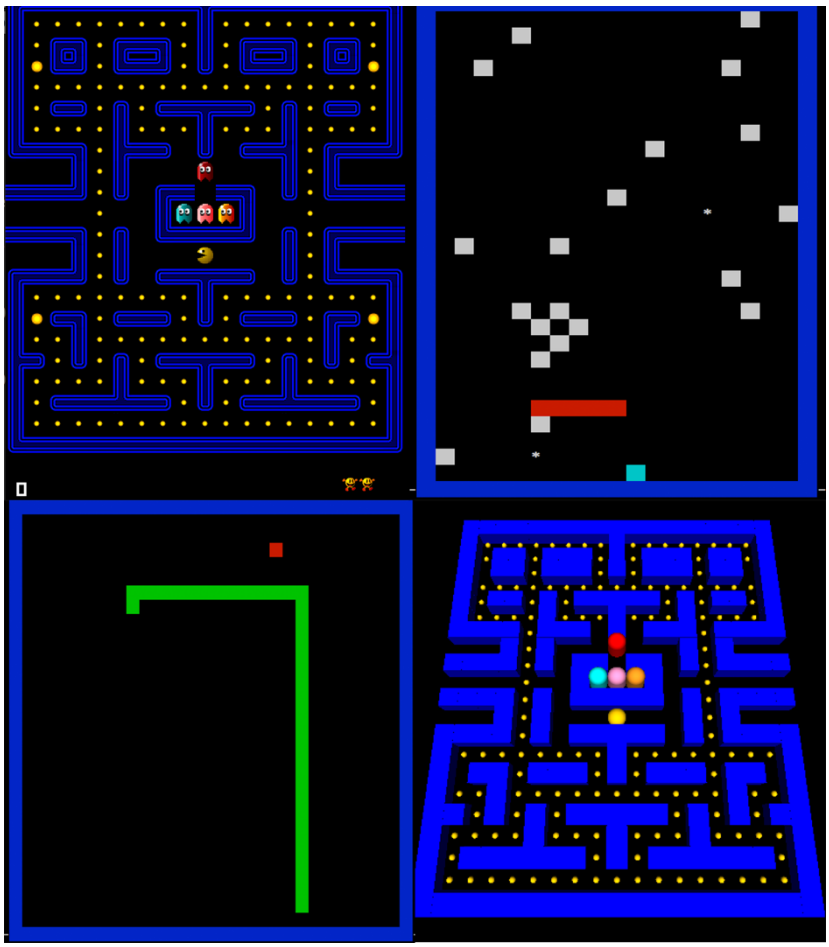

ARCADE

Guide utilisateur

Sommaire

Chapitre 1 : L'Arcade

1.1 Description

L'Arcade est un projet qu'un étudiant de 2ième année à Epitech doit réaliser dans le cadre du module de C++. En effet l'Arcade est un programme codé en C++, un langage de bas niveau permettant d'utiliser ce qu'on appel : "La Programmation orientée objet".

Le programme est une borne d'Arcade virtuelle, celui-ci permet à son utilisateur de jouer à plusieurs jeu dans des interfaces graphiques différentes (SFML, OpenGL ..).

Le projet se déroulant sur 4 semaines, 3 jeux et 3 interfaces graphiques sont déjà disponible, mais le projet ne s'arrête pas là, en effet n'importe quel utilisateur doit être capable grâce a ce guide d'ajouter le jeu ou l'interface graphique de son choix.

1.2 Fonctionnement

L'Arcade fonctionne grâce à des librairies dynamiques, ces librairies sont utilisées pour chaque jeu ainsi que pour chaque interface.

La consigne principale du sujet est de faire en sorte que les librairies graphiques et les libraires des jeux ne communiquent pas entre elles afin d'avoir une indépendance pour celles-ci.

Le tout est donc géré par un organisme extérieur qui a pour rôle d'assembler et de faire communiquer les deux.

De plus, un menu pour l'Arcade est présent, il doit pouvoir être affiché avec chaque interface graphique disponible pour les jeux. L'utilisateur peut, grâce aux arguments du programme, définir l'interface graphique du menu. Une fois sur le menu, l'utilisateur peut choisir son jeu et la l'interface graphique avec laquelle il veut lancer son jeu.

Une fois le jeu lance, l'utilisateur peut s'il le veut changer de jeu ou même d'interface graphique. (ref 1.3)

1.3 Commandes

L'Arcade propose un panel de commandes restreint mais qui permet d'effectuer les principales fonctionnalités du programme.

Touche	Action	Menu	Game
‘2’	Précédente librairie graphique		X
‘3’	Prochaine librairie graphique		X
‘4’	Précèdent jeu		X
‘5’	Prochain jeu		X
‘8’	Relancer le jeu		X
‘9’	Revenir au menu		X
Echap	Quitter	X	X

Ci – dessus, les touches principales liés au programme Arcade, des touches peuvent être ajouté, mais avant tout elle doivent apparaitre dans l’enum “InputButton” du fichier “arcade.hh”.

Pour ce qui est de la gestion de récupération des touches, chaque librairie opère différemment. Il sera donc indispensable de gérer chaque touche si une librairie venait à être ajouté.

1.4 Plateformes

L’Arcade est fonctionnel sur toute plateformes Linux ou OSX ayant installé les paquets des librairies graphiques.

1.5 Utilisation

Voici les différentes règles du Makefile afin de compiler l’Arcade :

Règle	Action
make	Compile les libraries des jeux
make lib	Compile les librairies graphiques
make arc	Compile le Menu et le Core
make everything	Compile tout

Pour l’execution :

- ./arcade ./lib/lib_arcade_xxx.so où **xxx** représente le nom de la lib.

Chapitre 2 : Le Menu

Arcade : Menu en Sfml

2.1 Description

Le menu permet de sélectionner un jeu, une librairie graphique, de saisir un nom ainsi que d'avoir un aperçu sur les meilleurs scores du jeu sélectionné.

Les jeux chargés au démarrage du menu seront ceux présent sous forme de librairie dynamique dans le dossier "games". La même chose s'applique pour les librairies graphiques, mais celles-ci doivent être présentes dans le dossiers "lib".

2.2 Fonctionnement

Le menu s'occupe lui-même de charger les librairies graphiques et les librairies de jeux dans le "Core" (ref 3.1). Il 'occupe également d'afficher le menu dans la librairie graphique passé en argument au programme.

Il ouvre simplement les dossiers et examine si les fichiers correspondent au format d'une librairie graphique ou d'un librairie d'un jeu du Arcade. *Ex : lib_arcade_opengl.so*
Si le format est valide, la librairie est chargée et envoyé directement au "Core".

Chapitre 3 : Le Core

3.1 Description

Le Core permet de lier les librairies graphiques ainsi que les librairies de jeu. En effet c'est lui qui contiendra chaque librairie dynamique, que ce soit les jeux ou les interfaces.

Il s'occupera aussi d'exécuter les fonctionnalités principales de l'Arcade qui sont disponibles grâce aux commandes cités plus tôt (ref 1.3).

Son role est essentiel dans le projet car c'est lui qui sera le cœur du programme, il alternera donc entre phases de calculs pour le jeux et affichage grâce aux librairies graphiques, il permet donc de passer aisément d'une interface a une autre, et d'un jeu a un autre.

3.2 Fonctionnement

Le Core fonctionne grâce à une simple boucle, avec une seule condition : tant que la fenêtre est ouverte. Grâces à ses méthodes pour ajouter des librairies et ses vecteurs pour les contenir, le Core contrôle chaque librairie et peut les faire interagir.

Il s'occupe dans un premier temps de récupérer chaque informations du jeu nécessaire pour l ' affichage (coordonnées des joueurs, positions des objets, score ..), puis il retransmet ces informations à une interface graphique afin que celle – ci puisse afficher les données .

Chapitre 4 : Les jeux

4.1 La classe abstraite AGame

Chaque jeu détient des similarités (Carte, nom, score..), la classe AGame est ici afin de ne pas répéter les mêmes choses. En effet elle doit être hérité par chaque jeu afin de pouvoir être utilisable dans le Core.

Le Core (ref 3.1) agit comme un manager de jeu, et pour cela il est essentiel que chaque jeu ait un type commun auquel se rattaché, c'est le rôle de AGame.

AGame contient un vecteur de ACharacter (ref 4.1), c'est ce vecteur qui permettra de récupérer les positions de chaque élément affichable a l'écran.

4.2 La classe abstraite ACharacter

Chaque objet sur un écran contient des informations similaires (position actuelle, position de départ, nom..), ACharacter est donc la pour regrouper ces informations.

AGame (ref 4.1) s'occupe de gérer des objets de type ACharacter, les déplacer, les éliminer ou encore les faire apparaître. Il n'est donc pas surprenant qu'une pomme qui apparaît sur l'écran dans le jeu Nibbler ai comme héritage ACharacter.

4.3 Ajouter un jeu

Comme dit précédemment le but du programme Arcade est d'être modelable et sans limite, on doit donc être capable d'ajouter n'importe quel jeu au programme.

Tout d'abord chaque nouveau jeu crée doit hériter de la classe AGame (ref 4.1), mais aussi de toutes ses fonctions virtuelles qui seront indispensables pour le Core.

Une fois les méthodes implémentées, il ne vous reste qu'a coder le cœur du jeu comme bon vous semble mais en respectant le fonctionnement de la classe ACharacter pour afficher les objets, personnages, etc dont vous aurez besoin.

Une fois votre jeu opérationnel, il ne vous reste qu'a crée votre libraire dynamique et l'insérer dans le dossier games, pour cela il suffira de créer des règles dans le Makefile à la racine du projet et d'ajouter votre règle à la règle 'all' du Makefile.

Voilà vous avez créer un jeu.

Chapitre 5 : Les Librairies Graphiques

5.1 Interface IDynLib

Comme son nom l'indique il s'agit d'une interface pour les librairies dynamiques, elle détient toute les fonctions nécessaires à chaque nouvelle librairie afin de permettre l'affichage des jeux.

Cette interface est gérée par le Core (ref 3.1) depuis un vecteur contenant les instances de chaque librairies graphiques (Sfml, Opengl, etc).

Chaque nouvelle librairie désirant être insérée dans l'Arcade devra hériter de cette classe et de ses méthodes virtuelles.

5.2 Ajouter une librairie

De la même manière que pour les jeux, il doit être possible d'ajouter n'importe quelle librairie graphique.

Comme expliquer précédemment, l'interface IDynLib est indispensable pour chaque librairie, sans elle il est impossible de faire communiquer le Core avec l'interface graphique.

Une fois les méthodes principales disponible dans IDynLib implémentées, vous pouvez personnaliser votre interface autant que vous le voulez avec un seul prérequis : ne pas faire appel à des classes de vos jeux.

Une fois que toute vos méthodes sont présentes et que la librairie est prête a l'emploi il ne vous reste plus qu'a créer la librairie dynamique et l'insérer dans le dossier 'lib', à l'image des jeux (ref 4.3).
Pour cela le Makefile est à votre disposition, il ne vous reste plus qu'a créer les bonnes règles et les ajouter à la règle lib déjà présente dans le Makefile.

Voilà votre librairie graphique est crée.