Arcade

1- Projet

2- Utilisation

3- Création de librairie Graphique

4- Création de librairie Game

1- Projet

L’arcade est un projet en groupe de 2 à 3. Notre groupe est constitué de 3 personne, Bryan Fortin, Théo Jeanningros et Carlo Guyot.

Le but du projet était de créer 3 librairies graphique et 2 librairies jeu. Pour notre part, nous avons utilisé la SFML, la NCURSE et la SDL pour le graphique et pour les jeux, le jeu Pacman et le jeu Nibbler.

2- Utilisation

Pour lancer l’arcade, il faudra compiler le programme et utiliser la commande si dessous.

USAGE

./arcade x

Description

x = libs/lib\_arcade\_ncurse.so

Pour cet exemple, nous allons utiliser la librairie NCURSE mais nous pouvons très bien lancer avec la librairie SFML ou SDL. Il nous faut juste remplacer ncurse par la librairie qu’on souhaite lancer.

Une fois lancer vous atterrirez sur le menu. Les touches à connaitre :

A = change de librairie graphique en prenant celle qui est placée avant dans le dossier libs

Z = change de librairie graphique en prenant celle qui se situe après dans le dossier libs

Q = quitter

N = lancer le jeu Nibbler

P = lancer le jeu Packam

W = change de librairie jeu en prenant celle qui est placée avant dans le dossier game

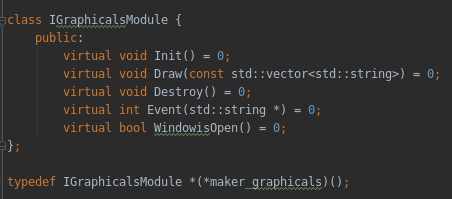
X = change de librairie jeu en prenant celle qui se situe après dans le dossier game

M = revenir au menu

R = restart le jeu

3- Création de librairie Graphique

Tous d’abord, je vais vous expliquer comment marche une classe constante.



Une classe constante ce distingue grâce à : virtual function() = 0. Ceci permet à la fonction de renvoyer NULL et donc de laisser place à la fonction qui se situe dans la classe enfant.



Ceci est une classe enfant, nous la reconnaissons car la class appel son parent avec : public Parent et aussi nous remarquons que nous avons les mêmes fonctions. On peut bien sûr ajouter des fonctions qui ne sont pas dans la classe parent mais il ne faudra pas mettre de virtual. Si la nouvelle fonction crée n’est pas appeler dans les fonctions du parent, elle ne sera pas appelée dans votre programme.

Passons à la pratique, pour créer notre librairie graphique nous devons créer la classe enfant. Dans la classe enfant, il y a 2 points important :

Ajouter toutes les fonctions de la classe parent

Créer une fonction qui retournera notre classe enfant pour la faire devenir parent. (extern “C” parent \*Create()).

Il est très important de l’appeler “Create”. Pour ouvrir une librairie nous allons utiliser dlopen et dlsym. Dlsym va chercher le nom de la fonction qu’on lui donne en paramètre, dans notre programme nous lui donnons “Create”. Si la fonction ne s’appelle pas “Create” la librairie ne s’ouvrira pas et le programme fermera.

La fonction init() sert à initialiser notre librairie graphique. La fonction event() return un int en fonction des evènements effectués. Elle reçoit en paramètre le mode action pour faire pour nos personnages dans nos jeux.

Ex : Up, Down, Left, Right

La fonction destroy() permet de quitter la librairie. La fonction draw() reçois en paramètre un tableau et affiche ce tableau et la fonction WindowisOpen() check si la window est toujours ouvert ou pas.

Les colors pour chaque Charactère :

\* = mur, NCURSE = White, SFML et SDL = img Block

W= Porte, NCURSE = ‘w’, SFML et SDL = img Blockred

\_ = fond, NCURSE = Yellow, SFML et SDL = Cyan

! et q = corps snake, NCURSE = Red, SFML et SDL = img corps

/ = tête snake, NCURSE = Yellow, SFML et SDL = img tête

] = pomme, NCURSE = Blue, SFML et SDL = img pomme

> = packam

+ - ; ? = ghost

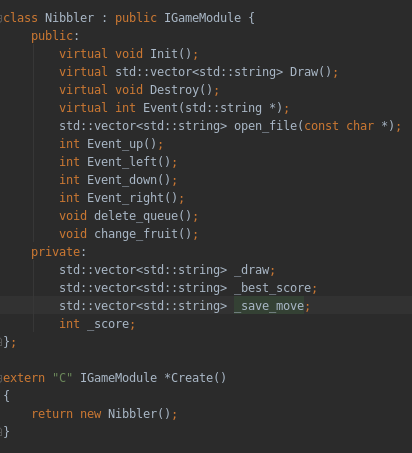
, = tp Packman

. = goûte

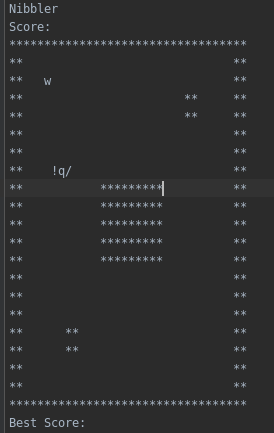
% = food

4- Création de librairie Game

Nous allons utiliser le même système que pour la création de librairie graphique avec une classe parent et une classe enfant.



Voici notre classe enfant, nous retrouvons les fonctions virtual de la classe parent et notre fonction Create. La fonction init() appelle oepn\_file et va initier notre tab draw. Nous utilisons des fichiers textes et nous les implantons dans des vectors.



Ci-dessus nous avons notre ficher text. Dans l’ordre toujours mettre le nom du jeu, le score, la map et le best score. Dans la fonction event(), nous avons en paramètre l’état de notre snake et en fonction de cette état nous appelons la fonction event\_up / down etc. Si notre snake va sur une pomme nous appelons la fonction change\_fruit et nous augmentons le score. Et dans la fonction destroy(), nous vérifions si le score et plus grand que le best\_score, si c’est le cas nous stockons le score.