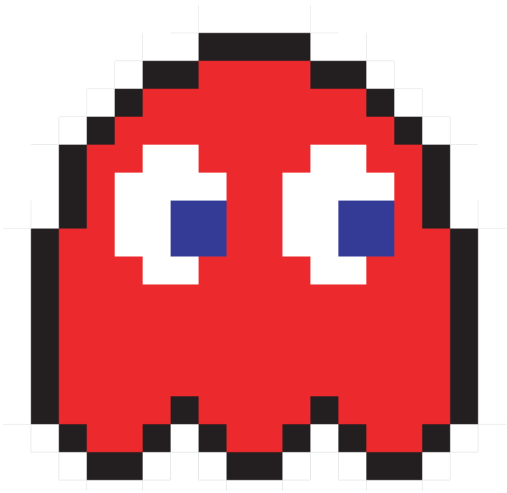

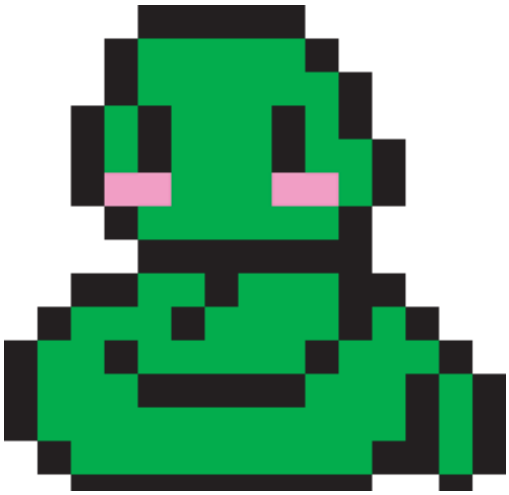

Arcade

A Retro Platform

Technical documentation

Project made by: Louis LECCE, Jean-Gaël CHOPPE and Alexis WALTER

GAME

To create a new game you should:

firstly create a class inherited from Igame class,
next implement all function of Igame:

```
class IGame {
public:
    virtual ~IGame() {};
    virtual void handleEvent(std::string &name) = 0;
    virtual void handleUpdate(int elapsedTime) = 0;
    virtual void handleRender(IGraphicRenderer &renderer) = 0;
    virtual std::map<std::string, std::string> getGameData() = 0;
    virtual  void setGameData(std::map<std::string, std::string> &data) = 0;
};
```

handleEvent: in this function you will receive as a string all event get by the graphical library following this array:

Event	Description	Game	Key (exemples, can be changed)
quit	quit the arcade		suppr or window close
next_game	next game		e
prev_game	previous game		c
next_graphic	next graphic lib		a
prev_graphic	previous graphic lib		w
menu	go back to the menu		esc
restart	restart the game		r
enter	enter event	X	enter
space	space event	X	space
left	left event	X	arrow or q
right	right event	X	arrow or d
up	up event	X	arrow or z
down	down event	X	arrow or s

the first collumn represent what you gonna find in the string handled by the function

handleUpdate: this function let you update your game according to the elapsed time (time since the last call of this function), update in this case means move your player or increase a score based on the time.

handleRenderer: in this function you will call the graphical function that you have define in your graphical library like «drawSprite»

get and set game data are the communication function. by calling the setGameData method, the Core is going to send you all the data needed by your game like a save or the name of the player get in the menu.

you will need to have an entry point in your file: IGame *entry(); in extern "C"
after that pass to the compilation step.

GRAPHICAL LIBRARY

To create a new graphic library you should:

firstly create a class inherited from IGraphic class,
next implement all function of IGraphic and IGraphicRenderer:

```
class IGraphicRenderer {
public:
    virtual ~IGraphicRenderer() {};

    virtual void drawScreen() = 0;

    virtual void clearScreen() = 0;

    virtual void drawRect(Rect &rect) = 0;

    virtual void drawCircle(Circle &circle) = 0;

    virtual void drawSprite(Sprite &sprite) = 0;

    virtual void drawText(Text &text) = 0;

    IGraphicRenderer &operator=(const IGraphicRenderer &) = delete;
};

class IGraphic: public IGraphicRenderer {
public:
    virtual ~IGraphic() {};

    virtual bool isOperational() = 0;

    virtual std::string handleEvent() = 0;
};
```

all the size and position send to drawSprite, drawText, drawCircle and drawRect are send in percent.
That means if you have a window 800x800 and you have to draw a rect of size 20 you have to draw a rect of size:
width: 20 x width / 100 = 160
height: 20 x height / 100 = 160 *in this case we have the same result because our window is a square

drawScreen: this function update the screen;
clearScreen: this one clear the screen
drawRect: this function allow you to draw a rectangle shape
drawCircle: this function allow you to draw a circle shape
drawSprite: this function allow you to draw a sprite
drawText: this function allow you to display text
isOperational: that return a boolean true if the window is open else it return false
handleEvent: this function return a string based on the array in the game documentation.
you will also need an entry point in your file: IGraphic *entry(); in extern "C"

COMPILATION

to compile your library/game you need to compile it with "-fpic -shared" to create a dynamic library which can be load by the arcade

And name with the g++ flag «-o» it like that:

for a game:

lib_arcade_"name of the game".so (without the quote)
and copy the output .so file into the ./games folder

for a graphical library:

lib_arcade_"name of the library".so (without the quote)
and copy the output .so file into the ./lib folder