EPITECH. TECHNOLOGY

Arcade

B-OOP-400

Adam De-Lacheisserie-Levy

Marius Marolleau

Sommaire :

Qu’est-ce que le projet Arcade ?

Arcade est une plateforme de jeu : un programme qui permet à l'utilisateur de choisir un jeu auquel jouer et qui garde un registre des scores des joueurs. Cela signifie qu'au lieu d'avoir des jeux intégrés directement dans le programme principal, ceux-ci sont des bibliothèques dynamiques distinctes qui sont chargées au moment de l'exécution.

Pour pouvoir manipuler les éléments de votre plateforme de jeu pendant l'exécution, vos bibliothèques graphiques et vos jeux doivent être implémentés en tant que bibliothèques dynamiques, chargées au moment de l'exécution. Cela rend le système plus flexible car de nouveaux jeux peuvent être ajoutés ou modifiés sans avoir à recompiler le programme principal.

Chaque interface graphique utilisateur (GUI) disponible pour le programme doit être utilisée en tant que bibliothèque partagée qui sera chargée et utilisée dynamiquement par le programme principal. Cela permet une modularité accrue, car différentes interfaces utilisateur peuvent être développées et chargées en fonction des besoins ou des préférences des utilisateurs.

Comment fontionne-t-il ?

Pour le lancer, c'est très simple, il faut simplement que les bibliothèques que vous souhaitez utiliser soient dans le dossier "lib/", et ensuite il vous suffit de lancer l'éxécutable suivie du chemin vers la librairie que vous souhaitez avoir pour le menu.

Exemple :

```
./arcade lib/arcade_sfml.so
```

Ici l'arcade à été lancé avec la SFML

Comment ajouter des bibliothèques ?

Pour ajouter une bibliothèque, vous devez tout d'abord coder les fonctions afin de les compiler. Parmis ces fonctions, il doit y en avoir avec comme attribut "extern 'C'" (Ce sera ces fonctions qui créeront les pointeurs sur fonctions plus tard).

Vous devez ensuite grâce à la compilation créer une bibliothèque partagée. Elle doit obligatoirement se nommer de la manière suivante : arcade_[nom_de_la_bibliothèque].so

Ensuite allez dans le fichier Core.cpp puis ajoutez votre bibliothèque en dessous des lignes déjà en place dans le constructeur du Core. Ex : (replacer les noms)

```
if (lib.substr(lib.find("lib/") + 4) == "arcade_[nom_de_la_bibliothèque].so")
    _currentGameGraphicLib = make_unique<[nom_du_Module_associé]>("", lib);
```

Puis dans la fonction assignLib, ajoutez encore une fois une ligne comme les précédentes en remplaçant par le nom de la bibliothèque. De la manière suivante :

```
if (libs.second == "arcade_[nom_de_la_bibliothèque].so")
    _currentGameGraphicLib = make_unique<[nom_du_Module]>(libs.first, "lib/" + libs.second);
```

Si il s’agit d’une bibliothèque graphique, il vous faut aussi l’ajouter à la fonction changeLibrary, ainsi :

```
if (library == "arcade_[nom_de_la_bibliothèque].so")
    this->_currentGameGraphicLib = make_unique<[nom_du_module]>(this->game, "lib/" + library);
```

A savoir que dans l’architecture du projet, il y a deux interfaces (IDisplayModule, IGameModule) ainsi que 2 class abstraites (ADisplayModule, AGameModule)

Dans le constructeur de votre nouveau module, il va falloir récupérer le.s pointeur.s sur fonction.s de.s fonction.s avec “extern C” en attribut, dans un reinterpret_cast<int (*)(I[Display][Game]Module &)>dlsym([le_resultat_de_dlopen], [nom_de_la_fonction_extern]) . Comme ceci :

```
this->snakeAlgo = reinterpret_cast<int (*)(IGameModule &)>(dlsym(this->libFd, "[nom_de_la_fonction_extern]"));
if (!snakeAlgo)
    throw ErrorCore("Invalid function.");
```

Pour finir, allez dans le constructeur du ADisplayModule afin d’y ajouter la nouvelle bibliothèque de la forme suivante {”[nom_de_la_bibliothèque]”, “lib/[nom_de_la_bibliothèque].so”}, dans le vector qu’il faut, soit celui des jeux, soit des graphiques.

Diagramme de l’architecture