ARCADE documentation

Comment implémenter une nouvelle librairie graphique ou de jeu

Librairie graphique :

- crée un dossier dans lib/ qui porte le nom de la nouvelle librairie

- crée dans ce dernier 2 dossier (src/ & include/) ainsi qu'un makefile respectant ce format :

```
CC = g++
CXXFLAGS = -Wall -Wextra -Werror -std=c++20 -I ./include -fPIC
LDFLAGS = -shared [flags de la lib]

SRC_DIR = ./src
SRC = $(SRC_DIR)/Lib[nom de la lib].cpp \
        $(SRC_DIR)/lib.cpp

BUILDDIR = ./obj
OBJ = $(patsubst $(SRC_DIR)/%.cpp, $(BUILDDIR)/%.o, $(SRC))

LIB = arcade_[nom de la lib].so

$(BUILDDIR)/%.o: $(SRC_DIR)/%.cpp
        @mkdir -p $(dir $@)
        $(CC) -o $@ -MD -c $< $(CXXFLAGS)

all: $(LIB)

$(LIB): $(OBJ)
        $(CC) -o $@ $^ $(LDFLAGS)

clean:
        rm -rf $(BUILDDIR)

fclean: clean
        rm -rf $(LIB)

re: fclean all

-include $(OBJ:%.o=%.d)

.PHONY: clean fclean re
```

- dans src/, crée un fichier lib.cpp respectant ce format :

```
#include "../include/Lib[nom de la lib].hpp"

Arcade::IGraphic *ret = nullptr;

__attribute__((constructor))
void create()
{
    ret = new Arcade::Lib[nom de la lib];
}

__attribute__((destructor))
void destroy()
{
    delete ret;
}

extern "C" Arcade::IGraphic *createGraphic()
{
    return ret;
}
```

- dans src/, crée un fichier Lib[nom de la lib].cpp respectant ce format :

```
#include "../include/Lib[Nom nouvelle lib].hpp"

namespace Arcade {
    Lib[Nom nouvelle lib]::Lib[Nom nouvelle lib]()
    {
        [créationd es objet de l'énum entities]
    }

    Lib[Nom nouvelle lib]::~Lib[Nom nouvelle lib]()
    {
        [destructeur de la lib]
    }

    Keys Lib[Nom nouvelle lib]::getEvents()
    {
        [récupère et renvoie les touche du clavier préssées avec les valeurs adéquat selon l'enum Keys]
    }

    std::string Lib[Nom nouvelle lib]::getName() const
    {
        [renvoie le nom de la lib graphique utilisée]
    }

    void Lib[Nom nouvelle lib]::drawFood(int i, int j)
    {
        [affiche de la nourriture]
    }

    void Lib[Nom nouvelle lib]::drawPlayer(int i, int j)
    {
        [affiche un joueur]
    }

    void Lib[Nom nouvelle lib]::drawNothing(int i, int j)
    {
        [affiche du vide]
    }

    void Lib[Nom nouvelle lib]::drawWall(int i, int j)
    {
        [affiche un mur]
    }

    void Lib[Nom nouvelle lib]::drawDoor(int i, int j)
    {
        [affiche une porte]
    }

    void Lib[Nom nouvelle lib]::drawPup(int i, int j)
    {
        [affiche un powerup]
    }

    void Lib[Nom nouvelle lib]::drawEnnemy(int i, int j)
    {
        [affiche un ennemi]
    }

    void Lib[Nom nouvelle lib]::draw(Matrix<Entities> &mat, int score)
    {
        [parcoure la matrice du jeu et affiche les élément de l'enumeration entities selon celles rencontrer dans le processus]
    }

    bool Lib[Nom nouvelle lib]::isOpen()
    {
        [renvoie l'etat de la fenetre]
    }

    void Lib[Nom nouvelle lib]::close()
    {
        [ferme la fenetre]
    }

    void Lib[Nom nouvelle lib]::open()
    {
        [met l'etat de la fenetre de la lib a ouvert]
    }

    void Lib[Nom nouvelle lib]::display_list(const std::vector<std::string>& list, size_t selected_index)
    {
        [afficher chaque element de la liste]
    }

    void Lib[Nom nouvelle lib]::drawMenu(std::vector<std::string> games_list, std::vector<std::string> graphics_list, int selected_game, int selected_graphical, std::vector<std::string> scores_list)
    {
        [affiche un menu avec la fonction display_list, à utiliser 2 fois pour le choix des lib graphique et du jeu]
    }

    std::string Lib[Nom nouvelle lib]::get_player_name()
    {
        [recupere les input utilisateur dans une string et renvoie cette dernière]
    }

    void Lib[Nom nouvelle lib]::drawGover()
    {
        [affiche un ecran game over]
    }
};
```

- crée dans include/ un fichier Lib[Nom de la lib].hpp respectant ce format :

```
#ifndef LIB[NOM DE LA LIB]_HPP_
#define LIB[NOM DE LA LIB]_HPP_

#include <[nom de la lib].h>
#include "../../../src/IGraphic.hpp"
#include "../../../src/Keys.hpp"
#include "../../../src/Matrix.hpp"
#include <chrono>
#include <thread>

namespace Arcade {
    class Lib[Nom de la lib] : public IGraphic {
        public:
            Lib[Nom de la lib]();
            ~Lib[Nom de la lib]();
            Keys getEvents() override;
            std::string getName() const override;
            void draw(Matrix<Entities> &, int score) override;
            bool isOpen() override;
            void close() override;
            void open() override;
            void drawMenu(std::vector<std::string> games_list, ...) override;
            std::string get_player_name() override;
            void drawGover() override;
        protected:
            int player;
            int ennemy;
            int food;
            int powerup;
            int wall;
            int door;
            int nothing;
            WINDOW *window;
        private:
            void drawFood(int i, int j);
            void drawPlayer(int i, int j);
            void drawNothing(int i, int j);
            void drawWall(int i, int j);
            void drawDoor(int i, int j);
            void drawPup(int i, int j);
            void drawEnnemy(int i, int j);
            void display_list(const std::vector<std::string>& list, size_t selected_index);
    };
};

#endif /* !LIB[NOM DE LA LIB]_HPP_ */
```

- Une fois ces étapes franchises, appeler le makefile de la nouvelle lib dans le makefile du dossier lib/

Librairie de jeu :

- crée un dossier dans lib/ qui porte le nom de la nouvelle librairie

- crée dans ce dernier un dossier src/ ainsi qu'un makefile respectant ce format :

```
CC = g++
CXXFLAGS = -Wall -Wextra -Werror -std=c++20 -I ./include -fPIC
LDFLAGS = -shared

SRC_DIR = ./src
SRC = $(SRC_DIR)/[Nom du jeu].cpp \
        $(SRC_DIR)/lib.cpp

BUILDDIR = ./obj
OBJ = $(patsubst $(SRC_DIR)/%.cpp, $(BUILDDIR)/%.o, $(SRC))

LIB = arcade_[nom du jeu].so

$(BUILDDIR)/%.o: $(SRC_DIR)/%.cpp
        @mkdir -p $(dir $@)
        $(CC) -o $@ -MD -c $< $(CXXFLAGS)

all: $(LIB)

$(LIB): $(OBJ)
        $(CC) -o $@ $^ $(LDFLAGS)

clean:
        rm -rf $(BUILDDIR)

fclean: clean
        rm -rf $(LIB)

re: fclean all

-include $(OBJ:%.o=%.d)

.PHONY: clean fclean reLIB]_HPP_ */
```

- dans src/, crée un fichier lib.cpp respectant ce format :

```
#include "[Nom du jeu].hpp"

Arcade::IGame *ret = nullptr;

__attribute__((constructor))
void create()
{
    ret = new Arcade::[Nom du jeu];
}

__attribute__((destructor))
void destroy()
{
    delete ret;
}

extern "C" Arcade::IGame *createGame()
{
    return ret;
}
```

- dans src/, crée un fichier [Nom du jeu].cpp respectant ce format :

```
#include "[Nom du jeu].hpp"
namespace Arcade {
    [Nom du jeu]::[Nom du jeu]() : data(Matrix<Entities>(Entities::nothing, 30, 30))
    {
        [crée la matrice pour le jeu]
    }

    [Nom du jeu]::~[Nom du jeu]()
    {
        [destructeur du jeu]
    }

    void [Nom du jeu]::updateMatrix(Keys &key)
    {
        [met a jour les valeurs de la matrice selon les action du joueur]
    }

    std::size_t [Nom du jeu]::getScore() const
    {
        [renvoie le score]
    }

    std::string [Nom du jeu]::getName() const
    {
        [renvoie le nom du jeu]
    }

    Matrix<Entities> [Nom du jeu]::getMatrix() const
    {
        [renvoie la matrice du jeu pour que le processus graphique puisse l'afficher]
    }

    bool [Nom du jeu]::randomApple()
    {
        [place la nourriture de manière aléatoire]
    }

    Direction [Nom du jeu]::keyToDirection(const Keys &key)
    {
        [change la direction dans laquelle le joueur va selon la touche préssée (récupérer par la lib graphique choisie)]
    }

    std::pair<int, int> [Nom du jeu]::getDirection(const Keys &key)
    {
        [recupere la direction dans laquelle le joueur va]
    }

    bool [Nom du jeu]::isOver()
    {
        [renvoie si la partie est terminée]
    }
}
```

- dans src/, crée un fichier [Nom du jeu].hpp respectant ce format :

```
#ifndef [NOM DU JEU]_HPP_
#define [NOM DU JEU]_HPP_

#include "../../../src/IGame.hpp"
#include <cstdlib>
#include <ctime>
#include "utility"
#include "vector"
#include "thread"
#include <chrono>

namespace Arcade {
    enum Direction {
        up = 1,
        down = -1,
        right = 2,
        left = -2
    };
    class [Nom du jeu] : public IGame {
        public:
            [Nom du jeu]();
            ~[Nom du jeu]();
            void updateMatrix(Keys &) override;
            std::size_t getScore() const override;
            std::string getName() const override;
            Matrix<Entities> getMatrix() const override;
            bool isOver() override;
        protected:
        private:
            Matrix<Entities> data;
            std::size_t score;
            Direction _direction = right;
            std::vector<std::pair<size_t, size_t>> _snake;
            std::pair<std::size_t, std::size_t> headPos;
            bool _isOver = false;
            bool randomApple();
            std::pair<int, int> getDirection(const Keys &);
            Direction keyToDirection(const Keys &);
    };
};
#endif /* ![NOM DU JEU]_HPP_ */
```

- Une fois ces étapes franchises, appeler le makefile du nouveau jeu dans le makefile du dossier lib/