La principale difficulté rencontrée le long de ce projet était lié à l'utilisation de SDL2, en effet la dernière version de cette bibliothèque qui est plus récente et qui n'est pas très simple a utilisé contrairement à l'ancienne version. Néanmoins, un temps a été consacré pour bien comprendre cette librairie afin rendre notre jeu jouable.

Ce qu'on aurait aimé améliorer dans le jeu, si on aurait eu plus de temps, aurait été de rajouter le song à notre jeu afin de le rendre plus dynamique.

Fierté du projet :

Nous sommes très fiers d'avoir surmonté les difficultés rencontrées le long du projet , lié principalement à l'utilisation de la bibliothèque SDL2 et au temps, et d'avoir pu au final avoir un jeu jouable et amusant.