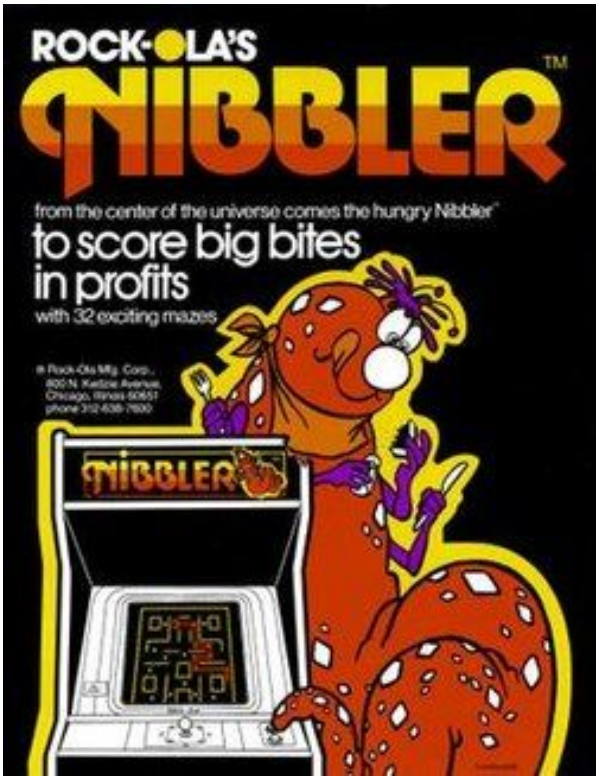
Documentation technique

ARCADE



Créé et testé par :

Victor “Patrick Chirack” Zhu
Stanislav “Cathie Tuche” Stefinyn
Romain “Carlos” Rousseau

Informations sur le programme

Langage : C++
Ligne de compilation: make (via Makefile à la racine du dossier)
Exécution: ./arcade lib/lib_arcade_ncurses.so
Librairies graphiques testées et intégrées :

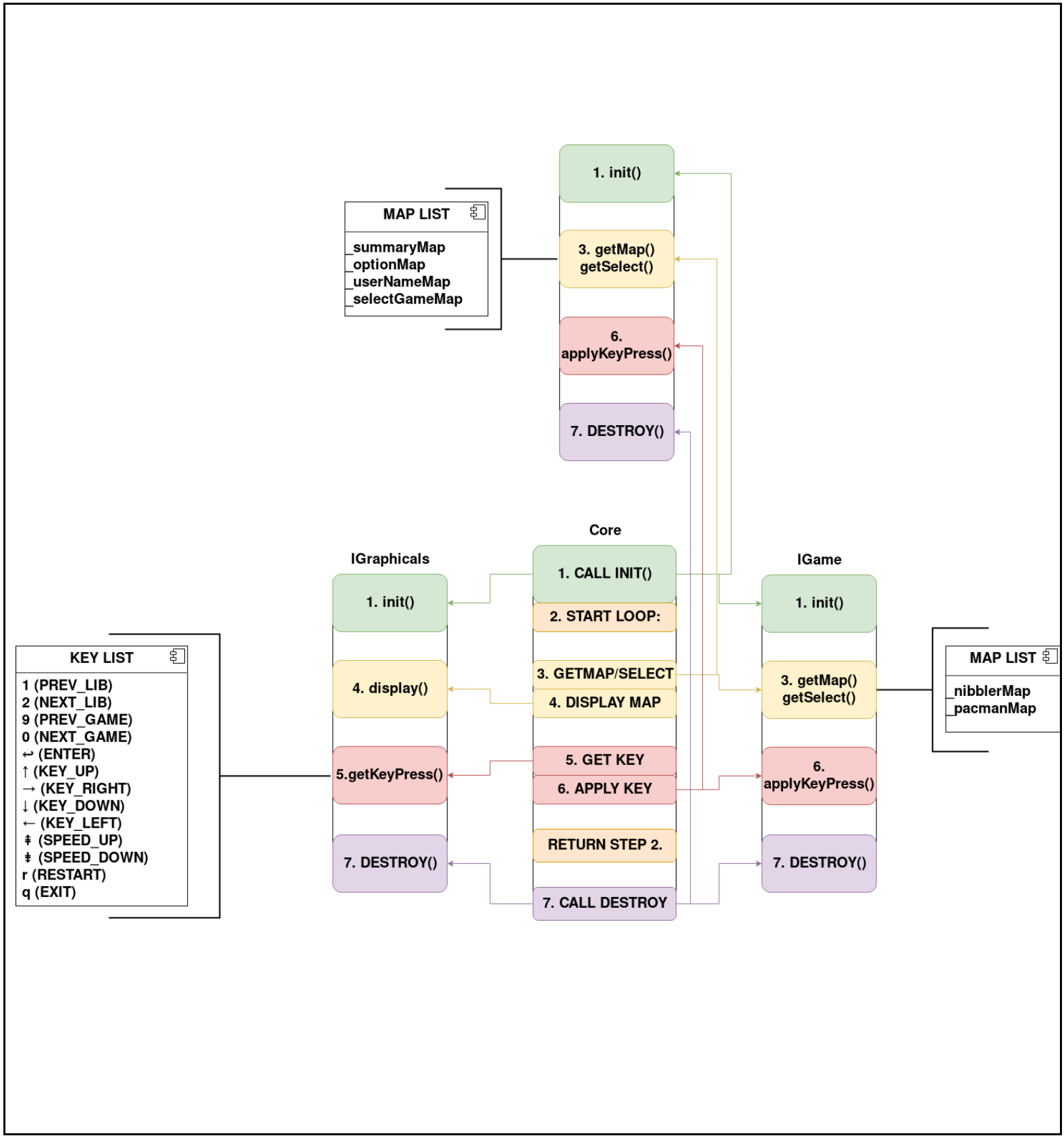
- Allegro
- LibCaca
- Ncurses
- Sfml

Jeux testés et intégrés :

- Pacman
- Nibbler

Organisation du dossier :

```
- arcade
- Makefile (Compile tous les autres Makefiles)
- core (Menu & début du programme)
        - Arcade
                - fichiers .cpp & .hpp
        - main.cpp
- games (Librairies de jeux)
        - IGames.hpp
        - lib_arcade_nibbler.so
        - lib_arcade_pacman.so
        - Nibbler
                - fichiers .cpp & .hpp
        - Pacman
                - fichiers .cpp & .hpp
- lib (Librairies graphiques)
        - IGraphicals.hpp
        - lib_arcade_allegro.so
        - lib_arcade_libcaca.so
        - lib_arcade_ncurses.so
        - lib_arcade_sfml.so
        - Allegro
                - fichiers .cpp & .hpp
        - Libcaca
                - fichiers .cpp & .hpp
        - Ncurses
                - fichiers .cpp & .hpp
        - Sfml
                - fichiers .cpp & .hpp
```



Uml du programme arcade

Intégration de librairies graphiques

Intégration en général

Il faut tout d'abord créer un dossier du nom de la librairie graphique choisie dans le dossier ***./lib***. Puis ensuite, créer trois fichiers à placer dans ce dossier : Makefile, Example.cpp, Example.hpp (où "example" est le nom de la librairie).

Pour intégrer une librairie graphique à notre programme arcade, il suffit simplement de créer une classe qui héritera de la classe IGraphical qui viendra wrapper les fonctionnalités de la librairie graphique choisie. En soit, le principe est simple: il faut adapter la librairie aux fonctions de la classe IGraphical.

Notre interface graphique : IGraphical.cpp

```
#ifndef IGRAPHICAL_HPP_
# define IGRAPHICAL_HPP_

# include <iostream>
# include <vector>

class   IGraphical {
public:
      enum ENTITY {
            PLAYER,
            ENEMY,
            ITEM,
            WALL,
            NONE
      };

      virtual ~IGraphical() = default;

      virtual void init() = 0;
      virtual void display(std::vector<std::string> toDisplay, int selection) = 0;
      virtual void destroy() = 0;
      virtual int  getKeyPress() = 0;
};

typedef IGraphical *create_graph_t();

#endif /* !IGRAPHICAL_HPP_ */
```

Explication des fonctions de la classe IGraphical

- **void init(void);**

La fonction init sert simplement à initialiser la librairie. Dans la plupart des cas, cela se limite à lib_init(); mais parfois il faut aussi initialiser l'écran, les touches, les évènements, le timer, etc ...

- **void display(std::vector<std::string> toDisplay, int selection);**

La fonction display sert à afficher le contenu de toDisplay qui contient nos lignes à afficher.
La variable selection sert à déterminer quelle ligne est surlignée dans notre menu. Dans le cas où selection vaut -1, nous sommes dans le jeu.

- **void destroy(void);**

La fonction destroy sert simplement à détruire les ressources et libérer la mémoire.

- **int getKeyPressed(void);**

La fonction getKeyPressed sert à récupérer et return le numéro de la touche. Le numéro des touches se base sur l'integer renvoyé par la fonction getch de la librairie graphique Ncurses.

Pour nos jeux, nous utilisons les touches suivantes :
1, 2, 9, 0, ↩, ↑, →, ↓, ←, ↧, ↕, r, q

Intégration de librairies de jeux

Intégration en général

Il faut tout d'abord créer un dossier du nom du jeu choisi dans le dossier ***./games***. Puis ensuite, créer trois fichiers à placer dans ce dossier : Makefile, Example.cpp, Example.hpp (où "example" est le nom du jeu).

Pour intégrer une librairie de jeu à notre programme arcade, il faut simplement adapter faire hériter notre nouvelle classe de la classe IGame et adapter les fonctions.

Notre interface graphique : IGames.cpp

```
#ifndef IGAMES_HPP_
# define IGAMES_HPP_

# include <iostream>
# include <fstream>
# include <vector>
# include <string>
# include <array>
# include <algorithm>
# include <thread>
# include <ctime>
# include <cstdlib>
# include <chrono>
# include <iterator>
# include <random>

class   IGame {
public:
      enum DIRECTION {
            UP,
            RIGHT,
            DOWN,
            LEFT,
            NONE
      };

      virtual ~IGame() = default;

      virtual void init() = 0;
      virtual void applyKeyPressed(int key) = 0;
```

```
        virtual unsigned int getSelect() = 0;
        virtual std::vector<std::string> getMap() = 0;
        virtual int getAction() = 0;
        virtual void setUsername(std::string username) = 0;
};

typedef IGame *create_game_t();

#endif /* !IGAMES_HPP_ */
```

Explication des fonctions de la classe IGames

- **void init(void);**

La fonction init sert à initialiser toutes les informations, attributs dont on aura besoin dans notre jeu (comme la map, le score, le timer, la position des ennemies, etc.). Ainsi, lorsqu'on lance le jeu tout est déjà prêt !

- **void applyKeyPress(int key);**

Cette fonction permet d'appliquer la touche sur laquelle on a appuyé et ainsi de faire toutes les modifications qu'il faut dans le jeu. Par exemple, si on appuie sur la touche du haut (UP) alors la direction du joueur va etre mise a 'UP' et des modifications vont être faites pour que le joueur monte.

- **enum DIRECTION {};**

Cela permet de définir la direction que le joueur prend, savoir si il va en haut (UP), en bas (DOWN), à gauche (LEFT), à droite (RIGHT) ou si il ne bouge pas (NONE).

- **std::vector<std::string> getMap();**

Cette fonction va juste retourner la map qu'il faut afficher. Ca peut être celle du nibbler, du pacman ou bien du menu.

- **unsigned int getSelect();**

Cette fonction est utilisée seulement pour le menu. Il retourne la ligne qu'on sélectionne (option, start game, exit, etc.). Dans le cas où l'on est dans un jeu, getSelect() retourne -1.

- **int getAction();**

Cette fonction sert à savoir si il a eu un évènement spécial comme la fin d'une partie, quand on change de librairie, ou quand on quitte le programme.

- **void setUsername(std::string username);**

Le setUsername() permet d'enregistrer le nom de l'utilisateur dans le jeu.