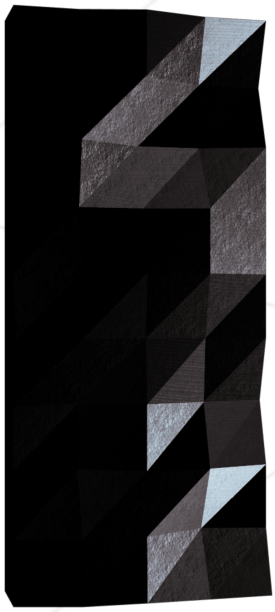

Nibbler

C++ Project

Summary: Le but de ce projet c est de faire votre propre jeu Snake mais avec au moins 3 differentes Interface Graphique. Ces interfaces graphiques seront des libraries partagées.

Contents

Chapter I

Snake

I.1 Introduction

Snake est un jeu classique des années 70. Il s'est retrouvé sur énormement de platforme et a englouti des heures de vie.

I.2 Objectifs

`Nibbler` sera encore un autre Snake mais ecrit en `C++` mais avec un gros twist: des librairies dynamique. A travers ce projet, vous decouvrirez la puissance des libraries dynamique pendant l'execution du programme. Votre but est divisé en 4 parties: Un executable et 3 librairies dynamique que votre executable devra charger au runtime. Chacune de vos librairies devra embarquer tout ce qui est nécessaire pour afficher correctement le jeu. L'executable principale s'occupera de toute la logique du jeu et se servira de la librairie seulement pour l'affichage. L'executable doit communiquer avec chaque librairie toujours de la meme maniere.

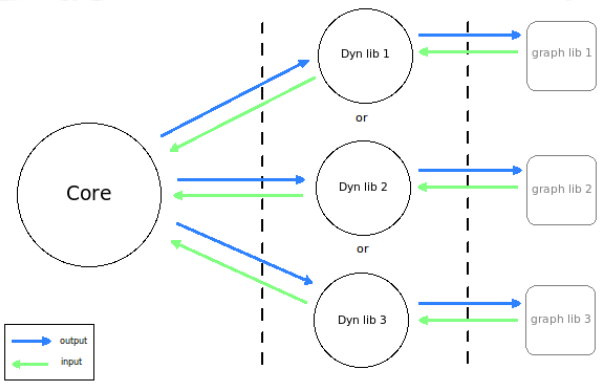

Figure I.1: `Architecture`

I.3 Instructions Generales

- Ce projet sera developpé en `C++`.
- Vous pouvez utiliser n'importe quel compilateur
- Votre code doit compiler avec: `-Wall -Wextra -Werror`.
- Vous pouvez utiliser n'importe quel version de `C++`.
- Vous pouvez utiliser n'importe quel librairie.
- Vous devez fournir un makefile avec les regles habituelles.
- Vos classes qui déclare au moins un attribut doivent etre de forme canonique. Herité d une class qui declare des attributs ne veut pas dire qu on declare des attributs.
- Il est interdit d'implementer des fonctions dans un fichier header à l exception des templates et destructeurs virtuels.
- Le mot clef `"using namespace"` est interdit.
- Important vous **NE DEVEZ PAS** inclure dans votre git une librairie que vous n'avez pas ecrit. Donc si vous avez besoin de librairie externe pensez a fournir un script d installation des dependances. Attention, votre code doit compiler sur une nouveelle session et sur les dumps de l ecole.

I.4 Partie obligatoire

I.4.1 Technique

Vous devez creer votre Snake en 4 partie: l executable et 3 librairies dynamique. Chaque librairie integrera toute la gestion graphique ainsi que la gestion des 'inputs' de votre jeu. Ce qui veut dire qu'il est strictement interdit de gerer les 'inputs' et/ou la moindre partie graphique dans l'executable lui meme. Il en va de meme en sens inverse pour la logique du jeu qui ne doit pas du tout se retrouver dans les librairies.

Votre executable doit prendre en parametre au moins la largeur et la hauteur de la zone de jeu. Pour ce qui est de quel librairie dynamique charger, cela peut etre fait en random sur une liste ou dans l ordre. Peu importe. On doit pouvoir passé d une librairie a l autre en utilisant `1`, `2` and `3`. Cela devrait induire un changement total de l affichage Vous devez aussi gerer les cas suivants:

- Si le nombre d'argument a votre programme est incorrect votre programme devrait afficher un message explicatif et sortir.
- Si la taille de la zone de jeu est 'impossible'(valeur trop petite ou trop grande) votre programme doit afficher une erreur correspondantee et sortir.
- Si une librairie dynamique n existe pas ou n est pas compatible. On s attend a un message d erreur explicite et une sortie en douceur
- Il doit y avoir un moyen de quitter le jeu proprement. Soit un menu ou une touche.

Si vous n'utilisez pas `dlopen`, `dlclose`, `dlsym` and `dlerror` dans votre executable, quelque chose doit clocher.Vos librairies dynamique peuvent etre considéré comme des plug-ins qui gere l'interface graphique et les 'inputs'. Votre executable ne doit avoir aucun code specialisé par librairie. Les interfaces et les abstractions sont les clefs ici.

Vous devriez aussi decouvrir que l'ABI utiliser pour charger vos librairies dynamiques au runtime est en `C` et non `C++`. Par consequent, vous devrez utiliser `extern C`. Ceci n est pas une excuse pour coder en `C` vos librairies et/ou votre executable. Ils doivent etre en `C++` Vous ne devriez utiliser `extern C` que pour les points d'entrée de vos librairies. Sinon le reste doit etre une utilisation de classe. Vous allez decouvrir que de creer une instance d'une classe dans une librairie dynamique et de l'utiliser dans l'executable n'est pas quelque chose de trivial.

Histoire d'etre 100cela veut dire que la boucle de jeu est dans l'executable. Si vous utilisez une librairie graphique qui a sa propre boucle, elle devra etre synchronisé avec celle de l'executable. Mais cela veut surement dit que cette librairie graphique ne convient peut etre pas à ce projet.

I.4.2 Les regles du jeu.

- L'unité de base est une case carré. La taille de la case est a votre convenance. Chaque librairies peut avoir une taille differente.
- La zone de jeu est un ensemble fini de carré. On ne peut passer à travers les bords
- Le serpent demarre avec une taille de 4 case au milieu de la zone de jeu. la direction importe peu
- Le serpent avance automatiquement a vitesse constante/ La vitesse est a votre choix. Chaque case que parcours la tete sera parcourue par le reste du corps.
- Le serpent ne peut reculler
- Le serpent ne peut que tourner a gauche ou a droite avec les touches du clavier.
- Le but du projet est de nourrir le serpent et le faire grandir. La zone de jeu devra toujours avoir au moins une case de nourriture. Cette derniere sera positionée de maniere pseudo aleatoire ou completement aleatoire.
- Une nourriture rempli une case
- Quand la tete du serpent entre dans une case avec de la nourriture la nourriture disparait et le serpent gagne une section supplementaire rajouter a la queue du serpent la prochaine fois que la queue se deplacera.
- Si la tete du serpent tape un mur ou un morceau de lui meme. La partie est terminée.

I.5 Bonus part

Quand vous avez fini avec toutes la parties obligatoire. Vous pouvez attaquer la partie bonus. Voici quelques exemples:

- Faire plusieurs mode de jeu (Changer la vitesse, de la nourriture qui disparait, un systeme de score, mettre des obstacles etc...)
- Du son! Mais attention ceci ne comptera que si celui ci est geré via des librairies dynamiques. Vous n etes pas obligés d en faire 3.
- Un mode multijoueur local.
- Un mode multijoueur via le reseau.

I.6 system de rendu et evaluation

Vous devez rendre le code de votre executable principal et de vos librairies avec les scriptes qui permettent de tout compiler. Votre rangement dans votre dossier doit etre clair. Apres la compilation votre executable et les librairies doivent etre a la racine de

votre repertoire de rendu.

Ne mettez pas de fichier generés dans votre repertoire git. Mettre des images et vos assets est Ok.

Relisez bien la partie Instructions sur les librairies externe.

Enjoy !