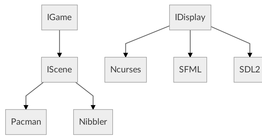
Documentation for Arcade Project

Welcome to Arcade Project

Arcade is a gaming platform: a program that lets the user choose a game to play and the graphic library to use.

Class Diagram



Useful function and macros

arcade::TypeEvent is an enumeration that contains the different types of events that the library can have, for example **arcade::WINDOW**, which corresponds to an event in the window.

Here is the TypeEvent declaration that is in the arcade namespace:

```
namespace arcade {
        enum TypeEvent {
                WINDOW,
                JOYSTICK,
                KEYBOARD,
                MOUSE
        };
}
```

Texture is a structure that will allow the exchange of information concerning the textures to be displayed as well as the texts and the characters.

Here is the Texture & Position declaration:

```
        struct Position {
                int x;
                int y;
        };

        struct Texture {
                std::string path;
                char similar;
                bool isFile;
                bool display;
                Position position;
        };
```

> **NOTE:**
>
> - The **path** variable is the path of the image, **it must be** either **relative** to the arcade executable or **complete**.
> - The **similar** variable is used to **replace the texture with a char** if the current graphic library doesn't support images.
> - The **isFile** variable is set to true if is a file, otherwise **path** is use as a text.
> - If the **display** variable is set to true then the texture is displayed, otherwise the texture is not loaded.
> - The **position** variable is the position of the **top-left** corner of the **Texture**.

Add your Graphics Library

> **Tips**: Your graphics library must be in the folder **lib/** on the root of the project and his name must be like **lib_arcade_${NAME}.so**.

> You must add the following two functions outside your class to allow loading:
>
> ```
> extern "C"  IDisplay    *create_object(size_t width size_t height);
> extern "C"  void                destroy_object(IDisplay *object);
> ```

Your class must inherit from the IDisplay class. Here is the abstract class IDisplay :

```
class IDisplay {
public:
  virtual ~IDisplay() = default;
public:
  virtual bool Display() = 0;
  virtual bool isKey() = 0;                             // Check if an event has been picked up
  virtual bool isOpen() = 0;                    // Check if the current windows is open
  // Check if eventName of typeEvent has been picked up
  virtual bool GetKey(arcade::TypeEvent typeEvent, std::string const &eventName) = 0;
  // Get current texture of the current scene
  virtual bool loadTexture(std::map<std::string, Texture> const &textureList) = 0;
  // Get the map of Characters for library doesn't support Textures (picture, etc...)
  virtual bool loadMap(std::vector<std::vector<char>> const &mapCharacter) = 0;
  // Get current text of the current scene
  virtual bool loadText(std::map<std::string, Texture> const &textureList) = 0;
  virtual void destroyWindow() = 0;
  virtual void changeLibrary(std::string const &path) = 0;
  virtual bool getChange() const = 0;
  virtual std::string const &getNewGamePath() const = 0;
  virtual void setNewGamePath(std::string const &path) = 0;
  virtual bool getSwitchScene() const = 0;
  virtual void setSwitchScene(bool state) = 0;
  virtual void setChange(bool state) = 0;
  virtual std::string const &getLibraryPath() const = 0;
protected:
  // set change to true for change Graphics Library
  bool change;
  // set newLibraryPath to path of the new Graphics Library
  std::string newLibraryPath;
  // set switchScene to true for load other scene
  bool switchScene;
  // path to the scene to load
  std::string newGamePath;
};
```

> **Note:**
>
> - If an exception needs to be thrown, use the **arcade::GraphicsLibraryError** class.
> - All the events that can be used are available in the utils.hpp file.

Add your Game

> **Tips**: Your game library must be in the folder **games/** on the root of the project and his name must be like **lib_arcade_${NAME}.so**.

> You must add the following two functions outside your class to allow loading:
>
> ```
> extern "C"  IGame       *init_game();
> extern "C"  void        destroy_game(IGame *game);
> ```

Your class must inherit from the IGame class. Here is the abstract class IGame :

```
class IGame {
public:
  virtual ~IGame() = default;
  // Getter of name
  virtual std::string    const &getName() const = 0;
  // Getter of description
  virtual std::string    const &getDescription() const = 0;
  // Start is call after the constructor, This function return a IScene pointer
  virtual IScene *start() = 0;
protected:
  std::string name;
  std::string description;
};
```

> **Note**:
>
> - Your game is a class list inherited by the IScene abstract class.
> - IGame is a container of IScene.
> - If an exception needs to be thrown, use the **arcade::LoaderError** class.

> **Note:**
> The idea behind the concept of scene is that each scene has its own event and its own texture, for example, the settings, displaying the game or a menu will each scene be different.

Here is the IScene abstract class:

```
class IScene {
public:
  virtual ~IScene() = default;
  // Handle event for the current scene
  virtual void sceneEvent(IDisplay *) = 0;
  // Send all the necessary textures of the current scene
  virtual textureList getTexture() const = 0;
  // Send all the necessary text of the current scene
  virtual textureList getText() const = 0;
  // Send 2D map of the current scene
  virtual mapChar getMap() const = 0;
  // This function is used for all that is not managed by the user such as moving a mob.
  virtual        void compute() = 0;
};
```

If you wish to have more information please contact us at the following addresses:
- brice.lang-nguyen@epitech.eu
- bastien.guillaumat@epitech.eu
- johanne-franck.ngbokoli@epitech.eu