Arcade

Cette documentation est à disposition pour vous aider à créer votre librarie graphique ou votre jeu au sein du projet “Arcade”.

Nommer votre binaire

Dans l’état actuel du projet, le nom de votre lib/jeu doit correspondre à une liste donnée, sans quoi iel ne sera pas chargé.e.

Librairies graphiques

- lib/arcade_sfml.so
- lib/arcade_sdl2.so
- lib/arcade_ncurses.so
- lib/arcade_opengl.so
- lib/arcade_libcaca.so
- lib/arcade_ndk++.so
- lib/arcade_aalib.so
- lib/arcade_allegro5.so
- lib/arcade_xlib.so
- lib/arcade_gtk+.so
- lib/arcade_irrlicht.so
- lib/arcade_vulkan.so
- lib/arcade_qt5.so

Jeux

- lib/arcade_nibbler.so
- lib/arcade_pacman.so
- lib/arcade_qix.so
- lib/arcade_centipede.so
- lib/arcade_solarfox.so

💡 N’oubliez pas de mettre votre .so dans le dossier lib/ du projet pour qu’il soit correctement chargé.

Créer une librairie graphique

Compilation

La compilation de votre librairie s'accompagne des arguments `-shared` et `-fPIC`.

```
// Ajout des fichiers source
SRC_NCRUSES   = src/libs/Ncurses.cpp
// Conversion en .o
OBJ_NCURSES   = $(SRC_NCRUSES:.cpp=.o)
// Nom du fichier ('lib/' pour le placer automatiquement dans le dossier lib/)
NAME_NCURSES  = lib/arcade_ncurses.so
// Flags de compilation
NCURSES_FLAGS = -shared -fPIC -lcurses

$(NAME_NCURSES): $(OBJ_NCURSES)
  @g++ $(OBJ_NCURSES) $(NCURSES_FLAGS) -o $(NAME_NCURSES)
```

Template

```
#ifndef MYLIB_HPP_
#define MYLIB_HPP_

#include "ILibrary.hpp"
// Ajoutez ici les includes de la lib graphique

class MyLib : public virtual ILibrary {
private:
    std::string name;
    std::string currentMusic;
    /*type*/    window;

public:
    MyLib();
    ~MyLib() noexcept;

public:
    void        refreshLib() noexcept final;
    void        deleteWindow() noexcept final;
    void        putText(const Text &txt) noexcept final;
    void        clearLib(const Arcade::Color &color) noexcept final;
    void        putSprite(const Sprite &sprite) final;
    bool        createWindow() noexcept final;
    std::string getName() const noexcept final;
    Arcade::Key getInput() noexcept final;
    void        playSound(const std::string &sound) noexcept final;
    void        cleanSound() noexcept final;
};

extern "C" ILibrary *createLib();

#endif
```

Méthodes

bool createWindow() noexcept	*Création d'une fenetre de jeu. return Echec/réussite de la création.*
void deleteWindow() noexcept	*Destruction d'une fenetre de jeu.*
Arcade::Key getInput() noexcept	*Récupère la touche appuyée sous forme d'Arcade::Key. Return un Arcade::Key.*
void clearLib(const Arcade::Color &color) noexcept	*Supprime tous les élément à l'écran avant d'afficher la couleur donnée en paramêtre. color → Couleur de fond.*
void refreshLib() noexcept	*Actualise l'affichage de la lib.*
void putText(const Text &txt) noexcept	*Affiche le Text donné en parametre. (Text → {str, color, x, y underline}).*
void putSprite(const Sprite &sprite)	*Affiche le Sprite donné en parametre. (Sprite → {filepath, color, x, y}).*
void playSound(const std::string &sound) noexcept	*Joue le son donné en parametre. Par défaut, la SFML se charge de lancer la musique pour toutes les librairies. Vous n'avez donc pas besoin d'implementer une logique à cette méthode.*
void cleanSound() noexcept	*clear les sons lancés.*
std::string getName() const noexcept	*Getter pour name.*

> La fonction createLib() retourne un nouvel objet MyLib.

Créer un Jeu

Compilation

La compilation de votre jeu s'accompagne des arguments `-shared` et `-fPIC`.

```
// Ajout des fichiers source
SRC_NIBBLER   = src/games/nibbler/Nibbler.cpp \
                src/games/AGame.cpp           \ // Fichier obligatoire à la compilation
                src/core/Sprite.cpp             // Fichier obligatoire à la compilation
// Conversion en .o
OBJ_NIBBLER   = $(SRC_NIBBLER:.cpp=.o)
// Nom du fichier ('lib/' pour le placer automatiquement dans le dossier lib/)
NAME_NIBBLER  = lib/arcade_nibbler.so
// Flags de compilation
SHARED_FLAGS  = -shared -fPIC

$(NAME_NIBBLER): $(OBJ_NIBBLER)
  @g++ $(OBJ_NIBBLER) -o $(NAME_NIBBLER) $(SHARED_FLAGS)
```

Template

```
#ifndef MYGAME_HPP_
#define MYGAME_HPP_

#include "AGame.hpp"
// Ajoutez vos autres fichiers ici

class MyGame : public virtual AGame {
private:
    // Ajoutez ici vos attributs uniques à votre jeu

public:
    MyGame();
    ~MyGame() noexcept = default;
    void                handleInput(const Arcade::Key &input) noexcept final;
    std::vector<Sprite> getDisplay() noexcept final;
    void                update() noexcept final;
    void                reset() noexcept final;

private:
    Arcade::GameState endGame() noexcept;
    // Ajoutez ici vos méthodes uniques à votre jeu
};

extern "C" IGame *createGame();

#endif
```

Méthodes

void handleInput(const Arcade::Key &input) noexcept	*Récupère l'input de la Gameloop pour lancer les différentes actions du jeu*
std::vector<Sprite> getDisplay() noexcept	*Renvoi un vecteur de Sprites contenant tous les éléments à afficher dans le jeu*
void update() noexcept	*Mets à jour les informations du jeu*
void reset() noexcept	*Remise à 0 des options du jeu*

> La fonction createGame() retourne un nouvel objet MyGame.

Particularités du constructeur

```
// Nom (avec le /lib), couleur du fond, chemin de la musique
MyGame::MyGame() : AGame("lib/arcade_nibbler.so", Arcade::Color::BLACK, "assets/sound/music.ogg")
{}
```