B4 - Programmation orientée-objet

Avetand Eliott - Biendine Grégoire - Devloo Alexis

Manuel

Explication du diagramme

ICore

- std::shared_ptr<arcade::displayer::IDisplayer> **initDisplayer**(std::string path) : initialise une librairie graphique grâce à son chemin relatif et la stock en pointeur intelligent dans un vecteur.

- std::shared_ptr<arcade::game::IGame> **initGame**(std::string path) : initialise une librairie de jeu grâce à son chemin relatif et la stock en pointeur intelligent dans un vecteur.

- std::vector<std::string> **scanDisplayers**() : scan le dossiers “lib/” à la recherche des libraires graphiques et renvoie un vecteur contenant leur chemin relatif.

- std::vector<std::string> **scanGames**() : scan le dossiers “lib/” à la recherche des libraires de jeux et renvoie un vecteur contenant leur chemin relatif.

- void **update**(size_t gameIndex, size_t displayerIndex) : modifie les librairies actuellement utilisées (car toutes les librairies utilisées sont stockées dans un vecteur et les variables contenant leur index indique lesquelles sont utilisées).

- void **start**() : boucle principale du jeu, elle redirige et récupère les données nécessaire pour jouer.

- void **fillData**() : récupère toutes les infos données par le jeu et les envoie à la librairie graphique.

IGame

- std::vector<std::string> **getDatas**() : récupère les données sous un format spécifique mentionné dans ‘./documentation.pdf’
- void **setInput**(arcade::KeyboardKeys input) : fourni au jeu les entrées envoyé par le joueur
- void **update**() : actualise les données de la boucle de jeu

IDisplayer

- void **drawMenu**(std::vector<std::string> gamePaths, std::vector<std::string> displayerPaths) : affiche le menu
- void **drawGame**(std::vector<std::string> map) : Affiche le jeu à partir des données du jeu fournis
- void **loadMenuAssets**(std::string actualGamePath, std::vector<std::string> gameLibPath, std::vector<std::string> displayerLibPath) : charge les sprites et textures du menu
- void **loadGameAssets**(std::string actualGamePath, std::vector<std::string> map) : charge les sprites et textures du jeu
- void **destroyWindow**() : détruit la librairie graphique
- bool **isRunning**() : Indique si le jeu est toujours en cours ou non
- arcade::KeyboardKeys **getInput**() : récupère toutes les entrées de l'utilisateur