Arcade

Generated by Doxygen 1.9.6

Chapter 1

Arcade - Graphical / Game Library usage

1.0.1 This document will explain how to create your own graphical / game library.

1.1 Introduction

Arcade relies on the use of a core to handle libraries. Those libraries are loaded dynamically at runtime. This allows you to create your own graphical / game library by following those steps.

1.2 Implementation

1. Insert your library in the `lib` folder.
2. The library must be named as follows: `lib_arcade_[library_name].so`.
3. This library must either implement `IDisplay` or `IGame`.
4. The library must export two main functions in order to be loaded and destroyed when needed:
   {c++}

```
// For a graphical library
extern "C"
{
    void *createDisplay() {
        NCurses *lib = new NCurses();

        lib->init();
        return lib;
    }

    void deleteDisplay(void *display) {
        NCurses *lib = reinterpret_cast<NCurses *>(display);

        lib->exit();
        delete lib;
    }
}

// For a game library
extern "C"
{
    Arcade::IGame *createGame() {
        return new Pacman();
    }

    void deleteGame(Arcade::IGame *display) {
        delete dynamic_cast<Pacman *>(display);
    }
}
```

   As shown above, it is recommended to implement a `init` and `exit` function to avoid constructor and destructor call issues. You also can use `void *` or `<MYLIB> *` as delete function parameter. It is also preferred not to use unique pointers as they won't be handled as we expect them to be.

1.3 Assets

GAME LIBRARY ONLY
In `ASSETS/[YOUR_GAME_NAME]/[GRAPHICAL_LIBRARY_NAME]`, add your assets.

GRAPHICAL LIBRARY ONLY
You might want to implement texture loading as follows:

- All your assets are fetched from the `ASSETS` folder.
- The assets are loaded from the `ASSETS/[GRAPHICAL_LIBRARY_NAME]/[GAME_NAME]` folder. (You can get the game name from the `IGame` interface)

1.4 Misc

- Make sure to link all needed libraries in your .so file.
- Arcade core uses C++20 features.

Chapter 2

Namespace Index

2.1 Namespace List

Here is a list of all namespaces with brief descriptions:

Chapter 3

Class Index

3.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

Chapter 5

Namespace Documentation

5.1 Arcade Namespace Reference

Namespace containing of all the arcade classes.

Classes

- class IDisplay

  Interface for Display Libraries.
- class IEntity

  Interface of an entity.
- class IGame

  Interface of the game.
- class IGameData

  Interface of the game data.

Typedefs

- typedef std::unordered_map< std::string, std::string > ControlMap

Enumerations

- enum Key {
  Unknown = -1 , A = 0 , B , C ,
  D , E , F , G ,
  H , I , J , K ,
  L , M , N , O ,
  P , Q , R , S ,
  T , U , V , W ,
  X , Y , Z , Num0 ,
  Num1 , Num2 , Num3 , Num4 ,
  Num5 , Num6 , Num7 , Num8 ,
  Num9 , Escape , LControl , LShift ,
  LAlt , LSystem , RControl , RShift ,
  RAlt , RSystem , Menu , LBracket ,

RBracket , Semicolon , Comma , Period ,
Apostrophe , Slash , Backslash , Grave ,
Equal , Hyphen , Space , Enter ,
Backspace , Tab , PageUp , PageDown ,
End , Home , Insert , Delete ,
Add , Subtract , Multiply , Divide ,
Left , Right , Up , Down ,
Numpad0 , Numpad1 , Numpad2 , Numpad3 ,
Numpad4 , Numpad5 , Numpad6 , Numpad7 ,
Numpad8 , Numpad9 , F1 , F2 ,
F3 , F4 , F5 , F6 ,
F7 , F8 , F9 , F10 ,
F11 , F12 , F13 , F14 ,
F15 , Pause , KeyCount }

Enum of all the possible keys that can be pressed.

5.1.1 Detailed Description

Namespace containing of all the arcade classes.

5.1.2 Typedef Documentation

5.1.2.1 ControlMap

```
typedef std::unordered_map<std::string, std::string> Arcade::ControlMap
```

5.1.3 Enumeration Type Documentation

5.1.3.1 Key

```
enum Arcade::Key
```

Enum of all the possible keys that can be pressed.

Enumerator

Unknown	Unhandled key.
A	The A key.
B	The B key.
C	The C key.
D	The D key.
E	The E key.
F	The F key.
G	The G key.

Enumerator

H	The H key.
I	The I key.
J	The J key.
K	The K key.
L	The L key.
M	The M key.
N	The N key.
O	The O key.
P	The P key.
Q	The Q key.
R	The R key.
S	The S key.
T	The T key.
U	The U key.
V	The V key.
W	The W key.
X	The X key.
Y	The Y key.
Z	The Z key.
Num0	The 0 key.
Num1	The 1 key.
Num2	The 2 key.
Num3	The 3 key.
Num4	The 4 key.
Num5	The 5 key.
Num6	The 6 key.
Num7	The 7 key.
Num8	The 8 key.
Num9	The 9 key.
Escape	The Escape key.
LControl	The left Control key.
LShift	The left Shift key.
LAlt	The left Alt key.
LSystem	The left OS specific key: window (Windows and Linux), apple (macOS), ...
RControl	The right Control key.
RShift	The right Shift key.
RAlt	The right Alt key.
RSystem	The right OS specific key: window (Windows and Linux), apple (macOS), ...
Menu	The Menu key.
LBracket	The [key.
RBracket	The] key.
Semicolon	The ; key.
Comma	The , key.
Period	The . key.
Apostrophe	The ' key.
Slash	The / key.
Backslash	The \ key.

Enumerator

Grave	The ` key.
Equal	The = key.
Hyphen	The - key (hyphen)
Space	The Space key.
Enter	The Enter/Return keys.
Backspace	The Backspace key.
Tab	The Tabulation key.
PageUp	The Page up key.
PageDown	The Page down key.
End	The End key.
Home	The Home key.
Insert	The Insert key.
Delete	The Delete key.
Add	The + key.
Subtract	The - key (minus, usually from numpad)
Multiply	The $*$ key.
Divide	The / key.
Left	Left arrow.
Right	Right arrow.
Up	Up arrow.
Down	Down arrow.
Numpad0	The numpad 0 key.
Numpad1	The numpad 1 key.
Numpad2	The numpad 2 key.
Numpad3	The numpad 3 key.
Numpad4	The numpad 4 key.
Numpad5	The numpad 5 key.
Numpad6	The numpad 6 key.
Numpad7	The numpad 7 key.
Numpad8	The numpad 8 key.
Numpad9	The numpad 9 key.
F1	The F1 key.
F2	The F2 key.
F3	The F3 key.
F4	The F4 key.
F5	The F5 key.
F6	The F6 key.
F7	The F7 key.
F8	The F8 key.
F9	The F9 key.
F10	The F10 key.
F11	The F11 key.
F12	The F12 key.
F13	The F13 key.
F14	The F14 key.
F15	The F15 key.
Pause	The Pause key.

Enumerator

KeyCount	Keep last – the total number of keyboard keys.

Chapter 6

Class Documentation

6.1 Arcade::IDisplay Class Reference

Interface for Display Libraries.

`#include <IDisplayModule.hpp>`

Public Member Functions

- virtual ~IDisplay ()=default
- virtual std::vector< Key > getPressedKeys ()=0

  Get the pressed keys.
- virtual void render (IGameData &gameData)=0

  Renders the game.
- virtual void renderMenu (const std::vector< std::string > &games, const std::vector< std::string > &graphics, int selectedGame, int selectedDisplay, const ControlMap &controls, const std::string &username, const std::string &bestScoreUsername, int bestScore)=0

  Renders the menu.

6.1.1 Detailed Description

Interface for Display Libraries.

This interface is used to create a display library. The display library is used to render the menu and the selected game. A display library must export two symbols : "createDisplay" and "destroyDisplay".

6.1.2 Constructor & Destructor Documentation

6.1.2.1 ~IDisplay()

`virtual Arcade::IDisplay::~IDisplay ( )  [virtual], [default]`

6.1.3 Member Function Documentation

6.1.3.1 getPressedKeys()

```
virtual std::vector< Key > Arcade::IDisplay::getPressedKeys ( )  [pure virtual]
```

Get the pressed keys.

This method return a list of Arcade::Key enum

Returns

A vector containing all the pressed keys by the user.

6.1.3.2 render()

```
virtual void Arcade::IDisplay::render (
            IGameData & gameData )  [pure virtual]
```

Renders the game.

This method is continuously called to render the game. It takes as parameter a IGameData object, containing all the data needed to render the game and any useful information. Textures are loaded from the path: ASSETS + GAME_NAME + "/" + LIB_NAME + "/" + textureName

Parameters

gameData	IGameData containing the entities, scores, controls, mapSize and gameOver.

6.1.3.3 renderMenu()

```
virtual void Arcade::IDisplay::renderMenu (
            const std::vector< std::string > & games,
            const std::vector< std::string > & graphics,
            int selectedGame,
            int selectedDisplay,
            const ControlMap & controls,
            const std::string & username,
            const std::string & bestScoreUsername,
            int bestScore )  [pure virtual]
```

Renders the menu.

Parameters

games	The list the game libraries
graphics	The list the graphics libraries
selectedGame	The index of the selected game
selectedDisplay	The index of the selected display
controls	A map, associating a key with string type to an action with a string type too

The documentation for this class was generated from the following file:

- IDisplayModule.hpp

6.2 Arcade::IEntity Class Reference

Interface of an entity.

`#include <IGameModule.hpp>`

Public Member Functions

- virtual std::vector< std::pair< float, float > > getPosition () const =0

  Gets all the positions of the entity.
- virtual std::pair< float, float > getSize () const =0

  Gets the size of the entity.
- virtual std::string getTexture () const =0

  Gets the texture name of the entity.
- virtual float getRotation () const =0

  Gets the rotation of the entity.

6.2.1 Detailed Description

Interface of an entity.

An entity is an object that can be displayed on the screen. Multiple entities can be displayed by adding a position to the entity.

6.2.2 Member Function Documentation

6.2.2.1 getPosition()

```
virtual std::vector< std::pair< float, float > > Arcade::IEntity::getPosition ( ) const  [pure virtual]
```

Gets all the positions of the entity.

Returns

A vector containing all the positions of the entity.

6.2.2.2 getRotation()

```
virtual float Arcade::IEntity::getRotation ( ) const  [pure virtual]
```

Gets the rotation of the entity.

Returns

The entity rotation in degrees.

6.2.2.3 getSize()

```
virtual std::pair< float, float > Arcade::IEntity::getSize ( ) const  [pure virtual]
```

Gets the size of the entity.

Returns

A pair of float representing the size of the entity.

6.2.2.4 getTexture()

```
virtual std::string Arcade::IEntity::getTexture ( ) const  [pure virtual]
```

Gets the texture name of the entity.

Returns

The entity texture name.

The documentation for this class was generated from the following file:

- IGameModule.hpp

6.3 Arcade::IGame Class Reference

Interface of the game.

`#include <IGameModule.hpp>`

Public Member Functions

- virtual ~IGame ()=default
- virtual void handleKeys (const std::vector< Key > &pressedKeys)=0

  Handles the keys pressed by the user.
- virtual void update (const std::string &username)=0

  Updates the game depending on a game tick (clock).
- virtual IGameData & getGameData () const =0

  Gets the game data.

6.3.1 Detailed Description

Interface of the game.

This interface is used to interact with the game. It contains the three main methods for the game to work.

6.3.2 Constructor & Destructor Documentation

6.3.2.1 ~IGame()

```
virtual Arcade::IGame::~IGame ( )  [virtual], [default]
```

6.3.3 Member Function Documentation

6.3.3.1 getGameData()

```
virtual IGameData & Arcade::IGame::getGameData ( ) const  [pure virtual]
```

Gets the game data.

Returns

An instance of IGameData containing all needed information.

6.3.3.2 handleKeys()

```
virtual void Arcade::IGame::handleKeys (
            const std::vector< Key > & pressedKeys )  [pure virtual]
```

Handles the keys pressed by the user.

This method receives the keys pressed by the user to handle them.

Parameters

pressedKeys	A vector of keys get from the display.

6.3.3.3 update()

```
virtual void Arcade::IGame::update (
            const std::string & username )  [pure virtual]
```

Updates the game depending on a game tick (clock).

This method is called continuously by the core to update the game.

Parameters

username	The username of the player.

The documentation for this class was generated from the following file:

- IGameModule.hpp

6.4 Arcade::IGameData Class Reference

Interface of the game data.

```
#include <IGameModule.hpp>
```

Public Member Functions

- virtual std::map< std::string, int > getScores () const =0
  Gets the scores of the game.
- virtual std::string getGameName () const =0
  Gets the name of the game.
- virtual std::vector< std::shared_ptr< Arcade::IEntity > > & getEntities ()=0
  Gets all game entities to display.
- virtual std::pair< int, int > getMapSize () const =0
  Gets the size of the map.
- virtual const ControlMap & getControls () const =0
  Gets the controls of the game.
- virtual bool isGameOver () const =0
  Returns either the game is over or not.

6.4.1 Detailed Description

Interface of the game data.

This interface is used to get the data of the game. Data are send to the display through the core to be interpreted and displayed.

6.4.2 Member Function Documentation

6.4.2.1 getControls()

```
virtual const ControlMap & Arcade::IGameData::getControls ( ) const  [pure virtual]
```

Gets the controls of the game.

Contains the controls of the game needed to play.

Returns

A map of name => control.

6.4.2.2 getEntities()

```
virtual std::vector< std::shared_ptr< Arcade::IEntity > > & Arcade::IGameData::getEntities (
) [pure virtual]
```

Gets all game entities to display.

Array of every entity active in the game at the moment.

Returns

6.4.2.3 getGameName()

```
virtual std::string Arcade::IGameData::getGameName ( ) const  [pure virtual]
```

Gets the name of the game.

Returns

The name of the game.

6.4.2.4 getMapSize()

```
virtual std::pair< int, int > Arcade::IGameData::getMapSize ( ) const  [pure virtual]
```

Gets the size of the map.

Returns

The size of the map in terms of cell.

6.4.2.5 getScores()

```
virtual std::map< std::string, int > Arcade::IGameData::getScores ( ) const  [pure virtual]
```

Gets the scores of the game.

Returns

A map of string and int. The key is the name of the player and the value is the score of the player.

6.4.2.6 isGameOver()

```
virtual bool Arcade::IGameData::isGameOver ( ) const  [pure virtual]
```

Returns either the game is over or not.

Returns

True if the game is over, false otherwise.

The documentation for this class was generated from the following file:

- IGameModule.hpp

Chapter 7

File Documentation

7.1 IDisplayModule.hpp File Reference

```
#include "IGameModule.hpp"
#include <fstream>
#include <sstream>
```

Include dependency graph for IDisplayModule.hpp:

7.2 IDisplayModule.hpp

Go to the documentation of this file.

```
00001 /*
00002 ** EPITECH PROJECT, 2023
00003 ** B-OOP-400-LIL-4-1-arcade-noa.trachez
00004 ** File description:
00005 ** IDisplayModule.hpp
00006 */
00007
00008 #ifndef IDISPLAY_HPP_
00009     #define IDISPLAY_HPP_
00010 #include "IGameModule.hpp"
00011 #include <fstream>
00012 #include <sstream>
00013
00014 namespace Arcade {
00015     class IDisplay;
00016 };
00017
00025 class Arcade::IDisplay {
00026 public:
00027     virtual ~IDisplay() = default;
00028
00035     virtual std::vector<Key> getPressedKeys() = 0;
00036
00045     virtual void render(IGameData &gameData) = 0;
00046
00056     virtual void renderMenu(const std::vector<std::string> &games, const std::vector<std::string>
      &graphics,
00057                             int selectedGame, int selectedDisplay, const ControlMap &controls,
00058                             const std::string &username, const std::string &bestScoreUsername, int
      bestScore) = 0;
00059 };
00060
00061 #endif /* !IDISPLAY_HPP_ */
```

7.3 IGameModule.hpp File Reference

```
#include <vector>
#include <unordered_map>
#include <map>
#include <memory>
#include <string>
```

Include dependency graph for IGameModule.hpp: This graph shows which files directly or indirectly include this file:

Classes

- class Arcade::IEntity

  Interface of an entity.
- class Arcade::IGameData

  Interface of the game data.
- class Arcade::IGame

  Interface of the game.

Namespaces

- namespace Arcade

  Namespace containing of all the arcade classes.

Typedefs

- typedef std::unordered_map< std::string, std::string > Arcade::ControlMap

Enumerations

- enum Arcade::Key {
  Arcade::Unknown = -1 , Arcade::A = 0 , Arcade::B , Arcade::C ,
  Arcade::D , Arcade::E , Arcade::F , Arcade::G ,
  Arcade::H , Arcade::I , Arcade::J , Arcade::K ,
  Arcade::L , Arcade::M , Arcade::N , Arcade::O ,
  Arcade::P , Arcade::Q , Arcade::R , Arcade::S ,
  Arcade::T , Arcade::U , Arcade::V , Arcade::W ,
  Arcade::X , Arcade::Y , Arcade::Z , Arcade::Num0 ,
  Arcade::Num1 , Arcade::Num2 , Arcade::Num3 , Arcade::Num4 ,
  Arcade::Num5 , Arcade::Num6 , Arcade::Num7 , Arcade::Num8 ,
  Arcade::Num9 , Arcade::Escape , Arcade::LControl , Arcade::LShift ,
  Arcade::LAlt , Arcade::LSystem , Arcade::RControl , Arcade::RShift ,
  Arcade::RAlt , Arcade::RSystem , Arcade::Menu , Arcade::LBracket ,
  Arcade::RBracket , Arcade::Semicolon , Arcade::Comma , Arcade::Period ,
  Arcade::Apostrophe , Arcade::Slash , Arcade::Backslash , Arcade::Grave ,
  Arcade::Equal , Arcade::Hyphen , Arcade::Space , Arcade::Enter ,
  Arcade::Backspace , Arcade::Tab , Arcade::PageUp , Arcade::PageDown ,
  Arcade::End , Arcade::Home , Arcade::Insert , Arcade::Delete ,
  Arcade::Add , Arcade::Subtract , Arcade::Multiply , Arcade::Divide ,
  Arcade::Left , Arcade::Right , Arcade::Up , Arcade::Down ,
  Arcade::Numpad0 , Arcade::Numpad1 , Arcade::Numpad2 , Arcade::Numpad3 ,

Arcade::Numpad4 , Arcade::Numpad5 , Arcade::Numpad6 , Arcade::Numpad7 ,
Arcade::Numpad8 , Arcade::Numpad9 , Arcade::F1 , Arcade::F2 ,
Arcade::F3 , Arcade::F4 , Arcade::F5 , Arcade::F6 ,
Arcade::F7 , Arcade::F8 , Arcade::F9 , Arcade::F10 ,
Arcade::F11 , Arcade::F12 , Arcade::F13 , Arcade::F14 ,
Arcade::F15 , Arcade::Pause , Arcade::KeyCount }

Enum of all the possible keys that can be pressed.

7.4 IGameModule.hpp

Go to the documentation of this file.

```
00001 /*
00002 ** EPITECH PROJECT, 2023
00003 ** B-OOP-400-LIL-4-1-arcade-noa.trachez
00004 ** File description:
00005 ** IGameModule.hpp
00006 */
00007
00008 #ifndef IGAME_HPP_
00009     #define IGAME_HPP_
00010
00011 #include <vector>
00012 #include <unordered_map>
00013 #include <map>
00014 #include <memory>
00015 #include <string>
00016
00020 namespace Arcade {
00021
00025     enum Key {
00026         Unknown = -1,
00027         A       = 0,
00028         B,
00029         C,
00030         D,
00031         E,
00032         F,
00033         G,
00034         H,
00035         I,
00036         J,
00037         K,
00038         L,
00039         M,
00040         N,
00041         O,
00042         P,
00043         Q,
00044         R,
00045         S,
00046         T,
00047         U,
00048         V,
00049         W,
00050         X,
00051         Y,
00052         Z,
00053         Num0,
00054         Num1,
00055         Num2,
00056         Num3,
00057         Num4,
00058         Num5,
00059         Num6,
00060         Num7,
00061         Num8,
00062         Num9,
00063         Escape,
00064         LControl,
00065         LShift,
00066         LAlt,
00067         LSystem,
00068         RControl,
00069         RShift,
00070         RAlt,
00071         RSystem,
00072         Menu,
00073         LBracket,
```

```
00074         RBracket,
00075         Semicolon,
00076         Comma,
00077         Period,
00078         Apostrophe,
00079         Slash,
00080         Backslash,
00081         Grave,
00082         Equal,
00083         Hyphen,
00084         Space,
00085         Enter,
00086         Backspace,
00087         Tab,
00088         PageUp,
00089         PageDown,
00090         End,
00091         Home,
00092         Insert,
00093         Delete,
00094         Add,
00095         Subtract,
00096         Multiply,
00097         Divide,
00098         Left,
00099         Right,
00100         Up,
00101         Down,
00102         Numpad0,
00103         Numpad1,
00104         Numpad2,
00105         Numpad3,
00106         Numpad4,
00107         Numpad5,
00108         Numpad6,
00109         Numpad7,
00110         Numpad8,
00111         Numpad9,
00112         F1,
00113         F2,
00114         F3,
00115         F4,
00116         F5,
00117         F6,
00118         F7,
00119         F8,
00120         F9,
00121         F10,
00122         F11,
00123         F12,
00124         F13,
00125         F14,
00126         F15,
00127         Pause,
00128         KeyCount,
00129     };
00130
00131     typedef std::unordered_map<std::string, std::string> ControlMap;
00132
00139     class IEntity {
00140     public:
00145         virtual std::vector<std::pair<float, float» getPosition() const = 0;
00146
00151         virtual std::pair<float, float> getSize() const = 0;
00152
00157         virtual std::string getTexture() const = 0;
00158
00163         virtual float getRotation() const = 0;
00164     };
00165
00172     class IGameData {
00173     public:
00178         virtual std::map<std::string, int> getScores() const = 0;
00179
00184         virtual std::string getGameName() const = 0;
00185
00192         virtual std::vector<std::shared_ptr<Arcade::IEntity» &getEntities() = 0;
00193
00198         virtual std::pair<int, int> getMapSize() const = 0;
00199
00206         virtual const ControlMap &getControls() const = 0;
00207
00212         virtual bool isGameOver() const = 0;
00213     };
00214
00221     class IGame {
00222     public:
```

```
00223         virtual ~IGame() = default;
00224
00231         virtual void handleKeys(const std::vector<Key> &pressedKeys) = 0;
00232
00239         virtual void update(const std::string &username) = 0;
00240
00245         virtual IGameData &getGameData() const = 0;
00246     };
00247 };
00248
00249 #endif /* !IGAME_HPP_ */
```

7.5 LIB.md File Reference