ARCADE DOCUMENTATION

Développé par Steven Deffontaine et Lucka Valtriani

I - Ajoutez une nouvelle librairie de jeux :

A - Explication

Il faut ajouter dans le dossier games (src/games/) le dossier de votre jeu.

La classe de votre jeu doit hériter de la classe IGame qui est l'interface des jeux avec toutes les fonctions utiles et nécessaires au fonctionnement des jeux de l'Arcade.

Il faudra donc les réécrire dans la Class de votre jeu.

Il faudra aussi ajouter une fonction entryPointGame qui permet au core de communiquer entre le core et le jeu.

B - Fonctions du jeu

```
extern "C" IGames *entryPointGame()
```

entryPointGame()

-> Elle crée le jeu

```
void Nibbler::init()
```

init()

-> Elle initialise le jeu avec toutes les données nécessaires (score, level, nom du jeu, map, etc)

```
void Nibbler::stop()
```

stop()

-> Elle permet d'arrêter le jeu comme un mode pause en modifiant simplement la variable de la boucle du jeu

```
void Nibbler::inputs(char c)
```

inputs()

-> Elle renvoie simplement l'entrée d'une touche utilisée, pratique pour gérer les touches du jeu (z, q, s, d, etc). Cette fonction permet au core de récupérer les touches de déplacements

```
const std::string &Nibbler::getName() const
```

getName()

-> Renvoie simplement le nom du jeu

```
const int &Nibbler::getScore() const
```

getScore()

-> Renvoie simplement le score du joueur jouant au jeu

```
const int &Nibbler::getLife() const
```

getLife()

-> Renvoie simplement le nombre de vie du joueur jouant au jeu

```
const int &Nibbler::getLevel() const
```

getLevel()

-> Renvoie simplement le niveau de la partie auquel le joueur joue au jeu

```
std::vector<std::string> Nibbler::getMap()
```

getMap()

-> Renvoie la map du jeu

```
bool Nibbler::isInfinite()
```

isInfinite()

-> Renvoie vrai ou feu simplement si c'est un jeu avec une boucle ou un jeu avec des déplacements avec des entrées

```
const std::map<char, std::pair<std::string, int>> &Nibbler::getSprite()
```

getSprite()

-> Renvoie simplement la variable contenant tous les sprites du jeu

```
const std::vector<std::tuple<std::string, std::pair<int, int>, std::tuple<int, int, int>>> &Nibbler::getText()
```

getText()

-> Renvoie simplement la variable contenant tous les textes du jeu

II - Ajoutez une nouvelle librairie graphique :

A - Explication

Il faut ajouter dans le dossier lib (src/lib/) le dossier de votre librairie graphique

La classe de votre librairie graphique doit hériter de la classe IGraphics qui est l'interface des librairies graphiques avec toutes les fonctions utiles et nécessaires au fonctionnement des affichages graphiques de l'Arcade.

Il faudra donc les réécrire dans la Class de votre jeu.

Il faudra aussi ajouter une fonction entryPointGraph qui permet au core de communiquer entre le core et le jeu.

B - Fonctions du jeu

```
extern "C" IGraphics *entryPointGraph()
```

entryPointGraph()

-> Elle crée la librairie graphique

```
void Arcade::SDL::createWindow()
```

createWindow()

-> Cette fonction permet d'initialiser tout selon a besoin la librairie graphique pour fonctionner et créer la fenêtre.

```
void Arcade::SDL::destroyWindow() const
```

destroyWindow()

-> Cette fonction permet de supprimer et détruire tout ce qui a été initialisé dans la fonction createWindow ce qui permet de ne pas avoir de problème de mémoire ou de surcharge au niveau du processeur.

```
char Arcade::SDL::getInput()
```

getInput()

-> Elle permet de gérer toutes les entrées clavier ainsi que les événements.

```
void Arcade::SDL::drawSprite(const std::string &text, int pos_x, int pos_y) const
```

drawSprite()

-> Cette fonction appelle tout simplement toutes les fonctions de votre librairie graphique nécessaires pour afficher des sprites sur votre fenêtre.

```
void Arcade::SDL::drawRect(int pos_x, int pos_y, int r, int g, int b) const
```

drawRect()

-> Cette fonction appelle simplement toutes les fonctions de votre librairie graphique nécessaires pour afficher des rectangles de couleur sur votre fenêtre.

```
void Arcade::SDL::sendMap(const std::vector<std::string> &map, std::vector<std::tuple<std::string, std::pair<int, int>, std::tuple<int, int, int>>> text)
```

sendMap()

-> Cette fonction va tout simplement afficher votre carte de jeu avec vos sprites ou vos rectangles. Elle gère également l'affichage des textes de vos jeux ou du menu.

III - Diagramme explicatif du projet :