B4 - Object-Oriented Programming

B-OOP-400

Arcade

A Retro Platform

Sommaire

Interfaces **3**

Implémenter un Jeu 3

Nommer votre jeu 3

Point d’entrée (entryPointGame) 3

IGameModule 3

init 4

stop 4

restart 4

run 4

getName 6

getHighScore 6

Implémenter une librairie graphique 7

Nommer votre librairie 7

Point d’entrée (entryPointDisplay) 7

IDisplayModule 7

init 8

stop 8

getName 8

catchInput 8

playMusic 8

stopCurrentMusic 9

isOpen 9

clear 9

drawText 9

drawShape 9

getTime 10

restartTime 10

display 10

Diagramme **11**

Interfaces

Cette partie va vous permettre d'ajouter une nouveau jeu, ou une nouvelle librairie graphique, à notre arcade.

Implémenter un Jeu

Nommer votre jeu

Le nom de votre jeu doit commencer par "lib_arcade_" et doit être suivi du nom de votre jeu. L'extension de votre jeu est ".so", vu que ce sera une librairie partagée.

Exemple: lib_arcade_solarfox.so

Point d'entrée (entryPointGame)

Chaque jeu doit comporter un point d'entrée, qui permettra à l'arcade de le charger, pour ensuite pouvoir le lancer. Vous devez donc fournir un point d'entrée, donnant accès à une IGameModule correspondant à votre jeu. Il est donc important que votre jeu hérite de IGameModule.

```
extern "C" IGameModule *entryPointGame(void)
{
   return new GameSolarfoxModule;
}
```

IGameModule

IGameModule est une interface utilisée par l'arcade pour pouvoir échanger avec votre jeu. Voici comment elle s'organise:

```
class IGameModule
{
 public:
```

```
    virtual ~IGameModule() = default;

    virtual void init(const std::string &playerName, const int &highScore) = 0;
    virtual void stop(void) = 0;
    virtual void restart(void) = 0;
    virtual void run(IDisplayModule *library, std::map<Input, bool> keyMap) = 0;

    virtual const std::string &getName() const = 0;
    virtual int getHighScore() const = 0;
};
```

init

Fonction appelée lors du lancement de votre jeu. Elle vous donne accès au highscore de votre jeu, ainsi qu'au nom de celui qui le possède. Elle vous permet d'initialiser toute les variables nécessaires au bon fonctionnement de votre jeu.

stop

Fonction appelée lors de l'arrêt de votre jeu. Elle vous permet de libérer de la mémoire potentiellement allouée dans votre jeu.

restart

Fonction appelée lors du restart de votre jeu.

run

Fonction appelée à chaque trame d'exécution de votre jeu. Elle vous donne accès à la librairie graphique en cours d'utilisation (Voir plus bas dans la documentation), et à la liste des inputs, possiblement saisies par le joueur. Vous avez donc accès à un booléen (True si touche pressée), pour chaque touche disponible, listées dans "Input" (Énumération).

```
enum Input {
  LEFT_ARROW_KEY,
  RIGHT_ARROW_KEY,
  UP_ARROW_KEY,
  DOWN_ARROW_KEY,
  SPACE_KEY,
  RETURN_KEY,
  ERASE_KEY,
```

```
  ESCAPE_KEY,
  A_KEY,
  B_KEY,
  C_KEY,
  D_KEY,
  E_KEY,
  F_KEY,
  G_KEY,
  H_KEY,
  I_KEY,
  J_KEY,
  K_KEY,
  L_KEY,
  M_KEY,
  N_KEY,
  O_KEY,
  P_KEY,
  Q_KEY,
  R_KEY,
  S_KEY,
  T_KEY,
  U_KEY,
  V_KEY,
  W_KEY,
  X_KEY,
  Y_KEY,
  Z_KEY,
  ONE_KEY,
  TWO_KEY,
  THREE_KEY,
  FOUR_KEY,
  FIVE_KEY,
  SIX_KEY,
  SEVEN_KEY,
  EIGHT_KEY,
  NINE_KEY,
  ZERO_KEY
};
```

getName

Fonction appelée pour connaître le nom du jeu en cours d'utilisation. La "string" que vous devez "return" correspond au nom que vous avez mis entre "lib_arcade_" et ".so", comme vu précédemment.

getHighScore

Fonction appelée à la fermeture de votre jeu. Elle doit "return" le plus grand score réalisé pendant la session de jeu.

ATTENTION: Il faut obligatoirement implémenter toute ces fonctions dans votre jeu pour qu'il puisse être correctement exécuté. L'ordre des méthodes de IGameModule est aussi à respecter.

Une fois cela fait, vous n'avez plus qu'à compiler votre code c++ en librairie partagée, et là placer dans le dossier "games", prévu à cet effet.

Si vous ne savez pas comment compiler en librairie partagée, voici un lien pouvant être utile.

Implémenter une librairie graphique

Nommer votre librairie

Le nom de votre librairie doit commencer par “lib_arcade_” et doit être suivi du nom de votre librairie. L’extension de votre librairie est “.so”, vu que ce sera une librairie partagée.

Exemple: lib_arcade_ncurses.so

Point d’entrée (entryPointDisplay)

Chaque librairie doit comporter un point d’entrée, qui permettra à l’arcade de la charger, pour ensuite pouvoir la lancer. Vous devez donc fournir un point d’entrée, donnant accès à une IDisplayModule correspondant à votre librairie. Il est donc important que votre librairie hérite de IDisplayModule.

```
extern "C" IDiaplayModule *entryPointDisplay(void)
{
   return new NcursesModule;
}
```

IDisplayModule

IDisplayModule est une interface utilisée par l’arcade pour pouvoir échanger avec votre librairie. Voici comment elle s’organise:

```
class IDisplayModule
{
   public:
       virtual ~IDisplayModule() = default;

       virtual void init(void) = 0;
       virtual void stop(void) = 0;
       virtual const std::string &getName(void) const = 0;
```

```
        virtual std::map<Input, bool> catchInput(void) = 0;

        virtual void playMusic(std::string) = 0;
        virtual void stopCurrentMusic() = 0;
        virtual bool isOpen(void) = 0;
        virtual void clear(void) = 0;

        virtual void drawText(int, int, const std::string &, int, Color) = 0;
        virtual void drawShape(int, int, std::vector< std::vector<Color> >) = 0;

        virtual float getTime(void) const = 0;
        virtual void restartTime(void) = 0;
        virtual void display(void) = 0;

};
```

init

Fonction appelée lors du lancement de votre librairie graphique. Elle vous permet d'initialiser toute les variables nécessaires au bon fonctionnement de la librairie.

stop

Fonction appelée lors de l'arrêt de votre librairie graphique.

getName

Fonction appelée pour connaître le nom de la librairie en cours d'utilisation. La "string" que vous devez "return" correspond au nom que vous avez mis entre "lib_arcade_" et ".so", comme vu précédemment.

catchInput

Fonction appelée pour savoir quelle touches ont été appuyé. Elle vous retournera un tableau de booléens selon l'énumération des entrées de l'utilisateur (vu précédemment).

playMusic

Fonction à appeler pour jouer de la musique dans votre jeu. Il faut néanmoins fournir le chemin de votre musique en paramètre de la fonction.

stopCurrentMusic

Fonction à appeler pour arrêter la musique actuellement jouée.

isOpen

Fonction à appeler pour obtenir le statut de la fenêtre en cours d'utilisation. Cette fonction "return" True quand la fenêtre est ouverte, et False à l'inverse.

clear

Fonction à appeler pour "clear" votre fenêtre.

drawText

Fonction à appeler pour afficher un texte à des coordonnées, d'une taille et d'une couleur donnée. Cette dernière fait d'ailleurs partie de l'énumération Color, présentée ci-dessous :

```
enum Color {
   NONE = -1, //transparent
   BLACK = 0, //noir
   RED = 1, //rouge
   GREEN = 2, //vert
   YELLOW = 3, //jaune
   BLUE = 4, //bleu
   MAGENTA = 5,
   CYAN = 6,
   WHITE = 7 //blanc
};
```

drawShape

Fonction à appeler pour afficher une forme à coordonnées, données et une forme. Cette dernière correspond à plusieurs vecteurs de "int", listant des couleurs à afficher (Voir Color).

exemple :

```
std::vector< std::vector<int> > shape = {
        {NONE, WHITE, NONE},
        {WHITE, WHITE, WHITE},
        {RED, NONE, RED}
};
```

getTime

Fonction à appeler lorsque vous désirez connaître le temps écoulé depuis le dernier appel de restartTime ou si elle n'a jamais été appelée le temps écoulé depuis le lancement du jeu.

restartTime

Fonction à appeler pour remettre le temps "return" par getTime à zéro.

display

Fonction à appeler pour afficher tous les éléments ajouté ou enlevé à votre fenêtre depuis l'ancien appel de la fonction "display".

ATTENTION: Il faut obligatoirement implémenter toute ces fonctions dans votre jeu pour qu'il puisse être correctement exécuté. L'ordre des méthodes de IDisplayModule est aussi à respecter.

Une fois cela fait, vous n'avez plus qu'à compiler votre code c++ en librairie partagée, et là placer dans le dossier "lib", prévu à cet effet.

Si vous ne savez pas comment compiler en librairie partagée, voici un lien pouvant être utile.

Diagramme

Diagramme représentatif de l’architecture de l’arcade