Arcade Documentation_

Project by:

Alex Limongi (group leader)

Brandon Segers

Baptiste Terras

Games:

Nibbler

Solar Fox

Graphical libraries:

nCurses

Libcaca

SFML

Implementation of graphic libraries:

To implement a new graphic library, you must create an object inheriting from this interface:

```
class IGraphLib {
    public:
        virtual ~IGraphLib() = default;
        virtual void setGameList(vector<string>) = 0;
        virtual void setLibList(vector<string>) = 0;
        virtual void init_menu() = 0;
        virtual void init_game() = 0;
        virtual playerEvent displayMenu() = 0;
        virtual void displayMap(map_info_t map) = 0;
        virtual playerEvent getKey() = 0;
};
```

I. Structures and enumerations:

`playerEvent` is defined by this enumeration:

```
enum playerEvent {
    PE_EXIT = -1,
    PE_NOACTION,
    PE_UP,
    PE_DOWN,
    PE_LEFT,
    PE_RIGHT,
    PE_ACTION1,
    PE_ACTION2,
    PE_ACTION3,
    PE_NEXT_GAME,
    PE_PREV_GAME,
    PE_NEXT_LIB,
    PE_PREV_LIB,
    PE_RESTART
};
```

`map_info_t` is defined by these structures:

```
typedef enum color_e
{
    RED = 1,
    GREEN,
    BLUE,
    MAGENTA,
    YELLOW,
    CYAN,
    /*...*/
    BLACK,
    WHITE,
} color_t;

typedef struct position_s
{
    int x;
    int y;
} position_t;

typedef struct pixel_s
{
    color_t color;
    position_t pos;
} pixel_t;

typedef struct map_info_s
{
    int score;
    vector<pixel_t> pixel;
    vector<string> map;
} map_info_t;
```

The `map_info_t` structure is the structure generated by the game.

It contains:

- the current `score`.
- a vector of pixel, each pixel defines the `color_t` of the character at the `position_t` x, y of the `map`.
- the `map` which represent the game.

Specific characters represent specific things:

- `^`, `v`, `<`, `>` : the player and its direction.
- `X` or `T` : the game borders
- `P` or `B` : points to gather.
- `E` : enemies.
- `o` or `6` : projectiles

II. Methods:

- `setGameList` :
  This method is used to set game list after the creation of the graphical library object.

- `setGameList` :
  This method is used to set graphical library list after the creation of the graphical library object.

  `init_menu` :
  This method is called before the menu loop start

- `init_game` :
  This method is called before the game loop start

- `displayMenu` :
  This method is called inside of the menu loop.
  Return the action to perform to the core.
  If `playerEvent::PE_RESTART` is returned the game start.

- `displayMap` :
  This method is called inside of the game loop.

- `getKey` :
  This method is called inside of the game loop
  Return the action to perform to the core

Implementation of game libraries:

To implement a new game library, you must create an object inheriting from this interface:

```
class game_lib {
    public:
        virtual ~game_lib() = default;
        virtual map_info_t game(playerEvent action) = 0;
        void *handle;
};
```

And an entryPoint function returning a new instance of your game object:

```
extern "C" game_lib *entryPoint()
{
    return new GameObject();
}
```

I. Structures and enumerations:

Define above.

II. Method:

The `game` method generates and returns a `map_info_t` structure as described above.

This method is called inside of the game loop.

III. Variable

The `handle` variable is only here to store the library handle.

You must not modify it!