Documentation Arcade

Sommaire :

I. Présentation du cœur du projet

1/ Le projet

Le projet "Arcade" est une plateforme de jeux en C++ qui vise à fournir aux utilisateurs une expérience de jeu interactive et diversifiée, tout en conservant les scores des joueurs. Le cœur du projet repose sur l'utilisation de bibliothèques dynamiques, à la fois pour les jeux et les interfaces graphiques.

L'objectif principal est de développer une application modulaire, où les jeux et les interfaces graphiques peuvent être intégrés ou échangés à l'exécution, favorisant ainsi une grande adaptabilité et évolutivité.

Ce projet met l'accent sur des concepts importants en programmation tels que l'encapsulation, le polymorphisme et l'usage des bibliothèques partagées. Il encourage également une collaboration efficace entre les équipes, en exigeant que les interfaces de bibliothèque graphique et de jeu soient partagées avec d'autres groupes de projet.

Dans l'ensemble, "Arcade" offre une expérience d'apprentissage pratique et approfondie sur la conception et l'implémentation de logiciels modulaires et extensibles en C++.

II. Librairies graphiques / jeux

1/ Présentation des interfaces

```
class IGame {
public:
IGame() = default;
virtual ~IGame() = default;
virtual void stop() = 0;
virtual void start() = 0;
virtual std::vector<ArcadeObject::IObjects *> getObjs(void) = 0;
virtual void inpsend(ArcadeObject::Touche) = 0;
virtual void setname(std::string name) = 0;
virtual std::string nameget(void) = 0;
};
```

L'interface IGame est un contrat pour tous les jeux qui seront intégrés dans le système Arcade. Elle garantit que chaque jeu a les méthodes minimales requises pour être compatible avec la plateforme. Voici un aperçu des méthodes de l'interface :

1. IGame() et ~IGame() : Ce sont le constructeur par défaut et le destructeur virtuel. Ils garantissent une bonne gestion de la mémoire pour chaque instance d'un jeu.
2. stop() et start() : Ces méthodes sont appelées pour arrêter et démarrer le jeu respectivement. Cela permet à la plateforme de contrôler le cycle de vie du jeu.
3. getObjs() : Cette méthode renvoie un vecteur de pointeurs vers des objets de jeu. Elle permet à la plateforme d'accéder aux objets de jeu pour les dessiner ou interagir avec eux.

4. inpsend(ArcadeObject::Touche) : Cette méthode permet de gérer les entrées du joueur. Elle est appelée chaque fois que le joueur appuie sur une touche.
5. setname(std::string name) et nameget() : Ces méthodes permettent de définir et d'obtenir le nom du jeu. Le nom est utilisé pour l'identification du jeu dans la plateforme.

En somme, toute classe de jeu qui implémente l'interface IGame doit fournir ces méthodes, ce qui garantit que la plateforme pourra contrôler le jeu, interagir avec ses objets, gérer les entrées du joueur, et identifier le jeu par son nom.

```
class IGraphic {
public:
IGraphic() = default;
virtual ~IGraphic() = default;
virtual void printdraw(void) = 0;
virtual void dispclear() = 0;
virtual void dispupdate(const std::vector<ArcadeObject::IObjects*>& objects) = 0;
virtual void stopdisp() = 0;
virtual void initialisation() = 0;
virtual ArcadeObject::Touche getevents() = 0;
protected:
private:
};
```

L'interface IGraphic définit le contrat pour toutes les bibliothèques graphiques qui seront utilisées dans le système Arcade. Elle garantit que chaque bibliothèque graphique possède les méthodes minimales requises pour être compatible avec la plateforme. Voici un aperçu des méthodes de cette interface :

6. IGraphic() et ~IGraphic() : Ce sont le constructeur par défaut et le destructeur virtuel. Ils garantissent une gestion appropriée

de la mémoire pour chaque instance de bibliothèque graphique.

7. printdraw() : Cette méthode est appelée lorsque la plateforme a besoin de dessiner les objets du jeu à l'écran.
8. dispclear() : Cette méthode est utilisée pour effacer l'affichage actuel, préparant la scène pour le prochain cycle de dessin.
9. dispupdate(const std::vectorArcadeObject::IObjects*& objects) : Cette méthode est utilisée pour mettre à jour l'affichage avec les nouveaux objets de jeu. Elle prend en argument un vecteur de pointeurs vers des objets de jeu.
10. stopdisp() : Cette méthode est appelée lorsque la plateforme a besoin d'arrêter l'affichage, généralement lorsqu'un jeu est terminé ou mis en pause.
11. initialisation() : Cette méthode est appelée pour initialiser la bibliothèque graphique, configurant tout ce dont elle a besoin pour commencer à dessiner à l'écran.
12. getevents() : Cette méthode est utilisée pour obtenir les événements d'entrée de l'utilisateur, tels que les touches pressées.

En résumé, toute classe de bibliothèque graphique qui implémente l'interface IGraphic doit fournir ces méthodes, ce qui garantit que la plateforme pourra contrôler l'affichage, dessiner les objets de jeu, effacer l'écran, initialiser et arrêter l'affichage, et recevoir les entrées de l'utilisateur.

III. liste des commandes

1/ Présentation de l’interface

13. MENU

- ENTER : Confirme la sélection.
- ESCAPE : Quitte le menu.
- UP/DOWN KEY : Change la sélection.
- LEFT KEY : Reviens au choix précédent.
- LETTERS : Permet d'écrire le nom d'utilisateur.
- RETURN : Supprime une lettre du nom d'utilisateur.

14. GAME

- LEFT/UP/RIGHT/DOWN KEYS : Contrôle les déplacements.
- SPACE : Redémarre le jeu.
- Q : Quitte l'Arcade.
- ESCAPE : Retourne au menu.
- A : Utilise la bibliothèque graphique précédente.
- Z : Utilise la bibliothèque graphique suivante.
- O : Utilise la bibliothèque de jeu précédente.
- P : Utilise la bibliothèque de jeu suivante.