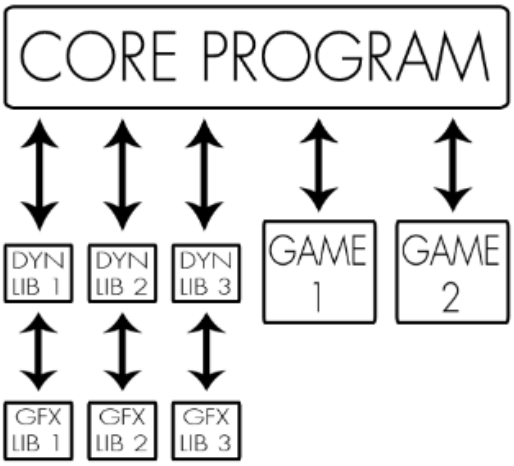
Arcade Documentation

Presentation:

Arcade is a gaming platform: a program that lets the user choose a game to play and keeps a register of player scores.

To be able to deal with the elements of your gaming platform at run-time, your graphics libraries and your games must be implemented as dynamic libraries, loaded at run-time.

Each GUI available for the program must be used as a shared library that will be loaded and used dynamically by the main program.



Create a new library:

1 – Load Instance:

Each library must include:

- A load<_>Instance function (<_> = Game or Graphic)
- The function should return an IGame interface for game libraries or an IGraphic interface for graphical libraries.
- The load<_>Instance function should take no parameters.

```
// Prototype for a game library
extern "C" IGame *loadGameInstance();

// Prototype for a graphical library
extern "C" IGraphic *loadGraphicInstance();
```

2 – Commons Libs Interfaces

/!\ all the interfaces must be with the namespace ‘Arcade’ and all methods in pure virtual.

IColor: The interface to handle colors

- **Void setColor(short r, short g, short b, short a):** set the rgba color.

- **Short getR():** return the Red color value.

- **Short getG():** return the Green color value.

- **Short getB():** return the Blue color value.

- **Short getA():** return the Alpha color value.

IEntity: The interface to handle Entities to display (pictures, characters)

- **Void setPos(std::size_t x, std::size_t y):** set the position of the entity. The position is given in cells and not in pixels.

- **Void setSize(std::size_t x, std::size_t y):** set the size of the entity. The size is given in pixels and not in cells.

- **Void setChar(char c):** set the charcter to the entity. (Use for the Ncurses lib for example).

- **Void setColor(std::unique_ptr<IColor> color):** set the IColor of the entity. The color not affect the sprite color it’s just for Ncurses.

- **Void setPath(const std::string &path):** set the path of the assets of the entity. The path is relative to the repository root and must not contain the extension.

- **Void setRotation(float rotation):** set the rotation of the entity.

- **Std::vector<std::size_t> getPos():** return the position of the entity in a vector.

- **Std::vector<std::size_t> getSize():** return the size of the entity in a vector.

- **Int getChar():** return the character of the entity.

- **Std::shared_ptr<IColor> getColor():** return the IColor of the entity.

- **Std::string getPath():** return the path of the entity asset.

- **Float getRotation():** return the rotation of the entity.

IText: The interface to handle text to display

- **Void setFontPath(const std::string &font):** set the path of the text font. The path is relative to the repository root and must not contain the extension.

- **Void setText(const std::string &text):** set the text to be displayed.

- **Void setColor(std::unique_ptr<IColor> color):** set the IColor of the text. The color not affect the sprite color it's just for Ncurses.

- **Void setPos(std::size_t x, std::size_t y):** set the position of the text. The position is given in cells and not in pixels.

- **Void setSize(std::size_t x):** set the size of the text. It corresponds to the font size.

- **Void setRotation(float rotation):** set the rotation of the text.

- **Std::string getFontPath():** return the path of the text font.

- **Std::string getText():** return the text.

- **Std::shared_ptr<IColor> getColor():** return the IColor of the text.

- **Std::vector<std::size_t> getPos():** return the position of the text in a vector.

- **td::size_t getSize():** return the size of the text.

- **Float getRotation():** return the rotation of the text.

ISound: The interface to handle sounds

The following enumeration enumerates the different states of the music.

```
enum Status {
START,
LOOP,
STOP,
DONE
};
```

- **Void setPathSound(const std::string &path)**: set the path of the sound. The path is relative to the repository root and must not contain the extension.

- ***Void setVolume(float volume):*** set the volume of the sound.

- **Void setStatus(int status):** set the enumeration status of the sound.

- **Std::string getPathSound():** return the path of the sound.

- ***Float getVolume()***: return the volume of the sound.

- **Int getStatus():** return the enumeration status of the sound.

Keys: The enumeration of the user inputs

```
enum Keys {
UNKNOWN = -1,
A = 0,
B = 1,
C = 2,
D = 3,
E = 4,
F = 5,
G = 6,
H = 7,
I = 8,
J = 9,
K = 10,
L = 11,
M = 12,
N = 13,
O = 14,
P = 15,
Q = 16,
R = 17,
S = 18,
T = 19,
U = 20,
V = 21,
W = 22,
X = 23,
Y = 24,
Z = 25,
ESCAPE = 26,
TAB = 27,
SHIFT = 28,
CONTROL = 29,
SPACE = 30,
```

```
ENTER = 31,
BACKSPACE = 32,
UP = 33,
DOWN = 34,
LEFT = 35,
RIGHT = 36,
ZERO = 37,
ONE = 38,
TWO = 39,
THREE = 40,
FOUR = 41,
FIVE = 42,
SIX = 43,
SEVEN = 44,
EIGHT = 45,
NINE = 46
};
```

3 – Graphic Library:

IGraphic: Graphic Interface for graphic libraries

- **Bool IsWindowOpen()**: return true if the window is open, false nor.

- **Void closeWindow()**: close the window.

- **Void clearWindow()**: clear the window, it erases all the elements display on the window.

- **Int getKeyEvent()**: return the int relative to the Keys enumeration.

- **Void displayWindow(std::vector<std::shared_ptr<IEntity>> entities)**: display on the window all the entities given in the vector.

- **Void displayText(std::vector<std::shared_ptr<IText>> texts)**: display on the window all the texts given in the vector.

- **Void playSound(std::vector<std::shared_ptr<ISound>> sounds)**: play the songs given in the vector.

4 – Game Library:

IGame: Game Interface for games libraries

- **Int startGame():** start the game

- **Int StopGame():** stop the game

- **Int getScore():** return the score of the actual game

- **Int simulate():** simulate a tick of the game. Return 0 all the time or –1 if the game have to be ended.

- **Void catchKeyEvent(int key):** get the given key pressed by the user relative to the Keys enumeration.

- **Void setUserName(const std::string &name):** set the username to the game.

- **Std::string getUserName():** return the username.

- **Std::vector<std::shared_ptr<IEntity>> getEntities():** return the vector of IEntity to display.

- **Std::vector<std::shared_ptr<IText>> getTexts():** return the vector of IText to display.

- **Std::vector<std::shared_ptr<ISound>> getSounds():** return the vector of ISound to play.