Object Oriented Programming - Arcade

Comment implémenter des libraires graphiques et de jeux ?

Afin que chacun puisse utiliser notre programme « Arcade » avec ses propres libraires, il est nécessaire de suivre une certaine architecture. Seul celles qui implémenteront les fonctionnalités pourront être utilisées sans aucun souci.

- Libraries graphiques:

Chaque librairie graphique doit implémenter l'interface IDisplayModule:

```
class IDisplayModule
{
    public:
        virtual ~IDisplayModule() = default;
        virtual bool createAsset(const std::string &path, const std::string &assetKey) = 0;
        virtual bool createText(const std::string &text, const std::string &textKey) = 0;
        virtual bool drawAsset(const std::string &assetName, int x, int y) = 0;
        virtual bool drawText(const std::string &textName, int x, int y) = 0;
        virtual void refreshWindow() = 0;
        virtual void clearScreen() = 0;
        virtual e_event catchEvent() = 0;
        virtual void createSound(const std::string &path, const std::string &soundKey) = 0;
        virtual void startSound(const std::string &soundKey) = 0;
        virtual void stopSound(const std::string &soundKey) = 0;
};
```

Voici le détail dans l'ordre des méthodes:

- createAsset

Cette méthode sert à créer des assets. Elles prends en paramètre le chemin vers le dossier ou se trouve les assets et le nom de l'asset (sans extension). La librairie graphique doit alors prendre le path, ajouter « /1d/ » ou « /2d » (en fonction de sont type) puis le nom de l'asset et enfin l'extension « .txt » ou « .png ».
Elle stockera alors dans une unordered_map interne l'asset avec sa clé (qui correspond au deuxième paramètre).
Cette méthode retourne un booléen True si tout c'est bien passé et False si une erreur est survenue.

- createText

Cette méthode sert à créer des textes. Elles prends en paramètre le texte a affiché et la clé à lui donner.
La classe stockera alors dans une unordered_map interne le texte avec sa clé (qui correspond au deuxième paramètre).
Cette méthode retourne un booléen True si tout c'est bien passé et False si une erreur est survenue.

- drawAsset

Cette méthode affiche un asset grâce à la clé mit en paramètre et au coordonnées données.
Attention, pour les libraires en 2d, il fait multiplier les coordonnées par 32 car les assets sont en 32x32.
Cette méthode retourne un booléen True si tout c'est bien passé et False si une erreur est survenue.

- drawText

Cette méthode affiche un texte grâce à la clé mit en paramètre et au coordonnées données.
Attention, pour les libraires en 2d, il fait multiplier les coordonnées par 32 car les textes doivent être en adéquation avec les assets.
Cette méthode retourne un booléen True si tout c'est bien passé et False si une erreur est survenue.

- refreshWindow

Cette méthode sert simplement à rafraichir la fenêtre en cours.

- clearScreen

Cette méthode sert à nettoyer et de la rendre vierge de tout assets ou textes affichés.

- catchEvent

Cette méthode renvoie un enum correspondant à l'input qu'elle a récupéré.

- createSound

Cette méthode sert à créer des sons. Elles prends en paramètre le chemin vers le dossier ou se trouve les sons et le nom de celui (sans extension). La librairie graphique doit alors prendre le path, ajouter le nom du son et enfin l'extension « .ogg ».
Elle stockera alors dans une unordered_map interne le son avec sa clé (qui correspond au deuxième paramètre).

- startSound

Cette méthode lance le son lié à la clé donnée.

- stopSound

Cette méthode stop le son lié à la clé donnée.

En plus de ces méthodes la classe contenue dans la librairie graphique doit lancer la fenêtre en 1920x1080 dans son constructeur et la détruire dans son destructeur.

- Libraries de jeux:

Chaque librairie de jeux doit implémenter l'interface IGameModule:

```
class IGameModule
{
    public:
        virtual ~IGameModule() = default;
        virtual displayModule::e_event game() = 0;
        virtual bool initGame(const std::shared_ptr<displayModule::IDisplayModule> &lib) = 0;
        virtual bool setLib(const std::shared_ptr<displayModule::IDisplayModule> &lib) = 0;
};
```

Voici le détail dans l'ordre des méthodes:

- Game

Cette méthode lance la boucle infinie de jeu et retourne un event en fonction de si le joueur souhaite changer de libraires graphiques, de jeux ou tout simplement quitter le programme.

- initGame

Cette méthode sert à initialiser le jeux, ses assets, ses textes, ses sons et la librairie graphiques à associer.

- setLib

Cette méthode sert simplement à changer de librarie graphique pour un rendu graphique différent.

- Les externs:

Afin que le DLLoader du programme puisse charger vos libraires graphiques, celles si doivent implémenter des extern spécifique:

```
extern "C"
{
    std::shared_ptr<nom de votre classe> allocator()
    {
        Return std::make_shared<nom de votre classe>();
    }
}
```

Cette extern va permettre au DLLoader d'aller récupérer une instance de votre classe graphique ou de jeux en appelant la fonction allocator.