zappy_gui

0.1.0

Generated by Doxygen 1.9.7

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 2

Class Index

2.1 Class List

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

Chapter 3

File Index

3.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 4

Class Documentation

4.1 gui::Argument Class Reference

Public Member Functions

- **Argument** (const uint16_t p, std::string h)

Public Attributes

- const uint16_t **port**
- const std::string **hostName**

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

- include/GUI/Argument.hpp

4.2 myLib::Clock Class Reference

Public Member Functions

- void **restart** ()
- void **pause** ()
- void **resume** ()
- Time **getElapsedTime** () const

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

- lib/static/myLib/include/myLib/Clock/Clock.hpp

4.3 gui::Gui Class Reference

Public Types

- enum class **RendererMode** { **GAME** , **SETTINGS** }

Public Member Functions

- **Gui** (const Argument &args)
- std::unique_ptr< IRenderer > & **getRenderer** ()
- void **Run** ()

Static Public Member Functions

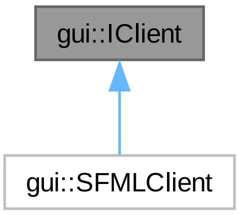
- static std::vector< std::string > **getData** (const std::string &data)

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

- include/GUI/Gui.hpp

4.4 gui::IClient Class Reference

Inheritance diagram for gui::IClient:



Public Member Functions

- virtual bool **connect** (uint16_t port, const std::string &machineName)=0
- virtual void **disconnect** ()=0
- virtual bool **sendCommand** (const std::string &cmd)=0
- virtual bool **getResponse** (const std::string &cmd)=0
- virtual std::string **getResponse** ()=0
- virtual bool **isConnected** ()=0

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

- include/GUI/Abstraction/IClient.hpp

4.5 gui::Inventory Class Reference

Public Member Functions

- **Inventory** (Resource food, Resource linemate, Resource deraumere, Resource sibur, Resource mendiane, Resource phiras, Resource thystame)
- **Inventory** (std::vector< Resource > cresources)

Public Attributes

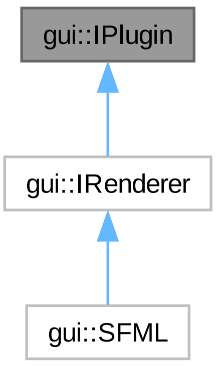
- std::vector< Resource > **resources**

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

- include/GUI/Inventory/Inventory.hpp

4.6 gui::IPlugin Class Reference

Inheritance diagram for gui::IPlugin:



Public Member Functions

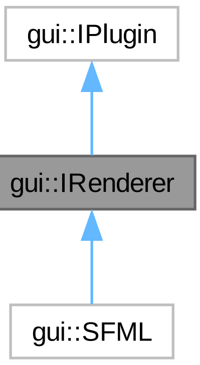
- virtual std::string **getPluginName** () const =0

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

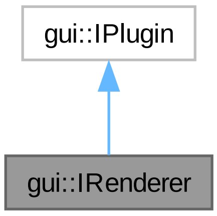
- include/GUI/Abstraction/IPlugin.hpp

4.7 gui::IRenderer Class Reference

Inheritance diagram for gui::IRenderer:



Collaboration diagram for gui::IRenderer:



Public Member Functions

- virtual void **setFPS** (unsigned int FPS)=0
- virtual IClient & **getClient** ()=0
- virtual bool **isRunning** ()=0
- virtual void **init** (const std::string &name, std::pair< const unsigned int, const unsigned int > resolution, unsigned int bitsPerPixel)=0
- virtual void **render** ()=0
- virtual KeyBoard::Key **getEvents** ()=0
- virtual void **close** ()=0

Public Member Functions inherited from gui::IPlugin

- virtual std::string **getPluginName** () const =0

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

- include/GUI/Abstraction/IRenderer.hpp

4.8 gui::KeyBoard Class Reference

Public Types

- enum **Key** {
  NONE = -1 , **CLOSE** = 0 , **KEY_LEFT** = 1 , **KEY_RIGHT** = 2 ,
  KEY_UP = 3 , **KEY_DOWN** = 4 , **KEY_SPACE** = 5 , **KEY_ENTER** = 6 ,
  KEY_ESCAPE = 7 , **COUNT** = 8 }

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

- include/GUI/KeyBoard.hpp

4.9 gui::Map Class Reference

Public Member Functions

- **Map** (unsigned int width, unsigned int height)
- unsigned int **getWidth** () const
- unsigned int **getHeight** () const

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

- lib/shared/Renderer/SFML/include/GUI/Map.hpp

4.10 gui::Mob Class Reference

Public Types

- enum class **Action** { **MOVE** , **FEED** , **ELEVATE** , **NONE** }

Public Member Functions

- Action **getAction** () const
- Inventory & **getInventory** ()
- Position & **getPosition** ()
- unsigned int **getLevel** () const
- void **setAction** (const Action &action)
- void **levelUp** ()

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

- lib/shared/Renderer/SFML/include/GUI/Mob.hpp

4.11 gui::Parser Class Reference

Classes

- class ParserException

Static Public Member Functions

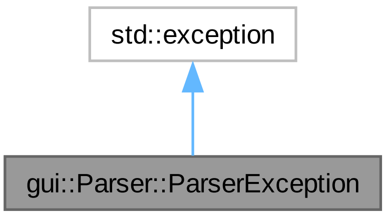
- static Argument **ParseArgs** (int argc, char ∗const argv[ ])
- static uint16_t **ParsePort** (const char ∗port)
- static std::string **ParseMachineName** (const char ∗machineName)
- static void **processData** (std::vector< std::string > data, Gui &gui)
- static Inventory **parseTileContent** (std::string tileContent)

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

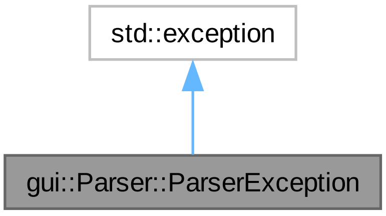
- include/GUI/Parser.hpp

4.12 gui::Parser::ParserException Class Reference

Inheritance diagram for gui::Parser::ParserException:



Collaboration diagram for gui::Parser::ParserException:



Public Member Functions

- **ParserException** (std::string msg)
- **ParserException** (const ParserException &)=delete
- ParserException & **operator=** (const ParserException &)=delete
- **ParserException** (const ParserException &&)=delete
- ParserException & **operator=** (const ParserException &&)=delete
- const char $*$ **what** () const noexcept override

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

- include/GUI/Parser.hpp

4.13 gui::PluginLoader Class Reference

Classes

- class PluginLoaderException

Public Types

- using **PluginCreator** = std::unique_ptr< IPlugin >($*$)()

Public Member Functions

- template<typename T >
  std::unique_ptr< T > **getPlugin** (const std::string &pluginName)
- void **closePlugins** ()

Static Public Member Functions

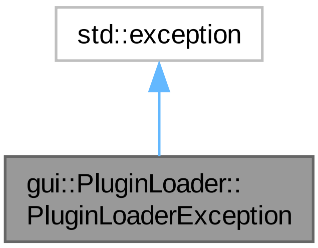
- static PluginLoader & **getInstance** ()

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

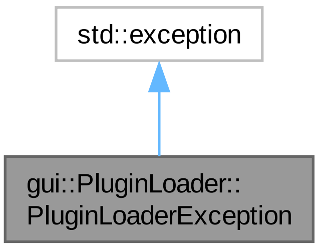
- include/GUI/PluginLoader.hpp

4.14 gui::PluginLoader::PluginLoaderException Class Reference

Inheritance diagram for gui::PluginLoader::PluginLoaderException:



Collaboration diagram for gui::PluginLoader::PluginLoaderException:



Public Member Functions

- **PluginLoaderException** (std::string msg)
- const char $*$ **what** () const noexcept override

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

- include/GUI/PluginLoader.hpp

4.15 gui::Protocol Class Reference

Static Public Member Functions

- static std::string **getCommand** (ProtocolKey key)
- static ProtocolKey **getKey** (const std::string &command)

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

- include/GUI/Protocol.hpp

4.16 myLib::Random Class Reference

Static Public Member Functions

- static int **randomInt** (int min, int max)
- static int **randomInt** ()
- static float **randomFloat** (float min, float max)
- static float **randomFloat** ()

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

- lib/static/myLib/include/myLib/Random.hpp

4.17 gui::Resource Class Reference

Public Types

- enum class **Type** {
  FOOD , **LINEMATE** , **DERAUMERE** , **SIBUR** ,
  MENDIANE , **PHIRAS** , **THYSTAME** , **NONE** }

Public Member Functions

- **Resource** (Type type, unsigned int quantity)
- bool **operator==** (const Resource &resource) const

Public Attributes

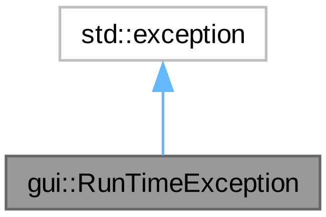
- Type **type**
- double **density**
- unsigned int **quantity**

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

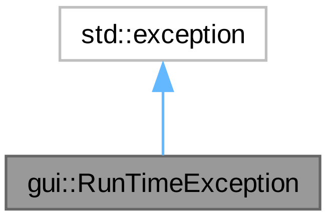
- include/GUI/Inventory/Resource.hpp

4.18 gui::RunTimeException Class Reference

Inheritance diagram for gui::RunTimeException:



Collaboration diagram for gui::RunTimeException:



Public Member Functions

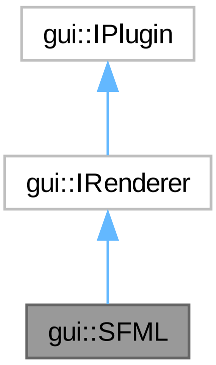
- **RunTimeException** (std::string msg)
- **RunTimeException** (const RunTimeException &)=delete
- RunTimeException & **operator=** (const RunTimeException &)=delete
- **RunTimeException** (const RunTimeException &&)=delete
- RunTimeException & **operator=** (const RunTimeException &&)=delete
- const char ∗ **what** () const noexcept override

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

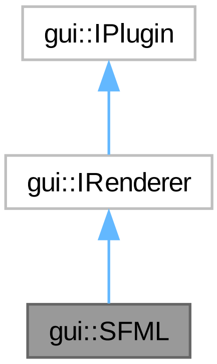
- include/GUI/RunTimeException.hpp

4.19 gui::SFML Class Reference

Inheritance diagram for gui::SFML:



Collaboration diagram for gui::SFML:



Public Member Functions

- void setFPS (const unsigned int FPS) override
- std::string getPluginName () const override
- IClient & getClient () override
- KeyBoard::Key getEvents () override
- bool isRunning () override
- void init (const std::string &name, std::pair< const unsigned int, const unsigned int > resolution, unsigned int bitsPerPixel) override

- void close () override
- void render () override
- bool **checkConnection** (sf::Clock clock)

- virtual void **setFPS** (unsigned int FPS)=0
- virtual IClient & **getClient** ()=0
- virtual bool **isRunning** ()=0
- virtual void **init** (const std::string &name, std::pair< const unsigned int, const unsigned int > resolution, unsigned int bitsPerPixel)=0
- virtual void **render** ()=0
- virtual KeyBoard::Key **getEvents** ()=0
- virtual void **close** ()=0

- virtual std::string **getPluginName** () const =0

Static Public Member Functions

- static KeyBoard::Key **getKeyboardEvent** (const sf::Event &event)

4.19.1 Member Function Documentation

4.19.1.1 close()

```
void gui::SFML::close ( )  [inline], [override], [virtual]
```

Implements gui::IRenderer.

4.19.1.2 getClient()

```
IClient & gui::SFML::getClient ( )  [inline], [override], [virtual]
```

Implements gui::IRenderer.

4.19.1.3 getEvents()

```
KeyBoard::Key gui::SFML::getEvents ( )  [override], [virtual]
```

Implements gui::IRenderer.

4.19.1.4 getPluginName()

```
std::string gui::SFML::getPluginName ( ) const  [inline], [override], [virtual]
```

Implements gui::IPlugin.

4.19.1.5 init()

```
void gui::SFML::init (
             const std::string & name,
             std::pair< const unsigned int, const unsigned int > resolution,
             unsigned int bitsPerPixel )  [override], [virtual]
```

Implements gui::IRenderer.

4.19.1.6 isRunning()

```
bool gui::SFML::isRunning ( )  [inline], [override], [virtual]
```

Implements gui::IRenderer.

4.19.1.7 render()

```
void gui::SFML::render ( )  [override], [virtual]
```

Implements gui::IRenderer.

4.19.1.8 setFPS()

```
void gui::SFML::setFPS (
             const unsigned int FPS )  [inline], [override], [virtual]
```

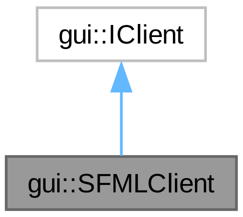
Implements gui::IRenderer.

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

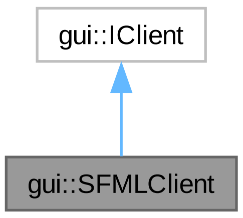
- lib/shared/Renderer/SFML/include/GUI/SFML.hpp

4.20 gui::SFMLClient Class Reference

Inheritance diagram for gui::SFMLClient:



Collaboration diagram for gui::SFMLClient:



Public Member Functions

- bool connect (uint16_t port, const std::string &machineName) override
- void disconnect () override
- bool sendCommand (const std::string &cmd) override
- bool getResponse (const std::string &cmd) override
- std::string getResponse () override
- bool isConnected () override

- virtual bool **connect** (uint16_t port, const std::string &machineName)=0
- virtual void **disconnect** ()=0
- virtual bool **sendCommand** (const std::string &cmd)=0
- virtual bool **getResponse** (const std::string &cmd)=0
- virtual std::string **getResponse** ()=0
- virtual bool **isConnected** ()=0

4.20.1 Member Function Documentation

4.20.1.1 connect()

```
bool gui::SFMLClient::connect (
            uint16_t port,
            const std::string & machineName )  [override], [virtual]
```

Implements gui::IClient.

4.20.1.2 disconnect()

```
void gui::SFMLClient::disconnect ( )  [inline], [override], [virtual]
```

Implements gui::IClient.

4.20.1.3 getResponse() [1/2]

```
std::string gui::SFMLClient::getResponse ( )  [override], [virtual]
```

Implements gui::IClient.

4.20.1.4 getResponse() [2/2]

```
bool gui::SFMLClient::getResponse (
            const std::string & cmd )  [override], [virtual]
```

Implements gui::IClient.

4.20.1.5 isConnected()

```
bool gui::SFMLClient::isConnected ( )  [override], [virtual]
```

Implements gui::IClient.

4.20.1.6 sendCommand()

```
bool gui::SFMLClient::sendCommand (
            const std::string & cmd )  [override], [virtual]
```

Implements gui::IClient.

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

- lib/shared/Renderer/SFML/include/GUI/SFMLClient.hpp

4.21 myLib::Time Class Reference

Public Member Functions

- **Time** (const double seconds)
- int **asSeconds** () const
- int **asMilliseconds** () const
- int **asMicroseconds** () const

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

- lib/static/myLib/include/myLib/Clock/Time.hpp

Chapter 5

File Documentation

5.1 IClient.hpp

```
/*
** EPITECH PROJECT, 2024
** zappy_gui
** File description:
** IClient
*/

#pragma once

#include <string>
#include <cstdint>

namespace gui {

    class IClient {

        public:

            virtual ~IClient() = default;

            virtual bool connect(uint16_t port, const std::string& machineName) = 0;
            virtual void disconnect() = 0;
            virtual bool sendCommand(const std::string& cmd) = 0;
            virtual bool getResponse(const std::string& cmd) = 0;
            virtual std::string getResponse() = 0;
            virtual bool isConnected() = 0;

    }; // class IClient

} // namespace gui
```

5.2 IPlugin.hpp

```
/*
** EPITECH PROJECT, 2024
** zappy_gui
** File description:
** IPlugin
*/

#pragma once

#include <string>

namespace gui {

    class IPlugin {

    public:

        virtual ~IPlugin() = default;

        [[nodiscard]] virtual std::string getPluginName() const = 0;

    }; // class IPlugin

} // namespace gui
```

5.3 IRenderer.hpp

```
/*
** EPITECH PROJECT, 2024
** zappy_gui
** File description:
** IRenderer
*/

#pragma once

#include "GUI/Abstraction/IPlugin.hpp"
#include "GUI/Abstraction/IClient.hpp"
#include "GUI/KeyBoard.hpp"

namespace gui {

    class IRenderer : public IPlugin {

        public:

            virtual void setFPS(unsigned int FPS) = 0;

            [[nodiscard]] virtual IClient& getClient() = 0;
            [[nodiscard]] virtual bool isRunning() = 0;

            virtual void init(const std::string &name, std::pair<const unsigned int,const unsigned
int> resolution, unsigned int bitsPerPixel) = 0;
            virtual void render() = 0;
            virtual KeyBoard::Key getEvents() = 0;
            virtual void close() = 0;

    }; // class IRenderer

} // namespace gui
```

5.4 Argument.hpp

```
/*
** EPITECH PROJECT, 2024
** zappy_gui
** File description:
** Argument
*/

#pragma once

#include <string>
#include <cstdint>

namespace gui {

    class Argument {

        public:

            Argument(const uint16_t p, std::string h) : port(p), hostName(std::move(h)) {};
            ~Argument() = default;

            const uint16_t port;
            const std::string hostName;

    }; // class Argument

} // namespace gui
```

5.5 Constant.hpp

```
/*
** EPITECH PROJECT, 2024
** Zappy | GUI
** File description:
** Constant.hpp
*/

/*
** @file Constant.hpp
** @brief Constants for the Zappy GUI
** @namespace gui
```

```
*/

#pragma once

#include <string_view>

namespace gui {

    static constexpr const int EPITECH_EXIT_SUCCESS = 0;
    static constexpr const int EPITECH_EXIT_ERROR = 84;

    static constexpr const std::string_view PLUGIN_RENDERER_SFML = "SFML";

    static constexpr const int MAX_OCTETS_READ = 4096;
    static constexpr const int TIMEOUT = 20;

    static constexpr const int MAX_PORT = 65535;

    static constexpr const unsigned int DEFAULT_FPS = 80;
    static constexpr const unsigned int DEFAULT_BITS_PER_PIXEL = 64;
    static constexpr const std::pair<const unsigned int, const unsigned int> DEFAULT_RESOLUTION {1920,
1080};
    static constexpr const std::string_view DEFAULT_NAME = "ZAPPY";

} // namespace gui
```

5.6 Gui.hpp

```
/*
** EPITECH PROJECT, 2024
** zappy_gui
** File description:
** Gui
*/

#pragma once

#include <memory>
#include <vector>

#include "GUI/Abstraction/IRenderer.hpp"
#include "GUI/Argument.hpp"

namespace gui {

    class Gui {

        public:

            enum class RendererMode {
                GAME,
                SETTINGS
            };

            explicit Gui(const Argument &args);
            ~Gui() = default;

            std::unique_ptr<IRenderer>& getRenderer() { return m_renderer; };

            void Run();

            static std::vector<std::string> getData(const std::string &data);

        private:

            std::unique_ptr<IRenderer> m_renderer;
            std::vector<std::string> m_data;
            RendererMode m_mode{RendererMode::GAME};

    }; // class Gui

} // namespace gui
```

5.7 Inventory.hpp

```
/*
** EPITECH PROJECT, 2024
** zappy_gui
** File description:
```

```
** Inventory
*/

#pragma once

#include <vector>

#include "GUI/Inventory/Resource.hpp"

namespace gui {

    class Inventory {

        public:

            Inventory(Resource food, Resource linemate, Resource deraumere, Resource sibur, Resource
mendiane, Resource phiras, Resource thystame) :
                resources({food, linemate, deraumere, sibur, mendiane, phiras, thystame}) {};
            Inventory(std::vector<Resource> cresources): resources(cresources) {};

            std::vector<Resource> resources;

    }; // class Inventory

} // namespace gui
```

5.8 Resource.hpp

```
/*
** EPITECH PROJECT, 2024
** zappy_gui
** File description:
** Resource.hpp
*/

#pragma once

#include "GUI/RunTimeException.hpp"

namespace gui {

    class Resource {

        public:

            enum class Type {
                FOOD,
                LINEMATE,
                DERAUMERE,
                SIBUR,
                MENDIANE,
                PHIRAS,
                THYSTAME,
                NONE
            };

            Resource(Type type, unsigned int quantity);

            bool operator==(const Resource &resource) const
            {
                return type == resource.type;
            }

            Type type;
            double density;
            unsigned int quantity;

    }; // class Resource

} // namespace gui
```

5.9 KeyBoard.hpp

```
/*
** EPITECH PROJECT, 2024
** zappy_gui
** File description:
** KeyBoard
*/
```

```

#pragma once

namespace gui {

    class KeyBoard {

        public:

            enum Key {
                NONE = -1, // Keep this at the beginning
                CLOSE = 0,
                KEY_LEFT = 1,
                KEY_RIGHT = 2,
                KEY_UP = 3,
                KEY_DOWN = 4,
                KEY_SPACE = 5,
                KEY_ENTER = 6,
                KEY_ESCAPE = 7,
                COUNT = 8 // corresponding to the size of the enum, keep this at the end
        };

    }; // class KeyBoard

} // namespace gui
```

5.10 Parser.hpp

```
/*
** EPITECH PROJECT, 2024
** zappy_gui
** File description:
** Parser
*/

#pragma once

#include "GUI/Argument.hpp"
#include "GUI/Gui.hpp"
#include "GUI/Inventory/Inventory.hpp"

namespace gui {

    class Parser {

        public:

            Parser() = default;
            ~Parser() = default;

            static Argument ParseArgs(int argc, char* const argv[]);

            static uint16_t ParsePort(const char* port);
            static std::string ParseMachineName(const char* machineName);

            static void processData(std::vector<std::string> data, Gui &gui);
            static Inventory parseTileContent(std::string tileContent);

            class ParserException : public std::exception
            {
                public:

                    explicit ParserException(std::string msg) : m_msg{std::move(msg)} {};
                    ~ParserException() override = default;

                    ParserException(const ParserException &) = delete;
                    ParserException &operator=(const ParserException &) = delete;
                    ParserException(const ParserException &&) = delete;
                    ParserException &operator=(const ParserException &&) = delete;

                    [[nodiscard]] const char *what() const noexcept override { return m_msg.c_str();
      };

                private:

                    std::string m_msg{0};

            }; // class ParserException

    }; // class Parser

} // namespace gui
```

5.11 PluginLoader.hpp

```
/*
** EPITECH PROJECT, 2024
** Zappy_gui
** File description:
** PluginLoader.hpp
*/

#include <unordered_map>
#include <vector>

#include "GUI/Abstraction/IRenderer.hpp"

namespace gui {

    class PluginLoader {

        public:

            using PluginCreator = std::unique_ptr<IPlugin> (*)();

            ~PluginLoader() = default;


            static PluginLoader &getInstance() {
                static PluginLoader instance;
                return instance;
            }

            template <typename T>
            std::unique_ptr<T> getPlugin(const std::string &pluginName);

            void closePlugins();

            class PluginLoaderException : public std::exception{

                public:

                    explicit PluginLoaderException(std::string msg) : m_msg(std::move(msg)) {};
                    [[nodiscard]] const char* what() const noexcept override { return m_msg.data(); };

                private:

                    std::string m_msg;

            }; // class PluginLoaderException

        private:

            PluginLoader() { loadPlugins(); };

            void loadPlugins();

            std::unordered_map<std::string, PluginCreator> m_plugins{0};
            std::vector<void*> m_handles{nullptr};

    }; // class PluginLoader

} // namespace gui
```

5.12 Protocol.hpp

```
/*
** EPITECH PROJECT, 2024
** zappy_gui
** File description:
** Protocol
*/

#pragma once

#include <unordered_map>
#include <string>

namespace gui {

    enum class ProtocolKey {
        COUNT,
        MAP_SIZE,
        TILE_CONTENT,
        MAP_CONTENT,
        TEAMS_NAME,
        PLAYER_CONNECTION,
        PLAYER_POSITION,
        PLAYER_LEVEL,
        PLAYER_INVENTORY,
        EXPULSION,
        BROADCAST,
        INCANTATION_START,
        INCANTATION_END,
        FORK,
        RESOURCES_DROP,
        RESOURCES_COLLECT,
        PLAYER_DEATH,
        EGG_LAID,
        PLAYER_EGG_CONNECTION,
        EGG_DEATH,
        TIME_UNIT_REQUEST,
        TIME_UNIT_MODIFICATION,
        END_GAME,
        MESSAGE,
        UNKNOWN,
        UNKNOWN_PARAMETER,
        EGG_MATURE,
    };

    const std::unordered_map<std::string, gui::ProtocolKey> ProtocolMap {
        {"msz", ProtocolKey::MAP_SIZE},
        {"bct", ProtocolKey::TILE_CONTENT},
        {"mct", ProtocolKey::MAP_CONTENT},
        {"tna", ProtocolKey::TEAMS_NAME},
        {"pnw", ProtocolKey::PLAYER_CONNECTION},
        {"ppo", ProtocolKey::PLAYER_POSITION},
        {"plv", ProtocolKey::PLAYER_LEVEL},
        {"pin", ProtocolKey::PLAYER_INVENTORY},
        {"pex", ProtocolKey::EXPULSION},
        {"pbc", ProtocolKey::BROADCAST},
        {"pic", ProtocolKey::INCANTATION_START},
        {"pie", ProtocolKey::INCANTATION_END},
        {"pfk", ProtocolKey::FORK},
        {"pdr", ProtocolKey::RESOURCES_DROP},
        {"pgt", ProtocolKey::RESOURCES_COLLECT},
        {"pdi", ProtocolKey::PLAYER_DEATH},
        {"enw", ProtocolKey::EGG_LAID},
        {"eht", ProtocolKey::EGG_MATURE},
        {"ebo", ProtocolKey::EGG_DEATH},
        {"pex", ProtocolKey::PLAYER_EGG_CONNECTION},
        {"sgt", ProtocolKey::TIME_UNIT_REQUEST},
        {"sst", ProtocolKey::TIME_UNIT_MODIFICATION},
        {"seg", ProtocolKey::END_GAME},
        {"smg", ProtocolKey::MESSAGE},
        {"suc", ProtocolKey::UNKNOWN},
        {"sbp", ProtocolKey::UNKNOWN_PARAMETER},
    };

    class Protocol {

        public:

            [[nodiscard]] static std::string getCommand(ProtocolKey key);
            [[nodiscard]] static ProtocolKey getKey(const std::string &command);

    }; // class Protocol

} // namespace gui
```

5.13 RunTimeException.hpp

```
/*
** EPITECH PROJECT, 2024
** Zappy_gui
** File description:
** RunTimeException.hpp
*/

#pragma once

#include <string>

namespace gui {

    class RunTimeException : public std::exception
    {
        public:

            explicit RunTimeException(std::string msg) : m_msg{std::move(msg)} {};
            ~RunTimeException() override = default;

            RunTimeException(const RunTimeException &) = delete;
            RunTimeException &operator=(const RunTimeException &) = delete;
            RunTimeException(const RunTimeException &&) = delete;
            RunTimeException &operator=(const RunTimeException &&) = delete;

            [[nodiscard]] const char *what() const noexcept override { return m_msg.c_str(); };

        private:

            std::string m_msg{0};

    }; // class RunTimeException

} // namespace gui
```

5.14 Map.hpp

```
/*
** EPITECH PROJECT, 2024
** zappy_gui
** File description:
** Map.hpp
*/

#pragma once

namespace gui {

    class Map {

        public:

            Map(unsigned int width, unsigned int height) : m_width(width), m_height(height) {};
            ~Map();

            unsigned int getWidth() const { return m_width; };
            unsigned int getHeight() const { return m_height; };

        private:

            unsigned int m_width;
            unsigned int m_height;

    }; // class Map

} // namespace gui
```

5.15 Mob.hpp

```
/*
** EPITECH PROJECT, 2024
** zappy_gui
** File description:
** Mob
*/

#pragma once

#include "GUI/Inventory/Inventory.hpp"
#include "GUI/Position.hpp"

namespace gui {

    class Mob {

        public:

            enum class Action {
                MOVE,
                FEED,
                ELEVATE,
                NONE
            };

            Action getAction() const { return m_action; };
            Inventory& getInventory() { return m_inventory; };
            Position& getPosition() { return m_position; };
            unsigned int getLevel() const { return m_level; };

            void setAction(const Action &action) { m_action = action; };

            void levelUp() { m_level++; };

        private:

            Action m_action{Action::NONE};
            Inventory m_inventory;
            Position m_position;
            unsigned int m_level{1};

    }; // class Mob

} // namespace gui
```

5.16 Position.hpp

```
/*
** EPITECH PROJECT, 2024
** zappy_gui
** File description:
** Position
*/

#pragma once

#include <utility>

namespace {

    class Position {

        public:

            Position(unsigned int x, unsigned int y) : x(x), y(y) {};
            ~Position();

            unsigned int x;
            unsigned int y;

    }; // class Position

} // namespace
```

5.17 SFML.hpp

```
/*
** EPITECH PROJECT, 2024
** zappy_gui
** File description:
** SFML
*/

#pragma once

#include <SFML/Graphics.hpp>
#include <array>

#include "GUI/Abstraction/IRenderer.hpp"
#include "GUI/SFMLClient.hpp"
#include "GUI/KeyBoard.hpp"

namespace gui {

class SFML : public IRenderer {

        public:

            SFML() = default;
            ~SFML() override = default;

            void setFPS(const unsigned int FPS) override { m_window.setFramerateLimit(FPS); };

            [[nodiscard]] std::string getPluginName() const override { return
PLUGIN_RENDERER_SFML.data(); };
            [[nodiscard]] IClient& getClient() override { return m_client; };
            [[nodiscard]] KeyBoard::Key getEvents() override;
            [[nodiscard]] bool isRunning() override { return m_window.isOpen() &&
checkConnection(m_timeoutClock); };

            void init(const std::string &name, std::pair<const unsigned int,const unsigned int>
resolution, unsigned int bitsPerPixel) override;
            void close() override { m_window.close(); getClient().disconnect(); };
            void render() override;

            static KeyBoard::Key getKeyboardEvent(const sf::Event &event);
            bool checkConnection(sf::Clock clock);

        private:

            sf::RenderWindow m_window;
            SFMLClient m_client;
            sf::Clock m_timeoutClock;

            static std::array<gui::KeyBoard::Key, sf::Keyboard::KeyCount> KEY_CODE_ARRAY;

    }; // class SFML

} // namespace sfml
```

5.18 SFMLClient.hpp

```
/*
** EPITECH PROJECT, 2024
** zappy_gui
** File description:
** Client.hpp
*/

#pragma once

#include <SFML/Network.hpp>

#include "GUI/Abstraction/IClient.hpp"
#include "GUI/Constant.hpp"

namespace gui {

    class SFMLClient : public IClient {

        public:

            ~SFMLClient() override = default;

            bool connect(uint16_t port, const std::string &machineName) override;
            void disconnect() override { m_socket.disconnect(); };

            bool sendCommand(const std::string &cmd) override;
            bool getResponse(const std::string &cmd) override;
            std::string getResponse() override;

            [[nodiscard]] bool isConnected() override;

        private:

            sf::TcpSocket m_socket{};

    }; // class Client

} // namespace gui
```

5.19 Clock.hpp

```
/*
** EPITECH PROJECT, 2024
** myLib | Clock
** File description:
** Clock.hpp
*/

/*
** @file Clock.hpp
** @brief Clock class for time management
** @namespace myLib
*/
```

```

#pragma once

#include <chrono>

#include "myLib/Clock/Time.hpp"

/*
** @brief TimePoint is a type alias for a time point which is a very long and complicated type in the
standard library
*/
using TimePoint = std::chrono::time_point<std::chrono::high_resolution_clock>;

namespace myLib {

    /*
    ** @brief Class for time management
    */
    class Clock {

        public:

            Clock() : m_start(std::chrono::high_resolution_clock::now()) {};

            ~Clock() = default;

            /*
            ** @brief Restart the clock
            */
            void restart() { m_start = std::chrono::high_resolution_clock::now(); };

            /*
            ** @brief Pause the clock
            */
            void pause();

            /*
            ** @brief Resume the clock
            */
            void resume();

            /*
            ** @brief Get the elapsed time since the last restart
            ** @return Time The elapsed time
            */
            [[nodiscard]] Time getElapsedTime() const;

        private:

            /*
            ** @property The start time
            */
            TimePoint m_start;

            /*
            ** @property The pause time
            */
            TimePoint m_pause;

            /*
            ** @property The "is in pause" boolean variable
            */
            bool m_paused{false};

    }; // Clock

} // namespace myLib
```

5.20 Time.hpp

```
/*
** EPITECH PROJECT, 2024
** myLib | Clock
** File description:
** Time.hpp
*/

/*
** @file Time.hpp
** @brief Class for time management
** @namespace myLib
*/

```

```cpp
#pragma once

namespace myLib {

    /*
    ** @class Time
    ** @brief Class used for time management
    */
    class Time {

        public:

            /*
            ** @brief Construct a new Time object
            */
            explicit Time(const double seconds) : m_seconds(seconds) {};

            /*
            ** @brief Transform the time to seconds
            ** @return int The time in seconds
            */
            [[nodiscard]] int asSeconds() const { return static_cast<int>(m_seconds); };

            /*
            ** @brief Transform the time to milliseconds
            ** @return int The time in milliseconds
            */
            [[nodiscard]] int asMilliseconds() const { return static_cast<int>(m_seconds * 1000); }

            /*
            ** @brief Transform the time to microseconds
            ** @return int The time in microseconds
            */
            [[nodiscard]] int asMicroseconds() const { return static_cast<int>(m_seconds * 1000000); };

        private:

            /*
            ** @property The time in seconds
            */
            double m_seconds{0.0F};

    }; // Time

} // namespace myLib
```

5.21 Random.hpp

```cpp
/*
** EPITECH PROJECT, 2024
** myLib
** File description:
** Random.hpp
*/

/*
** @file Random.hpp
** @brief Class for random number generation
** @namespace myLib
*/

#pragma once

#include <random>

namespace myLib {

    /*
    ** @class Random
    ** @brief Class for random number generation
    */
    class Random {

        public:

            /*
            ** @brief Generate a random integer between min and max
            ** @param min The minimum value
            ** @param max The maximum value
            ** @return int The random integer
            */
            static int randomInt(int min, int max);
            static int randomInt() { return randomInt(-1000, 1000); };

            /*
            ** @param min The minimum value
            ** @param max The maximum value
            ** @return float The random float
            */
            static float randomFloat(float min, float max);
            static float randomFloat() { return randomFloat(-1.0f, 1.0f); };

    }; // class Random

} // namespace myLib
```

Index