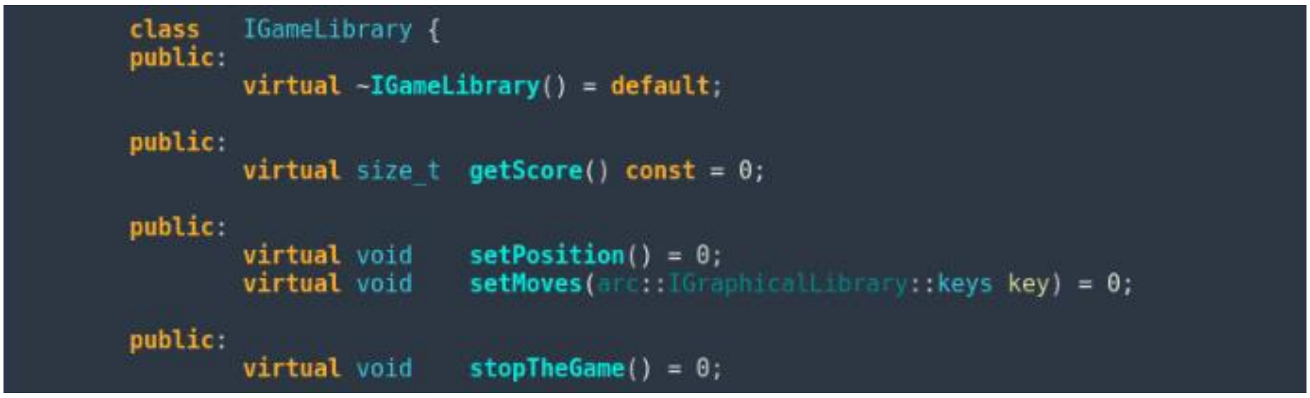
ARCADE DOCUMENTATION

An interface describes the behavior or capabilities of a C++ class without committing to a particular implementation of that class.

The C++ interfaces are implemented using **abstract classes** and these abstract classes should not be confused with data abstraction which is a concept of keeping implementation details separate from associated data.

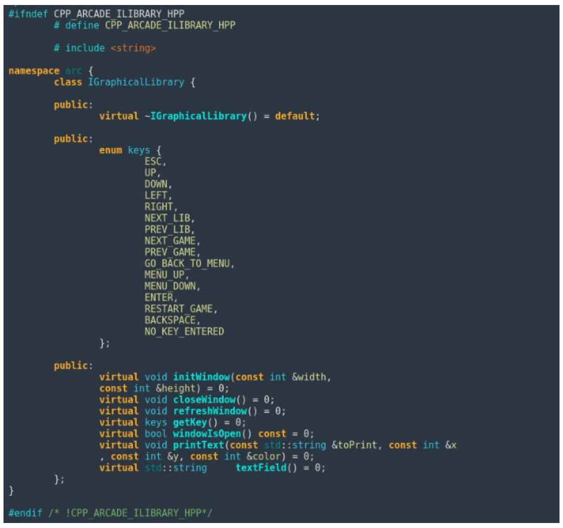
A class is made abstract by declaring at least one of its functions as **pure virtual** function. A pure virtual function is specified by placing "= 0" in its declaration as follows



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

public:
        virtual size_t  getScore() const = 0;

public:
        virtual void    setPosition() = 0;
        virtual void    setMoves(arc::IGraphicalLibrary::keys key) = 0;

public:
        virtual void    stopTheGame() = 0;
```

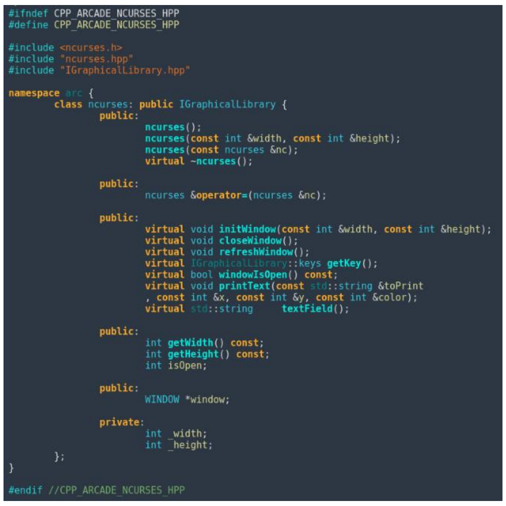
LIB:

Now, we have two library: Ncurses(page x for more information’s) & SFML (page x for more information’s) but you can add your favorite library because have our own Interface for library’s : IGraphicalLibrary



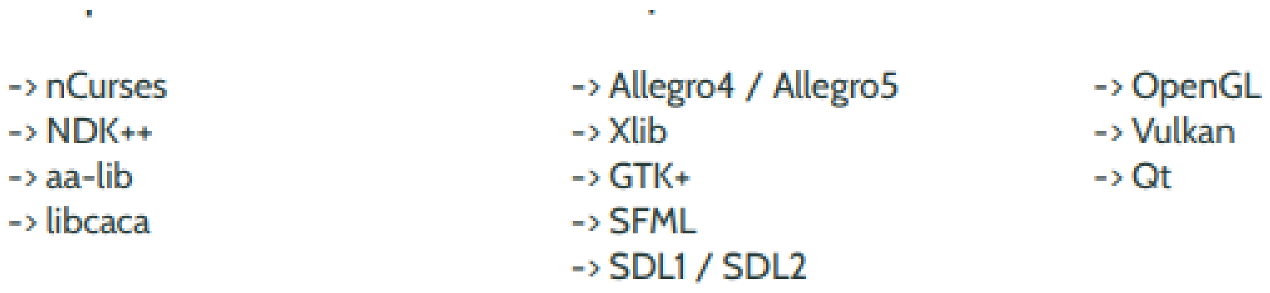
```
#ifndef CPP_ARCADE_ILIBRARY_HPP
        # define CPP_ARCADE_ILIBRARY_HPP

        # include <string>

namespace arc {
        class IGraphicalLibrary {

        public:
                virtual ~IGraphicalLibrary() = default;

        public:
                enum keys {
                        ESC,
                        UP,
                        DOWN,
                        LEFT,
                        RIGHT,
                        NEXT_LIB,
                        PREV_LIB,
                        NEXT_GAME,
                        PREV_GAME,
                        GO_BACK_TO_MENU,
                        MENU_UP,
                        MENU_DOWN,
                        ENTER,
                        RESTART_GAME,
                        BACKSPACE,
                        NO_KEY_ENTERED
                };

        public:
                virtual void initWindow(const int &width,
                const int &height) = 0;
                virtual void closeWindow() = 0;
                virtual void refreshWindow() = 0;
                virtual keys getKey() = 0;
                virtual bool windowIsOpen() const = 0;
                virtual void printText(const std::string &toPrint, const int &x
                , const int &y, const int &color) = 0;
                virtual std::string     textField() = 0;
        };
}

#endif /* !CPP_ARCADE_ILIBRARY_HPP*/
```

For exemple : to add Ncurses library :



```cpp
#ifndef CPP_ARCADE_NCURSES_HPP
#define CPP_ARCADE_NCURSES_HPP

#include <ncurses.h>
#include "ncurses.hpp"
#include "IGraphicalLibrary.hpp"

namespace arc {
	class ncurses: public IGraphicalLibrary {
		public:
			ncurses();
			ncurses(const int &width, const int &height);
			ncurses(const ncurses &nc);
			virtual ~ncurses();

		public:
			ncurses &operator=(ncurses &nc);

		public:
			virtual void initWindow(const int &width, const int &height);
			virtual void closeWindow();
			virtual void refreshWindow();
			virtual IGraphicalLibrary::keys getKey();
			virtual bool windowIsOpen() const;
			virtual void printText(const std::string &toPrint
			, const int &x, const int &y, const int &color);
			virtual std::string	textField();

		public:
			int getWidth() const;
			int getHeight() const;
			int isOpen;

		public:
			WINDOW *window;

		private:
			int _width;
			int _height;
	};
}

#endif //CPP_ARCADE_NCURSES_HPP
```

Library that you can add



-> nCurses
-> NDK++
-> aa-lib
-> libcaca

-> Allegro4 / Allegro5
-> Xlib
-> GTK+
-> SFML
-> SDL1 / SDL2

-> OpenGL
-> Vulkan
-> Qt

nCurses & SFML are already implemented and you can add more and more Library

Game that will be available

+ Nibbler

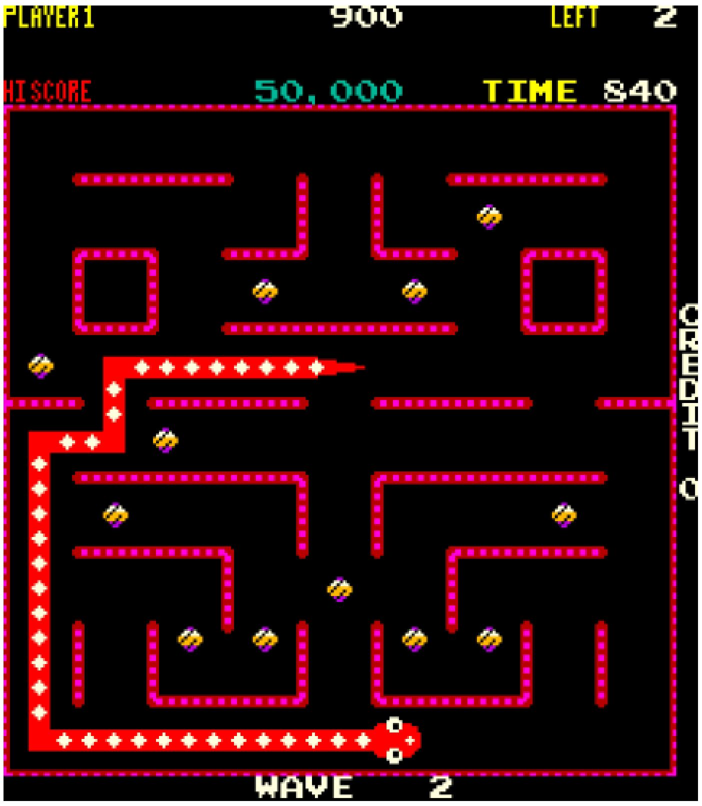
Nibbler is a simple arcade video game released in 1982.
Its concept has spread mainly thanks to the cult game **Snake**.
Nibbler itself was inspired by another great classic: **Blockade**, itself inspired from **Tron Light Cycle**.

The simplicity and addictiveness of **Snake** made it available on almost every existing platform under various names.
As you may know, **Snake** is about moving a snake around a map.
The snake is represented by sections and must eat food in order to grow.
The game is over when the head of the snake hits an edge of the map or one of the sections.
The goal of the game is to make the snake as long as possible.

Various versions of **Snake** exist.
Some of them include obstacles, others have a core system, or bonuses, etc.

Core rules

- The game area is a finite amount of cells. The edges of the area cannot be passed through.
- The snake starts with a size of 4 cells in the middle of the area.
- The snake moves forward automatically at a constant speed. Each section of its tail follows the exact same path as the head.
- The snake can turn right or left when the corresponding key is pressed.
- The goal of the game is to feed the snake so that it can grow. The game area MUST NEVER have less than one element of food.
- A food element fills a single cell.
- When the head of the snake goes over a cell with food, the food disappears and a one-cell-long section is added at the tail of the snake. The new section appears in the first free tile next to the last cell of the tail. If there is no free cell, the game is over. If a new section is added, a new food element appears.
- When the head of the snake runs into the border of the screen or a part of its body, the game is over



+ Pacman

Pacman is an arcade video game released in 1980.
The goal is to explore a maze in order to eat all the *"pacgums"* in it while avoiding ghosts.

Some *"pacgums"* let the player invert roles: **Pacman** can, for a short period of time, eat ghosts instead of being eaten.
Eaten ghosts do not disappear: their eyes head back to an unaccessible zone in the middle of the maze. They change back to normal ghosts after a short period.

Core rules

- The game area has a specific size. Going through one side of the area makes the player appear on the opposite side. All cells that are not walls may be walked through and contain *"pacgums"*.
- In the middle of the map is a small 5-cell-wide and 4-cell-high area that contains ghosts.
- Ghosts can get out of their box 10 seconds after the game starts.
- Pacman starts the game right under the ghosts.
- Some special, larger *"pacgums"* let Pacman eat ghosts. This effect lasts 10 seconds. During this period, ghosts become blue and flee Pacman instead of hunting him. Their movement speed is slower during this time. There are only 4 *"pacgums"* of this kind on the map.
- When Pacman eats a ghost, only its eyes remain. These eyes quickly go back to the ghost box, where the ghost is healed after a short period of time.
- The player wins when Pacman eats all the *"pacgums"*. A new map is loaded after that, or the current one is reloaded and movement is accelerated.
- On screen, Pacman and ghosts must not move cell by cell, but smoothly.

+ QIX

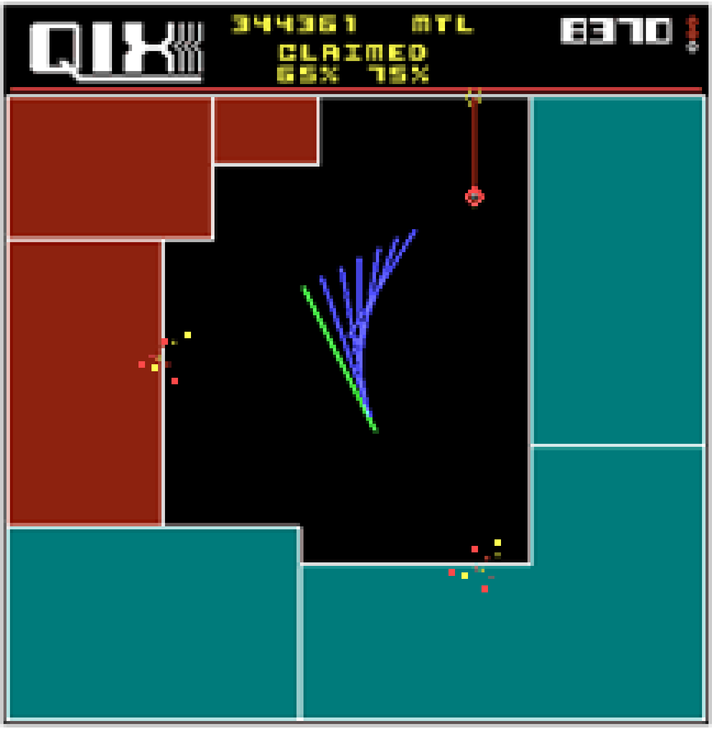
Qix is an arcade video game from 1981.
The game presents an area containing a monster: the **Qix**.
The player can move around this area and leaves a trail behind him.
When the player goes through a border of the screen, they appear on the other border and the area which was isolated from the **Qix** disappears.

The player wins when the remaining area containing the **Qix** represents less than 25% of the original area.
The player loses if the Qix crosses their trail.

The player's trail burns from its origin towards the player itself if the player stops. If the fire catches up to them, they lose.

Other monsters appear in the area "drawn" by the player. If one of these monsters touches the player, they lose.
Core rules

- The game area has a specific size. A cell can contain three different values that indicate its type: the cell can be walkable, non-walkable, or a border. A border is a specical walkable area.
- Non-walkable cells are space that was taken out by player action. Borders are cells that are directly in touch with a walkable cell and at least one non-walkable cell.
- The walkable area contains a monster: the Qix, which is several cells long and moves randomly.
- If the Qix touches the player or their trail, the player loses and goes back to the border they came from
- When the player walks on a walkable cell, they leave a trail behind them. This trail ignites if they stop. The player cannot cross the trail.
- When the player touches a border, the walkable area splits: the part containing the Qix remains and the other becomes non-walkable.
- Borders contain special monsters called Sparks. The player loses the game when they touch a Spark. Sparks can also move along the trail or old borders, even if they are in a non-walkable area, if it helps them hunt the player. A Spark cannot turn back.
- The player wins when the Qix is sealed inside less than 25% of the starting area.



+ Centipede

Centipede is an arcade video game released in 1981.
The game features an area with a lot of empty space and some boxes.
The player is at the bottom of the scrceen and can move in all directions with some limitations: they can only move up and down by a little.
However, they can move freely left and right.
The player can also shoot projectiles towards the top of the screen.

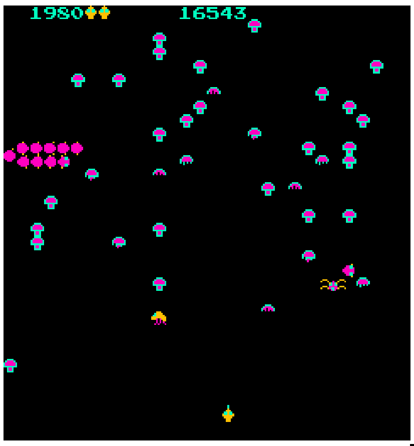
Regularly, **Centipedes** appear at the top of the screen.
They move from left to right or right to left and go down when they encounter a box or the border of the screen.
If one of them touches the player, they lose.
A **Centipede** has a head, a tail and a body between the two.

When a shot from the player hits a **Centipede**, the part that was hit turns into a box and the body is split in half, becoming two smaller **Centipedes**. The **Centipede** spawned from the tail part encounters the newly formed box and heads back the way it came from, after having moved down.

Boxes can be destroyed by several shots from the player.

There can only be one shot one the screen at any given time.
Core rules

- The game area is split in two: a walkable area and a non-walkable area. The walkable area is at the bottom of the screen. It fills the entire width of the screen but only 20% of its height.
- Centipedes come from the top of the screen. They move from side to side and encountering an obstacle or screen border makes them move down a line. A centipede is a snake composed of several parts.
- The player can shoot. When a shot hits a centipede, it is split in half. The part that was hit turns into an obstacle, the head part keeps moving on and the tail part collides with the obstacle, moving down a line and turning back.
- If a centipede touches the player, they lose the game.
- If a centipede touches the bottom of the screen, the player loses score.
- The player wins if they survive 20 centipedes. The map is then reset and the game starts over.
- A Centipede map contains randomly generated obstacles.
- An obstacle can be destroyed by shooting it 5 times.
- There can only be one shot on the screen at any given time.



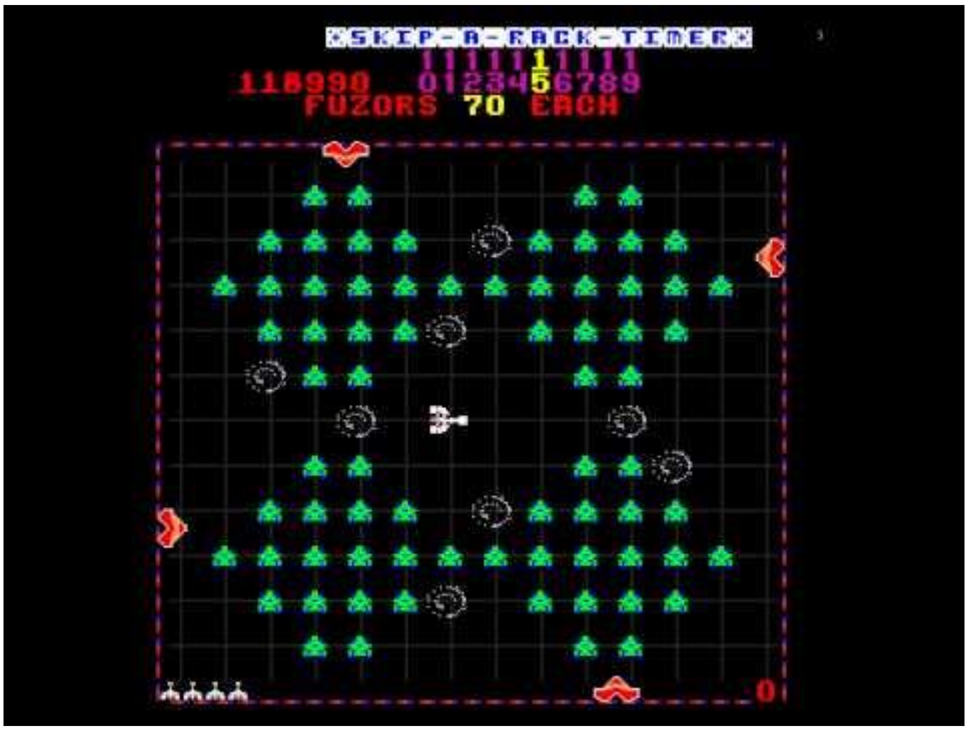
+ Solar Fox

Solar Fox is an arcade video game from 1981.
The game takes place in space and the player is in command of a space ship.
The play area is a grid, containing some powerups that can be picked up by shooting them.

Enemy guns appear on the border of the game area and shoot in a straight line. The player continuously moves forward and must dodge enemy shots.
They can also shoot to pick up powerups and intercept enemy shots.

Solar Fox is a dodge and collect game.
Many clones added functionalities and speed, making it a classic arcade game.
Core rules

- The game area is split in two: a central walkable area, with a margin of 2 or 3 cells between its limit and the screen border. The rest of the game area is non walkable and contains opponents.
- The player can move around the walkable area in every direction, but cannot turn back directly.
- The player cannot stop and always moves forward. A key lets them move faster.
- The walkable area is filled with powerups that the player must shoot to win.
- Opponents appear on the borders of the walkable area and shoot.
- Shots from the opponents can be destroyed by the player's shots.
- The player's shots only have a range of two cells. Their speed is three or four times faster than that of the player.
- The player loses the game if their spaceship is hit by a shot, special bad powerups or the walkable area's borders.
- The spaceship, lasers and opponents must not move cell per cell, but smoothly.



<u>Sources : Epitech ARCADE subject & Googles Images/Wikipédia.</u>