Arcade Documentation

Classe: Core

Cette classe représente le cœur du programme, il permet de lancer la librairie que aura été chargé préalablement par l'utilisateur, de lancer les fonctions de jeu et d'appeler le menu. C'est par cette classe que tout va se dérouler.

```
class Core {
    public:

        // Ctor && Dtor
        Core(std::string path);
        ~Core();

        // Functions
        int arcade();

        // Getter && setter
        arcade::IDisplayModule *getDisplayModule() const;
        arcade::IGameModule *getGameModule() const;
        arcade::DlLoader<arcade::IDisplayModule> getDisplayLoader() const;
        arcade::DlLoader<arcade::IGameModule> getGameLoader() const;
        void setDisplayModule(arcade::IDisplayModule *newDisplay);
        void setGameModule(arcade::IGameModule *newGame);

    private:
        // variables
        arcade::DlLoader<arcade::IDisplayModule> _displayLoader;
        arcade::DlLoader<arcade::IGameModule> _gameLoader;
        std::unique_ptr<arcade::IDisplayModule> _displayModule;
        std::unique_ptr<arcade::IGameModule> _gameModule;
        arcade::Menu _menu;
        ScoresManager _scoreManager;

        // Functions
        bool checkInput(std::vector<arcade::Inputs> const& inputs, arcade::Inputs checkInput) const;
        std::chrono::duration<double> getElapsedTime(
            std::chrono::time_point<std::chrono::system_clock> start,
            std::chrono::time_point<std::chrono::system_clock> end
        ) const;
        int playPause(std::vector<arcade::Inputs> & inputs);
        int playMenu();
};
```

Classe: DlLoader

Cette classe permet le chargement, la gestion et la fermeture des jeux et des libs graphiques.

```cpp
template<typename T>
class DlLoader {
    public:

        // Ctor && Dtor
        DlLoader(): libHandle(nullptr) {};
        ~DlLoader()
        {
            if (libHandle) {
                dlclose(libHandle);
            }
        };

        // Template
        T *createLib(const std::string &path)
        {
            libHandle = dlopen(path.c_str(), RTLD_LAZY);
            if (!libHandle) {
                std::cerr << "dlopen: " << dlerror() << std::endl;
                return (nullptr);
            }
            T *(*fptr)();
            T *module;
            fptr = (T *(*)()) dlsym(libHandle, "createObject");
            module = fptr();
            return (module);
        }

        T *reloadLib(const std::string &path)
        {
            if (libHandle) {      Thomas Dalem, 2 days ago • [FIX]: game
                dlclose(libHandle);
            }
            return (createLib(path));
        }

    private:
        void *libHandle;
```

Classe: Menu

C’est la classe qui va permettre de gérer les menus de pause et de lancement du jeu.

```cpp
class Menu {
    public:
        Menu();
        ~Menu();

        bool getChangeLibs() const;
        void setChangeLibs(bool status);
        std::string const& getSelectedGraphLib() const;
        std::string const& getSelectedGameLib() const;
        std::vector<arcade::Element> const& getElements() const;
        std::vector<arcade::Text> const& getTexts() const;
        std::vector<arcade::Text> const& getPauseTexts() const;
        std::vector<arcade::Element> const& getPauseElements() const;
        std::string const& getPlayerName() const;

        void retreiveLibs();
        void retreivePause();
        void playMenu(std::vector<arcade::Inputs> const& inputs, std::string const& textInputs);
        void pauseMenu(std::vector<arcade::Inputs> const& inputs);
        void setDisplayScores(std::vector<Score> const& scores);

    private:
        int manageInputs(std::vector<arcade::Inputs> const& inputs);
        std::string getLibName(std::string const& filename) const;      Thomas Dalem, 3 days ago • [ADD]: a b
        void getPlayerName(std::vector<arcade::Inputs> const& inputs, std::string const& textInputs);
        void displayLibs(
            std::map<std::string, std::string> const& map,
            std::map<std::string, std::string>::iterator &selectedLib,
            double posX
        );

    private:
        std::map<std::string, std::string> _gamesLibs;
        std::map<std::string, std::string> _graphLibs;
        std::map<std::string, std::string> _pauseGraphLibs;
        std::vector<arcade::Element> _elements;
        std::vector<arcade::Element> _pauseElements;
        std::vector<arcade::Text> _texts;
        std::vector<arcade::Text> _pauseTexts;
        std::map<std::string, std::string>::iterator _selectedGameLib;
        std::map<std::string, std::string>::iterator _selectedGraphLib;
        bool _changeLibs;
        bool _pressedContinue;
        std::string _playerName;
};
```

Classe: ScoresManager

La classe qui permet de gérer le score et de les afficher dans le menu de départ.

```cpp
class ScoresManager {
    public:
        ScoresManager();
        ~ScoresManager();

        void addScore(std::string const& name);
        void updateActualScore(int score);
        void registerActualScore();
        std::vector<Score> const& getScores() const;

    private:
        void retreiveScores();
        void writeScores() const;
        void setScorePlayerName(Score &score, std::string const& str) const;

    private:
        Score _actualScore;
        std::vector<Score> _scores;
};
```

Enum: Inputs

Cet enum contient la liste de toutes les touches utilisables dans le jeu et dans le menu.

Implémentation des libs de jeu et graphiques

Libs graphiques

Pour implémenter les libs graphiques, il faut créer la fonction **createObject**() qui permettra de return un constructeur vers la classe de la lib comme ceci:

```
extern "C" SDL2Display *createObject()
{
    return (new SDL2Display);
}
```

Pour que cela fonctionne, il faut que la classe de la lib graphique hérite de l'interface **IDisplayModule**:

```
class IDisplayModule {
    public:
        virtual ~IDisplayModule() = default;

        virtual void display(std::vector<Element> const& elements, std::vector<Text> const& text) = 0;
        virtual std::vector<Inputs> getInputs() = 0;
        virtual std::string getTextInput() = 0;
};
```

Pour que le core les détecte et affiche les libs dans le menu, il faut que le nom de la lib compilée soit du style "l**ib_arcade_GRAPHLIB.so**" et que celle-ci se trouve au chemin "**/lib/NOM_DE_LA_LIB**"

Libs de jeu

Pour implémenter les libs de jeu, il faut créer la fonction **createObject**() qui permettre de return un constructeur vers la classe de la lib comme ceci:

```
extern "C" PacmanGame *createObject()
{
    return (new PacmanGame);
}
```

Pour faire fonctionner la lib dans le projet arcade, il faut que la classe de la lib de jeu hérite de l'interface **IGameModule**:

```
class IGameModule {
    public:
        virtual ~IGameModule() = default;

        virtual int playLoop(std::vector<arcade::Inputs> const& inputs) = 0;
        virtual void restart() = 0;
        virtual std::vector<arcade::Element> const& getElements() const = 0;
        virtual std::vector<arcade::Text> const& getTexts() const = 0;
        virtual bool getIsGame() const = 0;
};
```

La classe Core appellera à chaque tour de boucle la fonction playLoop().

Pour que le core détecte les libs de jeu dans le menu, il faut que les noms des libs compilées soient du style “**lib_arcade_LIBGAME.so**" et que celles-ci se trouvent au chemin:

“**/games/NOM_DE_LA_LIB**”.