Arcade

Other group

charles.raimbault@epitech.eu

Explanatory manual

An explanatory manual has been generated by Doxygen in a HTML format in /doc

Draw.io

Our architecture have been fully represented in a Draw.io :
https://www.draw.io/?state=%7B%22ids%22:%5B%221Zq1VpeW_EqprgKpBIExYpuWdQAxUMRU4%22%5D,

How to implement a new graphic library

Configuration files

You must implement 2 configurations files, the first one must contain the path of the sprites you'll be using, their size, how the sprite can be represented in Ncurses and the path to the lib. It have have to be separated by a line.

Configuration example

./textures/lib_ncurses.bmp
200
200
ncurses
9
./lib/lib_arcade_ncurses.so

The second one must represent the 'map' that your lib will print, with code representing which part of the configuration file it have to use.

Map example

1111111111111111111
1222222221222222221
1511211121211121151
1222222222222222221
1211212111112121121
1222212221222122221
1111211101011121111
0001210000000121000
1111210110110121111
0000200104010020000
1111210111110121111
0001210000000121000
1111210111110121111
1222212221222122221
1211211121211121121
1521222223222221251
1121212111112121211
1222212221222122221
1211111121211111121
1222222222222222221
1111111111111111111

Functions to implement

You must implement a runGraph fonction that will return return true if you want to exit the program, otherwise return false.

```
bool YourClassName::runGraph();
```

You must implement a translateKey fonction translating the key returned by the events of your lib to a key that will be understood by the core

```
void YourClassName::translateKey()
```

Here is the translation

{‘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},
{‘0’, 26},
{‘1’, 27},
{‘2’, 28},
{‘3’, 29},
{‘4’, 30},
{‘5’, 31},
{‘6’, 32},
{‘7’, 33},
{‘8’, 34},
{‘9’, 35},
{‘Up’, 36},
{‘Down’, 37},
{‘Right’, 38},
{‘Left’, 39},
{‘Enter’, 40},
{‘Space’, 41},
{‘Backspace’, 42},
{‘Tab’, 43},

You must implement all these functions that you can copy / paste from below

```
void YourClassName::setIsNewMap(bool newMap)
```

```
{
    _isNewMap = newMap;
}

bool YourClassName::getIsNewMap(void) const
{
    return (_isNewMap);
}

void YourClassName::setIsNewKey(bool newKey)
{
    _isNewKey = newKey;
}

bool YourClassName::getIsNewKey(void) const
{
    return (_isNewKey);
}

void YourClassName::setLastKey(int key)
{
    _key = key;
}

int YourClassName::getLastKey(void) const
{
    return (_key);
}

void YourClassName::setScore(size_t score)
{
    _score = score;
}

size_t YourClassName::getScore() const
{
    return (_score);
}

void YourClassName::setPathConfig(std::string path) noexcept
{
    _pathConfig = path;
}

std::string YourClassName::getPathConfig() const noexcept
{
```

```
    return (_pathConfig);
}

void YourClassName::setIsNewPathConfig(bool isNewPath) noexcept
{
    _isNewPathConfig = isNewPath;
}

bool YourClassName::getIsNewPathConfig() const noexcept
{
    return (_isNewPathConfig);
}
```

You must implement these two fonctions respectively

```
void buildMap(std::shared_ptr<std::vector<std::string>> = nullptr);
void setMap(std::shared_ptr<std::vector<std::string>>);
```

These fonctions will generate your map in your lib.

> Keep in mind that some lib does'nt need buildMap, but this fonction will be called BEFORE setMap

You need to implement in entryPoint

```
extern "C"
{
    IGraphic *entryPoint(void)
    {
        YourClassName *instance = new YourClassName();
        return (instance);
    }
}
```

SFML Example

```
#include "ClassSFML.hpp"

ClassSFML::ClassSFML():
    _wind(nullptr),
```

```
    _key(0),
    _isNewPathConfig(false),
    _isNewMap(false),
    _isNewKey(false)
{
    _wind = std::make_unique<sf::RenderWindow>();
    _wind->create(sf::VideoMode(SCREEN_WIDTH, SCREEN_HEIGHT), "Arcade SFML");
    _wind->setPosition(SCREEN_POS);
}

ClassSFML::~ClassSFML()
{
    _wind->close();
}

void ClassSFML::displayGame()
{
    for (auto it = _map->begin(); it != _map->end(); ++it)
        for (auto it_sprite = it->begin(); it_sprite != it->end(); ++it_sprite)
            if (it_sprite->first != NOTHING)
                _wind->draw(it_sprite->second);
}

bool ClassSFML::getEvent()
{
    while (_wind->pollEvent(_event)) {
        if (_event.type == sf::Event::Closed) {
            _wind->close();
            return (true);
        }
        if (_event.type == sf::Event::KeyPressed) {
            translateKey();
            setIsNewKey(true);
        }
    }
    return (false);
}

bool ClassSFML::runGraph()
{
    if (getIsNewPathConfig() == true) {
        _parsing.clearData();
        setIsNewPathConfig(false);
        _parsing.setFilename(getPathConfig());
        _parsing.readFile();
        setMapTexture();
```

```
    }
    if (!_wind->isOpen())
        return (true);
    if (getEvent())
        return (true);
    if (getIsNewMap()) {
        setMapTexture();
        _wind->clear();
        setIsNewMap(false);
    }
    _wind->clear();
    displayGame();
    _wind->display();
    return (false);
}

void ClassSFML::setMapTexture()
{
    std::vector<DataParsingConfig> parsingResult = _parsing.getResult();
    float x = 0;
    float y = 0;

    _textures.clear();
    for (auto it = parsingResult.begin(); it != parsingResult.end(); ++it) {
        sf::Vector2i size = {it->sizeX, it->sizeY};
        std::shared_ptr<sf::Texture> tmp (new sf::Texture);
        tmp->loadFromFile(it->path, sf::IntRect(0, 0, size.x, size.y));
        _textures.push_back(std::make_pair(size, tmp));
    }
    for (auto it_y = _map->begin(); it_y != _map->end(); ++it_y) {
        x = 0;
        for (auto it_x = it_y->begin(); it_x != it_y->end(); ++it_x) {
            if (it_x->first != NOTHING) {
                it_x->second.setTexture(*_textures.at(it_x->first - 48).second.get());
                it_x->second.setPosition({x, y});
            }
            x += _textures.at(it_x->first - 48).second->getSize().x;
        }
        y += _textures.begin()->second->getSize().y;
    }
}

void ClassSFML::buildMap(std::shared_ptr<std::vector<std::string>> map = nullptr)
{
    _map = std::make_unique<std::vector<std::vector<std::pair<char, sf::Sprite>>>>();
```

```
    for (auto it = map->begin(); it != map->end(); ++it) {
        std::vector<std::pair<char, sf::Sprite>> tmp;
        for (auto it_str = it->begin(); it_str != it->end(); ++it_str) {
            sf::Sprite sprite;
            tmp.push_back(std::make_pair(*it_str, sprite));
        }
        _map->push_back(tmp);
    }
}

void ClassSFML::setMap(std::shared_ptr<std::vector<std::string>> map)
{
    if (!map || !_map)
        return;
    auto it_my_map_y = _map->begin();

    for (auto it_y = map->begin(); it_y != map->end(); ++it_y, ++it_my_map_y) {
        auto it_my_map_x = it_my_map_y->begin();
        for (auto it_x = it_y->begin(); it_x != it_y->end(); ++it_x, ++it_my_map_x)
            it_my_map_x->first = *it_x;
    }
}

void ClassSFML::translateKey()
{
    for (size_t i = 0; KeySFML[i].code_lib != -1; ++i) {
        if (_event.key.code == KeySFML[i].code_lib) {
            setLastKey(KeySFML[i].code_core);
            setIsNewKey(true);
            break;
        }
    }
}

void ClassSFML::setIsNewMap(bool newMap)
{
    _isNewMap = newMap;
}

bool ClassSFML::getIsNewMap(void) const
{
    return (_isNewMap);
}

void ClassSFML::setIsNewKey(bool newKey)
{
```

```
    _isNewKey = newKey;
}

bool ClassSFML::getIsNewKey(void) const
{
    return (_isNewKey);
}

void ClassSFML::setLastKey(int key)
{
    _key = key;
}

int ClassSFML::getLastKey(void) const
{
    return (_key);
}

void ClassSFML::setScore(size_t score)
{
    _score = score;
}

size_t ClassSFML::getScore() const
{
    return (_score);
}

void ClassSFML::setPathConfig(std::string path) noexcept
{
    _pathConfig = path;
}

std::string ClassSFML::getPathConfig() const noexcept
{
    return (_pathConfig);
}

void ClassSFML::setIsNewPathConfig(bool isNewPath) noexcept
{
    _isNewPathConfig = isNewPath;
}

bool ClassSFML::getIsNewPathConfig() const noexcept
{
    return (_isNewPathConfig);
```

```
}

extern "C"
{
    IGraphic *entryPoint(void)
    {
        ClassSFML *instance = new ClassSFML();
        return (instance);
    }
}
```

How to implement a new game library

Configuration files

They must be implemented the same way as graphic libraries

Class implementation

In your class must be implemented two private attribute

```
std::string _pathConfig = "./path/to/your/file.config";
std::string _pathMap = "./path/to/your/map.config";
```

Functions to implement

You must implement a runGame fonction that will handle the logic of your game, plus the clock

```
bool Pacman::runGame();
```

You must implement a readMap fonction that will read the map

```
void Pacman::readMap();
```

You must implement all these functions that you can copy / paste from below

```
void YourClassName::setMap(std::shared_ptr<std::vector<std::string>> map)
{
```

```
    _map = map;
}

std::shared_ptr<std::vector<std::string>> YourClassName::getMap(void) const
{
    return (_map);
}

void YourClassName::setIsNewMap(bool map)
{
    _isNewMap = map;
}

bool YourClassName::getIsNewMap(void) const
{
    return (_isNewMap);
}

void YourClassName::setIsNewKey(bool isNewKey)
{
    _isNewKey = isNewKey;
}

bool YourClassName::getIsNewKey(void) const
{
    return (_isNewKey);
}

void YourClassName::setLastKey(int key)
{
    _key = key;
}

int YourClassName::getLastKey(void) const
{
    return (_key);
}

void YourClassName::setScore(size_t score)
{
    _score = score;
}

size_t YourClassName::getScore() const
{
    return (_score);
```

```
}

const std::string YourClassName::getPathConfig() const noexcept
{
    return (_pathConfig);
}

const std::string YourClassName::getPathMap() const noexcept
{
    return (_pathMap);
}

void YourClassName::setMove(Move lastMove)
{
    _lastMove = lastMove;
}

Move YourClassName::getMove() const
{
    return (_lastMove);
}

extern "C"
{
    IGame *entryPoint(void)
    {
        YourClassName *instance = new YourClassName();
        return (instance);
    }
}
```

Have fun !