Arcade

Table of Contents

Introduction

'Arcade' is an extensive C++ Epitech project done in second year (tek 2) and consists of using multiple graphical libraries to run multiple games and being capable to switch between the different graphical libraries during runtime.

This document explains in technical detail how to extend the game from our implemetation.

Directory structure

```
.
├── assets
├── includes
├── lib
└── src
    ├── Games
    └── Libraries
```

Adding a new graphics library

Here are the requirements for adding new graphics libraries:

- Your library must inherit the given `IGraphics.hpp` interface
- Your library source files must be located in `src/Libraries`
- The generated `.so` file must be placed in `lib/`
- Library source files must be named as `Name.hpp` & `Name.hpp`

The library must have a `createLibrary()` function that returns an `IGraphic` interface within it's respective header file.

```
extern "C" IGraphic *createLibrary();
```

The `createLibrary()` must incorporate the `dlopen()`, `dlsym()`, and `dlclose()` functions to open the library, bind functions, and close safely.

The library will be added to an `std::vector` on initialization of the program.

A basic graphics library class would look like:

```cpp
class Name : public IGraphic
{
public:
    Name();
    ~Name();

    void startWindow();
    EventEnum getEvent();
    void drawMenu();
    void drawGame();
    std::string getNameGame();
    void destroyWindow();
    std::string getLibName();
    void drawBackground(const std::string &Background);
        std::pair<int, int> sendBgSize() override;
    void clearWindow();
    void updateWindow();
    void drawMain(std::vector<Pixel>);
    void drawSprite(std::vector<Pixel> sprite);
    void putText(const Text &text) override;

    Name::RenderWindow _window;
    Name::Font _font;
    Name::Texture _bgTexture;
    Name::Sprite _bgSprite;
    std::string _gameName;
    std::pair<int, int> _bgSize = std::make_pair(600, 600);
private:
    Name::Time m_elapsedTime;
};
extern "C" IGraphic *createLibrary();
```

When in runtime, you may hit the `[F1]` and `[F2]` key to switch between graphics libraries.

Makefile

> **Notice:** You must add the respective rules to the `Makefile` for them to be compiled properly.

Adding a new game

Here are the requirements for adding new graphics libraries:

- Your library must inherit the given `IGames.hpp` interface
- Your library source files must be located in `src/Games`
- The generated `.so` file must be placed in `lib/`
- The format of the output `.so` file is `arcade_[name].so` where `[name]` is the name of the game
- Game source files must be named as `Name.hpp` & `Name.hpp`

The game must at least incorporate all functions from the `IGames` interface within it's respective header file.

```
class IGames {
    public:
        virtual bool isGameOver() = 0;
                virtual void reset() = 0;
                virtual std::string getName() = 0;
                virtual std::string getBg() = 0;
        virtual int getScore() = 0;

        virtual std::vector<Pixel> getMain() = 0;
        virtual std::vector<Pixel> getSprite() = 0;
        virtual void getInput(MonEnum) = 0;
        virtual void updateGame() = 0;
        virtual void bgSize(std::pair<int, int>) = 0;


    protected:
    private:
};
```

Makefile

Notice: You must add the respective rules to the `Makefile` for them to be compiled properly.

How to Play

The program must take as a startup argument the graphics library to use initially.

You may run the program via the following:

```
$ ./arcade ./lib/arcade_ncurses.so
```

Controls

- `[F1]` - Previous graphics library
- `[F2]` - Next graphics library
- `[F3]` - Previous game
- `[F4]` - Next game
- `[P]` - Pause
- `[ESC]` - Exit

How to build the program (Makefile)

You can build the program via the given `Makefile`

```
make re # Compile the entire program
```