arcade :
a retro platform

<u>documentation</u>

Table of contents

What’s arcade ?

Arcade is a **console interface** for retro video games and a **game engine** that can run a game on few graphic libraries.

Installation and prerequisite :

- First, clone the repository.

- Then, make sure to have all these libraries installed:

  - SDL2
  - SFML
  - NCURSES

The Core :

- Every file in “/core” **mustn’t** be modified.

- Core is the interface that link graphical libraries in “/**graphicals**” and games in “/**games**”.

Add a new library :

- The library you want to implement must be written in **C++**.
- It must be compiled into a **dynamic library** (with a **.so** extension).
- You should name your library accordingly to this :

lib_arcade**_$LIBNAME**.so, where $**LIBNAME** the name of your library.

- Place it in the “/**lib**” directory.

Functions that you must implement :

- Your library must contain a class inheriting the **IGraphical** interface, located in “/**graphicals**” folder.
- You must implement all the following functions, without that your library will not work with our Arcade.

- void **display**() :
  This function must display the current scene. It is used in the game loop.

- EventType **getEventType**() :
  This function must return the event type. It is called once per frame.

- EventKey **getKeyPressed**() :
  This function must returns the key pressed, if any. It is also called once per frame.

- virtual void **setListGames**(const std::vector<std::string> &games, const std::function<void (const std::string &)> &fct, int chosen = -1) :
  The core provides a list of games to the user through this function. When a game is chosen from the list (first argument), it should be passed to the function (second argument). The third argument represents the currently chosen game, if any.

- virtual void **setListLibraries**(const std::vector<std::string> &libraries, const std::function<void (const std::string &)> &fct, int chosen = -1) :
  The core provides a list of libraries to the user through this function. When a library is chosen from the list (first argument), it should be passed to the function (second argument). The third argument represents the currently chosen library, if any. If no library is chosen, it is set to -1.

- virtual void **setScores**(const std::vector<std::pair<std::string, std::string>> &scores) :
  This function sets the list of scores. First element of the pair is the username, the second is the score.

- virtual void **setControls**(const std::map<std::pair<EventType, EventKey>, std::function<void ()>> &controls) :
  This function sets the controls for the game. They should be checked for only when the current scene is GAME.

- virtual void **setFunctionPlay**(const std::function<void()> &function) :
  This is the function that lets the game start. Should only be called when a game is chosen.

- virtual void **setFunctionRestart**(const std::function<void()> &function) :
  This function should be used in the GAME and END_GAME scenes. It restarts the game.

- virtual void **setFunctionMenu**(const std::function<void()> &function) : This function should be used in the GAME and END_GAME scenes. It restarts the game.

- virtual void **setFunctionTogglePause**(const std::function<void()> &function) :
  This function should be used to pause and unpause the game when it is running (GAME scene).

- virtual const std::string &**getUserName**() :
  Should return the username entered by the player in the main menu scene.

- virtual void **setUserName**(const std::string &username) :
  This function is called to set the username of the player in case he entered one earlier.

- virtual Scene **getScene**() const :
  Returns the current scene.

- virtual void **setScene**(Scene scene) :
  Sets the current scene.

- virtual void **setHowToPlay**(const std::vector<std::pair<std::string, std::string>> &info) :
  This is information for the player about the game controls. First element of the pair is the description of the action.

- virtual void **setGameStats**(const std::vector<std::pair<std::string, std::string>> &info) :
  This is called in a loop when the game is running. Sets the current game information. The first element of the pair

- virtual void **updateGameInfo**(const std::vector<std::shared_ptr<Thing>> &gameMap) :
  This is called in a loop when the game is running. This vector contains entities that should be displayed in the game scene.

- virtual void **setMapSize**(size_t height, size_t width) :
  Sets the size of the game map.

- virtual void **setGameTitle**(const std::string &game) :
  Sets the title of the game.

- virtual void **setGamePause**(bool pause) :
  Informs the game library whether the game is paused or not. It is called when the library is loaded and when there is a change.

- virtual void **setScore**(std::string score) :
  Sets the current score.

Add a new game :

- The game you want to implement must be written in C++.
- It must be compiled into a dynamic library (with a .so extension).
- You should name your game accordingly to this :

lib_arcade_$GAMENAME.so, where $GAMENAME the name of your game.

- Place it in the "/lib" directory at the root of your project.

Functions that you must implement :

- Your game must contain a class inheriting the IGames interface located in the include folder located at the root of the project.
- You must implement all functions, without that your library will not work with our Arcade.

  - virtual size_t getMapHeight() const :
    Getter for the height of the map.
  - virtual size_t getMapwidth () const :
    Getter for the width of the map.
  - virtual const std::string &getMusic() const :

    Getter for the music.
  - virtual const std::string &getSound() const :
    Getter for the sound.
  - virtual const std::string &getScore() const :
    Getter for the score.
  - virtual const std::map<std::pair<EventType, EventKey>, std::function<void ()>> &getControls() const :
    Getter for the controls of the game.

- virtual const std::vector<std::shared_ptr<Thing>> &**getEntities**() const :
  Getter for the entities.

- virtual const std::vector<std::pair<std::string, std::string>> &**getGameControls**() const :
  Getter for the game controls indication.

- virtual const std::vector<std::pair<std::string, std::string>> &**getGameStats**() const :
  Getter for the game stats (score, other stats).

- virtual void **restart**() :

  Call this function to restart the entire game. The game must reset himself.

- virtual void **updateGame**(int dir) :
  updateGame function should be called in a loop. It's used to advance the game and update all logic.

- virtual bool **isGameOver**() const :
  Check if the game is over.

- virtual const std::string &**getTitle**() const :
  Get the name of the game.