DOCUMENTATION

B4- C++

B-PAV-330

Arcade

A modular game platform

EPITECH.
L'ECOLE DE L'INNOVATION ET DE L'EXPERTISE INFORMATIQUE

TABLE DES MATIERES

1 – La borne d'arcade.

1.1 – Principe et utilisation de la borne.

2 – Création d'une nouvelle librairie graphique.

2.1 – Interface à respecter.

2.2 – Principe des fonctions.

3 – Création d'un nouveau jeu.

3.1 – Interface à respecter.

3.2 – Principe des fonctions.

4 – Créer un loader graphique.

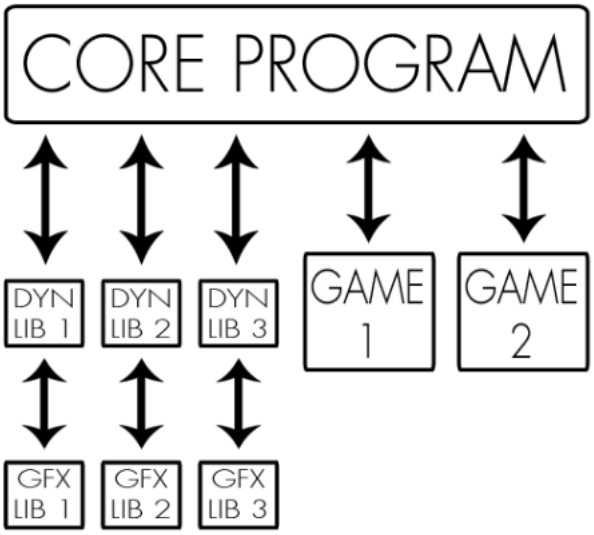
4.1 – Interface à respecter.

4.2 – Principe des fonctions.

1 – La borne d’arcade.

1.1 – Principe et utilisation de la borne.

Arcade est un programme qui laisse l’utilisateur choisir un jeu à jouer et sauvegarder son score. Les jeux sont compatibles avec plusieurs librairies graphiques créer selon une interface.



Le Projet Arcade en lui-même est un menu :

En effet Le programme Arcade ne fait que lire des librairies dynamiques (.so) stocker dans des dossiers particuliers.

De cette façon le programmes est très modulable et permet un développement sur le long terme.

Le binaire du programme se nomme « arcade » et ne prendre en paramètres qu’un seul argument :

Le chemin relatif de la lib graphique servant de base, par exemple :

- ./arcade ./lib/lib_arcade_opengl.so

Lorsque le programme est en cours les touches clavier suivantes sont utilisables :

- ‘2’ → Librairie graphique précédente.
- ‘3’ → Prochaine Librairie graphique.
- ‘4’ → Jeu précédent.
- ‘5’ → Prochain jeu.
- ‘8’ → Relance le jeu.
- ‘9’ → Retour au menu principal.
- ‘Échappe’ → Quitte le programme.

2 – Création d'une nouvelle librairie graphique.

2.1 – Interface à respecter.

Interface :
class ILib
{
  void init(const std::string& windowName = Arcade) = 0
  void close(void) = 0
  void loadSprite(const std::string& spriteName, const std::string& fileName) = 0
  void blitSprite(int x, int y, const std::string& spriteName, int width, int height) = 0
  void loadFont(const std::string& fileName) = 0
  void blitText(const std::string& text, int x, int y, int size) = 0
  void refresh(void) = 0
  void defCommand(std::vector<std::function<(void(void))>>, arcade::CmdMode) = 0
  void checkEvent(void) = 0
}

Les librairies graphiques doivent aussi contenir la fonction : extern "C" Ilib *create_GFX(void)

Cette fonction retourne un pointeur de la classe de la librairie.

2.2 – Principe des fonctions.

- void init(const std::string& windowName = Arcade) :

La fonction « init » a pour rôle, l'initialisation de la librairie graphique. Comprenant L'ouverture de la fenêtre de jeu et des variables interne à la librairie.

- void close(void) :

La fonction « close » a pour rôle, la fermeture de la librairie graphique. Comprenant la fermeture de la fenêtre et la libération de la mémoire prise par les variables internes de la librairie.

- void loadSprite(const std::string& spriteName, const std::string& fileName) :

La fonction « loadSprite » a pour rôle, le chargement en mémoire d'images (ou autrement appelé « sprite ») pour les ajouter à l'écran plus tard.

- void blitSprite(int x, int y, const std::string& spriteName, int width, int height) :

La fonction « blitSprite » a pour rôle, d'ajouter les images précédemment ajouter en mémoire sur l'écran aux positions x et y et de taille width et height.

2 – Création d'une nouvelle libraire graphique.

2.2 – Principe des fonctions.

- void loadFont(const std::string& fileName) :

La fonction « loadFont » tient le même rôle que la fonction « loadSprite » mais pour des fichiers de police de caractère. A savoir qu'une seule police d'écriture ne peut être charger à la fois.

- void blitText(const std::string& text, int x, int y, int size) :

La fonction « blitText » a pour rôle, l'affichage d'un text (variable text) en position x et y avec une taille de caractère size (variable size).

- void refresh(void) :

La fonction « refresh » a pour rôle, le rafraîchissement de la fenêtre du programme affichant à l'écran les images précédemment charger.

- Void defCommand(std::vector<std::function<(void (void))>>, arcade::CmdMode) :

La fonction « defCommand » a pour rôle, la définition de touche qui selon le mode (CmdMode GAME ou MENU) demander associe des touches qui seront toujours les mêmes à des fonctions préalablement assigner par les jeux.

Pour le mode GAME les touches sont :
2 → Librairie graphique précédente 3 → Prochaine librairie graphique
4 → Jeu précédent 5 → Prochain jeu 8 → Relance le jeu 9 → Retour au menu
« Echappe » → Quitte le programme

Pour le mode MENU les touches sont :
2 → Librairie graphique précédente 4 → Prochaine librairie graphique
4 → Jeu précédent 5 → Prochain jeu « caractère alphanum » → écrie un caractère
« Entrer » → Valide les sélections « supprimer » → efface une lettre
« Echappe » → Quitte le programme

- Void checkEvent(void) :

La fonction « checkEvent » vérifie si une touche du clavier utilisée et lance les fonctions précédemment initialiser.

3 – Création d’un nouveau jeu.

3.1 – Interface à respecter.

Chaque jeu doit contenir ses fonctions pour pouvoir lancer les boucles de jeux :

extern “C” arcade::GameExitState launchGame(arcade::IDlGfxLoader *loader, const std::string& playerName, Ilib *lib)

extern “C” void Play(void);

Un jeu doit aussi contenir la fonction : IGame *create_game(void) :
Cette fonction renvoie un pointeur sur la classe du jeu

3.2 – Principe des fonctions.

- launchgame(arcade ::IDlGfxLoader *loader, const std::string& playerName) :

La fonction « launchGame » a pour but, le lancement de la boucle de jeu principale à l’aide d’un loader de lib graphique permettant de changer en plein jeu de lib graphique, à l’aide du nom du jeu et d’une liste de librairie graphiques.

- Play(void) :

La fonction « Play » n’est utile qu’a la validation du projet dans l’école vous pouvez donc ne pas la créer ou créer une fonction vide.

4 – Créer un loader graphique.

4.1 – Interface à respecter.

```
IDlGfxLoader
{
   arcade::ILib    *getCurrentGfxLib(void);
   arcade::ILib    *getPreviousGfxLib(void);
   arcade::ILib    *getNextGfxLib(void);

   std::vector<std::string>    getGfxLibNames(void) const;
   void        closeOldGfx(void);
}
```

4.2 – Principe des fonctions.

- arcade::ILib   *getCurrentGfxLib(void) :

La fonction « getCurrentGfxLib » a pour but le renvoie d'un pointeur sur la classe de librairie actuellement utiliser.

- arcade::ILib *getPreviousGfxLib(void) :

La fonction « getPreviousGfxLib » renvoie un pointeur sur la librairie graphique précédente et la désigne comme librairie actuellement utilisée.

- arcade::ILib *getNextGfxLib(void) :

La fonction « getNextGfxLib » renvoie un pointeur sur la librairie graphique suivante et la désigne comme librairie actuellement utilisée.