Projet Arcade:

Summary

Introduction

Notre groupe est composé de 3 membres, Samuel Parayre, Manon Leplanquois, Guillaume CORBET. Le projet arcade est un projet en C++ réalisé à EPITECH en 2ème année.

Le but de ce projet est de réaliser le core d'une arcade fonctionnant avec n'importe quel librairie graphique et librairie de jeux tant que celle-ci suivent les mêmes interfaces. Nous avons donc 3 parties:

1. Core
2. Libs
3. Games

Core

Le core est la partie principale du projet elle va se charger de load et lancer les éléments des librairies graphique et des librairies de jeux.

C'est aussi cette partie qui gère tous ce qui est le menu, la boucle du jeu et de l'affichage.

Libs

La partie libs est celle qui gère les librairies graphique elle est lié à une interface IGraphic, avec cette interface toute les fonctions nécessaires au fonctionnement de la librairie graphique seront obligatoire, le programme ne fonctionnera pas sans ces fonctions.

Games

La partie games est celle qui gère les librairies de jeux elle est lié à une interface IGames, avec cette interface toute les fonctions nécessaires au fonctionnement de la librairie de jeu seront obligatoires, le programme ne fonctionnera pas sans ces fonctions.

Description

Maintenant nous allons vous décrire ces interfaces pour que vous puissez créer vos propre jeux et interface graphique pour les rendre utilisable par notre core.

IGraphic

> void setRotationMap(std::vector<std::vector<std::string>> rotationMap)

Applique la map contenant les informations sur le sens de rotation du sprite de chaque élément mobile du jeu.

> void setAnimationMap(std::map<char, std::string> animationMap)

Applique la map contenant les informations sur le type d'animation de chaque sprite animé du jeu et le nombre d'étapes de l'animation.

> void setSpritesNames(std::map<char, std::string> spritesNames)

Applique la map contenant les informations sur le nom du sprite correspondant à chacun des caractères contenus dans la map du jeu.

> void setMovementDirection(std::map<std::string, char> movement)

Applique la map donnant le sens de déplacement des éléments mobiles du jeu.

> int getEventGraphic()

Récupère les évenements lié au graphique. Renvoie un int lié au nombre sur le tableau ascii.

> int getEventPseudoGraphic()

Récupère les évenements lié au graphique spécialement pour l'entrée du pseudo. Renvoie un int lié au nombre sur le tableau ascii.

> void refreshGraphic()

Actualise le graphique lors d'une modification.

> bool isScreenTooSmallGraphic()

Regarde si l'écran est assez grand pour afficher tous les éléments. Renvoie un bool, true pour trop petit, false pour assez grand.

> void endGraphic()

Ferme tous les éléments du graphique.

> void displayMenuGraphic(int place, std::vectorstd::string *list)

Affiche le menu de jeux et de graphique. Prend comme paramètre un int place qui corresponds à l'emplacement de la flèche de séléction, et un vector d'élements à afficher.

> void displayMenuPseudoGraphic(std::string name)

Affiche le menu de rentrée du Pseudo. Prend comme paramètre une string name qui correspond au name entrée.

```
void displayScreenTooSmallGraphic()
```

Affiche le graphique lorsque l'écran est trop petit.

```
void displayMap(std::vectorstd::string map)
```

Affiche la map du jeu. Prends comme paramètre un vector contenant toute la map.

```
int pause()
```

Gère le menu de pause.

```
void displayScore(int score, std::string highscore)
```

Affiche le score au dessus de la map. Prends comme paramètre un int score correspondants au score actuel à afficher et une string highscore étant le pseudo et le record.

```
void startGraphic()
```

Initialise tous ce qui sera nécessaire au fonctionnement de la librairie graphique.

```
void displayVictory(int score, int place, std::string name)
```

Affiche le menu de Victoire. Prends comme paramètre un int score, étant le score obtenue, un int place étant la flèche de séléction et une string name étant le pseudo du joueur.

```
void displayDefeat(int score, int place, std::string name)
```

Affiche le menu de Défaite. Prends comme paramètre un int score, étant le score obtenue, un int place étant la flèche de séléction et une string name étant le pseudo du joueur.

IGames

```
std::vectorstd::string getMap()
```

Récupère la map du jeu. Renvoie un vecteur de string contenant la map.

```
void createOrigMap()
```

Crée la map contenant les informations concernant les éléments immobiles du jeu (murs) tels que le sens de positionnement du sprite.

```
std::vector<std::vectorstd::string> getOrigMap()
```

renvoie la map crée par la fonction createOrigMap().

```
void move(int x, int y)
```

Fait se déplacer tous les éléments à bouger sur la Map. Prends comme paramètre 2 int x et y étant le vecteur de déplacement du Player.

```
std::map<std::string, char> getMovements()
```

Renvoie la map contenant les informations des mouvements des entités mobiles du jeu.

```
std::map<char, std::string> getSpritesNames()
```

Renvoie la map contenant la liste des sprites du jeu.

```
std::map<char, std::string> getAnimationMap()
```

Renvoie la map contenant les informations sur l'animation des entités mobiles du jeu.

```
void setX(int x)
```

Setter pour la variable x du vecteur de déplacement.

```
void setY(int y)
```

Setter pour la variable y du vecteur de déplacement.

```
int getX()
```

Getter pour la variable x du vecteur de déplacement.

```
int getY()
```

Getter pour la variable y du vecteur de déplacement.

```
void setLast(struct timeval last)
```

Setter pour le chrono de déplacement.

```
struct timeval getLast()
```

Getter pour le chrono de déplacement.

```
void setScore(int score)
```

Setter pour le score.

```
int getScore()
```

Getter pour le score.

```
bool isWin()
```

Récupère si le player à gagné. Renvoie true si il a gagné false si il n'a pas gagné.

```
bool isLost()
```

Récupère si le player à perdu. Renvoie true si il a perdu false si il n'a pas perdu.

```
void initGame()
```

Initialise tous les éléments du jeu.

```
void restartGame()
```

Réinitialise tous les éléments du jeu en gardant le score.

> void getHighscoreFile()

Récupère le record stocké dans le fichier.

> void setHighscoreFile(std::string highscore)

Met à jour le fichier avec le nouveau record.

> std::string getHighscore()

Getter pour le highscore.

> void setHighscore(std::string highscore)

Setter pour le highscore.