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** , **END** }

Public Member Functions

- **Gui** (const Argument &args)
- std::unique_ptr< IRenderer > & **getRenderer** ()
- void **Run** ()
- void **initMap** (const std::pair< unsigned, unsigned > &size)
- void **initEgg** (const unsigned int &eggId, const int &playerId, const std::pair< unsigned int, unsigned int > &pos)
- void **matureEgg** (const unsigned int &eggId)
- void **eggDeath** (const unsigned int &eggId)
- Map & **getMap** ()
- int **getFrequency** () const
- std::vector< std::string > & **getTeamNames** ()
- std::vector< Player > & **getPlayers** ()
- RendererMode **getMode** () const
- void **addTeamName** (const std::string &teamName)
- void **addPlayer** (const Player &player)
- void **setMap** (const Map &map)
- void **setFrequency** (int freq)
- void **setMode** (RendererMode mode)

Static Public Member Functions

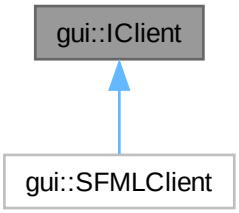
- static std::vector< std::string > **getData** (const std::string &data)

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)
- void **setQuantity** (Resource::Type type, unsigned int quantity)

Public Attributes

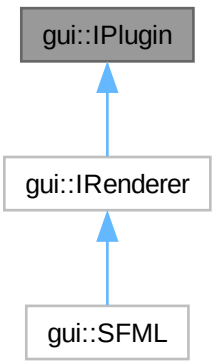
- std::vector< Resource > **resources**

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

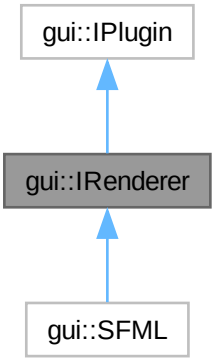
- virtual std::string **getPluginName** () const =0

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

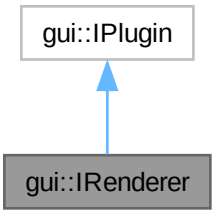
- include/GUI/Abstraction/IPlugin.hpp

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** (Map &map)=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, const std::vector< std::vector< Tile > > &tiles)
- unsigned int **getWidth** () const
- unsigned int **getHeight** () const
- void **setWidth** (unsigned int width)
- void **setHeight** (unsigned int height)
- void **addTile** (const Tile &tile)
- std::vector< std::vector< Tile > > & **getTiles** ()
- void **countResources** ()

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

- include/GUI/Map/Map.hpp

4.10 gui::Parser Class Reference

Classes

- class ParserException

Static Public Member Functions

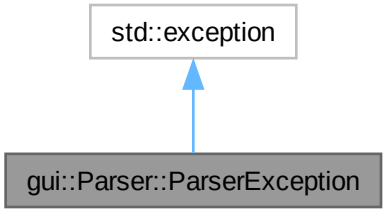
- static Argument **getOptions** (int argc, char ∗const argv[ ], const std::string &optString)
- static uint16_t **parsePort** (const char ∗port)
- static std::string **parseMachineName** (const char ∗machineName)
- static void **processData** (const std::vector< std::string > &data, Gui &gui)
- static Tile **parseTileContent** (std::string &tileContent)
- static Player::Orientation **parseOrientation** (const std::string &orientation)

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

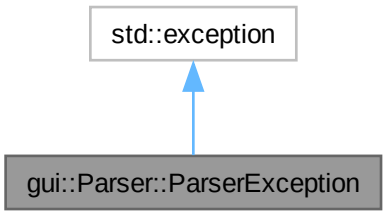
- include/GUI/Parser.hpp

4.11 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

4.12 gui::Player Class Reference

Public Types

- enum class **Action** { **MOVE** , **FEED** , **ELEVATE** , **NONE** }
- enum **Orientation** { **NORTH** = 1 , **EAST** = 2 , **SOUTH** = 3 , **WEST** = 4 }

Public Member Functions

- Action **getAction** () const
- Orientation **getOrientation** () const
- Inventory & **getInventory** ()
- Position & **getPosition** ()
- unsigned int **getLevel** () const
- unsigned int **getId** () const
- std::string **getTeamName** () const
- void **setAction** (const Action action)
- void **setOrientation** (const Orientation orientation)
- void **setId** (const unsigned int id)
- void **setTeamName** (const std::string &teamName)
- void **setLevel** (const unsigned int level)
- void **levelUp** ()

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

- include/GUI/Player.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

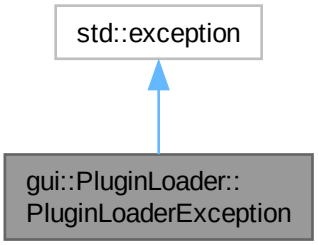
- static PluginLoader & **getInstance** ()

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

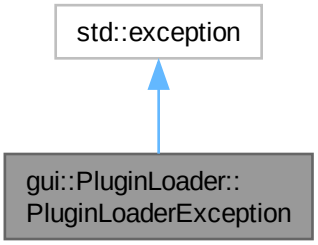
- include/GUI/PluginLoader.hpp

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::Position Class Reference

Public Member Functions

- **Position** (unsigned int cx, unsigned int cy)

Public Attributes

- unsigned int **x**
- unsigned int **y**

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

- include/GUI/Position.hpp

4.16 gui::Protocol Class Reference

Static Public Member Functions

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

Static Public Attributes

- static const std::unordered_map< std::string, std::function< void(gui::Gui &, std::string)> > **ProtocolMap**

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

- include/GUI/Protocol.hpp

4.17 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

4.18 gui::Resource Class Reference

Public Types

- enum **Type** {
  FOOD = 0 , **LINEMATE** = 1 , **DERAUMERE** = 2 , **SIBUR** = 3 ,
  MENDIANE = 4 , **PHIRAS** = 5 , **THYSTAME** = 6 , **NONE** = 7 }

Public Member Functions

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

Public Attributes

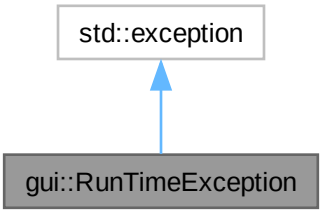
- Type **type**
- double **density**
- unsigned int **quantity**

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

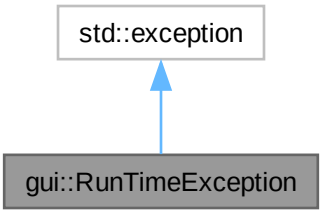
- include/GUI/Inventory/Resource.hpp

4.19 gui::RunTimeException Class Reference

Inheritance diagram for gui::RunTimeException:



Collaboration diagram for gui::RunTimeException:



Public Member Functions

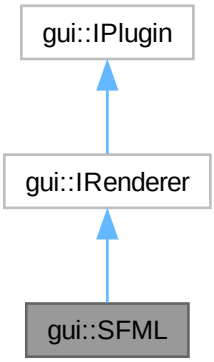
- **RunTimeException** (std::string msg)
- **RunTimeException** (const RunTimeException &)=delete
- RunTimeException & **operator=** (const RunTimeException &)=delete
- **RunTimeException** (const RunTimeException &&)=delete
- RunTimeException & **operator=** (const RunTimeException &&)=delete
- const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

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

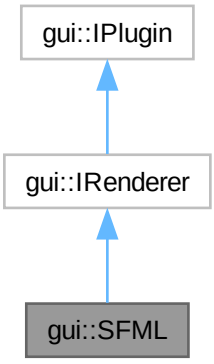
- include/GUI/RunTimeException.hpp

4.20 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 (Map &map) override
- bool **checkConnection** (sf::Clock clock)
- std::vector< std::pair< sf::Sprite, std::string > > & **getSprites** ()
- std::vector< std::pair< sf::Texture, std::string > > & **getTextures** ()
- void **addSprite** (const sf::Sprite &sprite, const std::string &name)
- void **addTexture** (const sf::Texture &texture, const std::string &name)

- 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** (Map &map)=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)

4.20.1 Member Function Documentation

4.20.1.1 close()

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

Implements gui::IRenderer.

4.20.1.2 getClient()

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

Implements gui::IRenderer.

4.20.1.3 getEvents()

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

Implements gui::IRenderer.

4.20.1.4 getPluginName()

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

Implements gui::IPlugin.

4.20.1.5 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.

4.20.1.6 isRunning()

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

Implements gui::IRenderer.

4.20.1.7 render()

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

Implements gui::IRenderer.

4.20.1.8 setFPS()

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

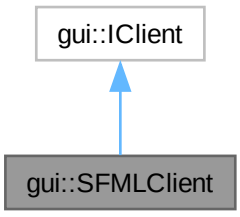
Implements gui::IRenderer.

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

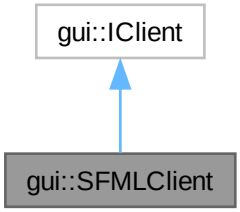
- lib/shared/Renderer/SFML/include/GUI/SFML.hpp

4.21 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

4.21.1 Member Function Documentation

4.21.1.1 connect()

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

Implements gui::IClient.

4.21.1.2 disconnect()

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

Implements gui::IClient.

4.21.1.3 getResponse() [1/2]

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

Implements gui::IClient.

4.21.1.4 getResponse() [2/2]

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

Implements gui::IClient.

4.21.1.5 isConnected()

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

Implements gui::IClient.

4.21.1.6 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

4.22 gui::Tile Class Reference

Public Member Functions

- **Tile** (Inventory inventory, const Position &position)
- **Tile** (const Tile &tile)=default
- Inventory **getInventory** () const
- void **setInventory** (Inventory inventory)
- Position **getPosition** () const
- void **setPosition** (Position position)

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

- include/GUI/Map/Tile.hpp

4.23 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

```
00001 /*
00002 ** EPITECH PROJECT, 2024
00003 ** zappy_gui
00004 ** File description:
00005 ** IClient
00006 */
00007
00008 #pragma once
00009
00010 #include <string>
00011 #include <cstdint>
00012
00013 namespace gui {
00014
00015     class IClient {
00016
00017         public:
00018
00019             virtual ~IClient() = default;
00020
00021             virtual bool connect(uint16_t port, const std::string& machineName) = 0;
00022             virtual void disconnect() = 0;
00023             virtual bool sendCommand(const std::string& cmd) = 0;
00024             virtual bool getResponse(const std::string& cmd) = 0;
00025             virtual std::string getResponse() = 0;
00026             virtual bool isConnected() = 0;
00027
00028     }; // class IClient
00029
00030 } // namespace gui
```

5.2 IPlugin.hpp

```
00001 /*
00002 ** EPITECH PROJECT, 2024
00003 ** zappy_gui
00004 ** File description:
00005 ** IPlugin
00006 */
00007
00008 #pragma once
00009
00010 #include <string>
00011
00012 namespace gui {
00013
00014     class IPlugin {
00015
00016     public:
00017
00018         virtual ~IPlugin() = default;
00019
00020         [[nodiscard]] virtual std::string getPluginName() const = 0;
00021
00022     }; // class IPlugin
00023
00024 } // namespace gui
```

5.3 IRenderer.hpp

```
00001 /*
00002 ** EPITECH PROJECT, 2024
00003 ** zappy_gui
00004 ** File description:
00005 ** IRenderer
00006 */
00007
00008 #pragma once
00009
00010 #include "GUI/Abstraction/IPlugin.hpp"
00011 #include "GUI/Abstraction/IClient.hpp"
00012 #include "GUI/KeyBoard.hpp"
00013 #include "GUI/Map/Map.hpp"
00014
00015 namespace gui {
00016
00017     class IRenderer : public IPlugin {
00018
00019         public:
00020
00021             virtual void setFPS(unsigned int FPS) = 0;
00022
00023             [[nodiscard]] virtual IClient& getClient() = 0;
00024             [[nodiscard]] virtual bool isRunning() = 0;
00025
00026             virtual void init(const std::string &name, std::pair<const unsigned int,const unsigned
    int> resolution, unsigned int bitsPerPixel) = 0;
00027             virtual void render(Map &map) = 0;
00028             virtual KeyBoard::Key getEvents() = 0;
00029             virtual void close() = 0;
00030
00031     }; // class IRenderer
00032
00033 } // namespace gui
```

5.4 Argument.hpp

```
00001 /*
00002 ** EPITECH PROJECT, 2024
00003 ** zappy_gui
00004 ** File description:
00005 ** Argument
00006 */
00007
00008 #pragma once
00009
00010 #include <string>
00011 #include <cstdint>
00012
00013 namespace gui {
00014
00015     class Argument {
00016
00017         public:
00018
00019             Argument(const uint16_t p, std::string h) : port(p), hostName(std::move(h)) {};
00020             ~Argument() = default;
00021
00022             const uint16_t port;
00023             const std::string hostName;
00024
00025     }; // class Argument
00026
00027 } // namespace gui
```

5.5 Constant.hpp

```
00001 /*
00002 ** EPITECH PROJECT, 2024
00003 ** Zappy | GUI
00004 ** File description:
00005 ** Constant.hpp
00006 */
00007
00008 /*
00009 ** @file Constant.hpp
00010 ** @brief Constants for the Zappy GUI
```

```
00011 ** @namespace gui
00012 */
00013
00014 #pragma once
00015
00016 #include <string_view>
00017
00018 namespace gui {
00019
00020     static constexpr const int EPITECH_EXIT_SUCCESS = 0;
00021     static constexpr const int EPITECH_EXIT_ERROR = 84;
00022
00023     static constexpr const std::string_view PLUGIN_RENDERER_SFML = "SFML";
00024
00025     static constexpr const int MAX_OCTETS_READ = 4096;
00026     static constexpr const int TIMEOUT = 20;
00027
00028     static constexpr const int MAX_PORT = 65535;
00029
00030     static constexpr const unsigned int DEFAULT_FPS = 80;
00031     static constexpr const unsigned int DEFAULT_BITS_PER_PIXEL = 64;
00032     static constexpr const std::pair<const unsigned int, const unsigned int> DEFAULT_RESOLUTION {1920,
      1080};
00033     static constexpr const std::string_view DEFAULT_NAME = "ZAPPY";
00034     static constexpr const int MAX_MAP_SIZE = 30;
00035
00036 } // namespace gui
```

5.6 Gui.hpp

```
00001 /*
00002 ** EPITECH PROJECT, 2024
00003 ** zappy_gui
00004 ** File description:
00005 ** Gui
00006 */
00007
00008 #pragma once
00009
00010 #include <memory>
00011 #include <vector>
00012
00013 #include "GUI/Abstraction/IRenderer.hpp"
00014 #include "GUI/Argument.hpp"
00015 #include "GUI/Map/Map.hpp"
00016 #include "GUI/Player.hpp"
00017
00018 namespace gui {
00019
00020     class Gui {
00021
00022         public:
00023
00024             enum class RendererMode {
00025                 GAME,
00026                 SETTINGS,
00027                 END
00028             };
00029
00030             explicit Gui(const Argument &args);
00031             ~Gui() = default;
00032
00033             std::unique_ptr<IRenderer>& getRenderer() { return m_renderer; };
00034
00035             void Run();
00036
00037             void initMap(const std::pair<unsigned, unsigned> &size);
00038             void initEgg(const unsigned int &eggId, const int &playerId, const std::pair<unsigned int,
      unsigned int> &pos);
00039             void matureEgg(const unsigned int &eggId);
00040             void eggDeath(const unsigned int &eggId);
00041
00042             [[nodiscard]] static std::vector<std::string> getData(const std::string &data);
00043
00044             [[nodiscard]] Map& getMap() { return m_map; };
00045             [[nodiscard]] int getFrequency() const { return m_frequency; };
00046             [[nodiscard]] std::vector<std::string>& getTeamNames() { return m_teamNames; };
00047             [[nodiscard]] std::vector<Player>& getPlayers() { return m_players; };
00048             [[nodiscard]] RendererMode getMode() const { return m_mode; };
00049
00050             void addTeamName(const std::string &teamName) { for (auto &team : m_teamNames) if (team ==
      teamName) return; m_teamNames.push_back(teamName); };
00051             void addPlayer(const Player &player) { m_players.push_back(player); };
```

```
00052             void setMap(const Map &map) { m_map = map; };
00053             void setFrequency(int freq) { m_frequency = freq; };
00054             void setMode(RendererMode mode) { m_mode = mode; };
00055 
00056         private:
00057 
00058             std::vector<std::string> m_teamNames;
00059             std::vector<Player> m_players;
00060             std::unique_ptr<IRenderer> m_renderer;
00061             std::vector<std::string> m_data;
00062             RendererMode m_mode{RendererMode::GAME};
00063             std::pair<int, int> m_mapSize{0, 0};
00064             Map m_map{30, 30, {}};
00065             int m_frequency{0};
00066 
00067     }; // class Gui
00068 
00069 } // namespace gui
```

5.7 Inventory.hpp

```
00001 /*
00002 ** EPITECH PROJECT, 2024
00003 ** zappy_gui
00004 ** File description:
00005 ** Inventory
00006 */
00007 
00008 #pragma once
00009 
00010 #include <vector>
00011 
00012 #include "GUI/Inventory/Resource.hpp"
00013 
00014 namespace gui {
00015 
00016     class Inventory {
00017 
00018         public:
00019 
00020             Inventory() = default;
00021 
00022             Inventory(Resource food, Resource linemate, Resource deraumere, Resource sibur, Resource
      mendiane, Resource phiras, Resource thystame) :
00023                 resources({food, linemate, deraumere, sibur, mendiane, phiras, thystame}) {};
00024             explicit Inventory(std::vector<Resource> cresources): resources(std::move(cresources)) {};
00025 
00026             void setQuantity(Resource::Type type, unsigned int quantity) {
00027                 for (auto &resource : resources) {
00028                     if (resource.type == type) {
00029                         resource.quantity = quantity;
00030                         return;
00031                     }
00032                 }
00033             };
00034 
00035             std::vector<Resource> resources;
00036 
00037     }; // class Inventory
00038 
00039 } // namespace gui
```

5.8 Resource.hpp

```
00001 /*
00002 ** EPITECH PROJECT, 2024
00003 ** zappy_gui
00004 ** File description:
00005 ** Resource.hpp
00006 */
00007 
00008 #pragma once
00009 
00010 #include "GUI/RunTimeException.hpp"
00011 
00012 namespace gui {
00013 
00014     class Resource {
00015 
```

```
00016         public:
00017
00018             enum Type {
00019                 FOOD = 0,
00020                 LINEMATE = 1,
00021                 DERAUMERE = 2,
00022                 SIBUR = 3,
00023                 MENDIANE = 4,
00024                 PHIRAS = 5,
00025                 THYSTAME = 6,
00026                 NONE = 7
00027             };
00028
00029             Resource(Type type, unsigned int quantity);
00030
00031             bool operator==(const Resource &resource) const
00032             {
00033                 return type == resource.type;
00034             }
00035
00036             Type type;
00037             double density;
00038             unsigned int quantity;
00039
00040     }; // class Resource
00041
00042 } // namespace gui
```

5.9 KeyBoard.hpp

```
00001 /*
00002 ** EPITECH PROJECT, 2024
00003 ** zappy_gui
00004 ** File description:
00005 ** KeyBoard
00006 */
00007
00008 #pragma once
00009
00010 namespace gui {
00011
00012     class KeyBoard {
00013
00014         public:
00015
00016             enum Key {
00017                 NONE = -1, // Keep this at the beginning
00018                 CLOSE = 0,
00019                 KEY_LEFT = 1,
00020                 KEY_RIGHT = 2,
00021                 KEY_UP = 3,
00022                 KEY_DOWN = 4,
00023                 KEY_SPACE = 5,
00024                 KEY_ENTER = 6,
00025                 KEY_ESCAPE = 7,
00026                 COUNT = 8 // corresponding to the size of the enum, keep this at the end
00027         };
00028
00029     }; // class KeyBoard
00030
00031 } // namespace gui
```

5.10 Map.hpp

```
00001 /*
00002 ** EPITECH PROJECT, 2024
00003 ** zappy_gui
00004 ** File description:
00005 ** Map.hpp
00006 */
00007
00008 #pragma once
00009
00010 #include <array>
00011 #include <iostream>
00012
00013 #include "GUI/Constant.hpp"
00014 #include "GUI/Map/Tile.hpp"
00015
```

```
00016 namespace gui {
00017
00018     class Map {
00019
00020         public:
00021
00022             Map(unsigned int width, unsigned int height, const std::vector<std::vector<Tile»& tiles) :
      m_width(width), m_height(height), m_tiles(tiles) {};
00023             ~Map() = default;
00024
00025             [[nodiscard]] unsigned int getWidth() const { return m_width; };
00026             [[nodiscard]] unsigned int getHeight() const { return m_height; };
00027             void setWidth(unsigned int width) { m_width = width; };
00028             void setHeight(unsigned int height) { m_height = height; };
00029             void addTile(const Tile& tile) { m_tiles.at(tile.getPosition().x).at(tile.getPosition().y)
      = tile; };
00030
00031             [[nodiscard]] std::vector<std::vector<Tile»& getTiles() { return m_tiles; };
00032
00033             // DEBUG - TO REMOVE
00034             void countResources() {
00035                 for (auto &row : m_tiles) {
00036                     for (auto &tile : row) {
00037                         std::cout « "Tile: " « tile.getPosition().x « " " « tile.getPosition().y «
      '\n';
00038                         for (auto &resource : tile.getInventory().resources) {
00039                             std::cout « "Resource: " « resource.quantity « " " « resource.density «
      '\n';
00040                         }
00041                     }
00042                 }
00043             };
00044
00045         private:
00046
00047             unsigned int m_width;
00048             unsigned int m_height;
00049
00050             std::vector<std::vector<Tile» m_tiles;
00051
00052     }; // class Map
00053
00054 } // namespace gui
```

5.11 Tile.hpp

```
00001 /*
00002 ** EPITECH PROJECT, 2024
00003 ** zappy_gui
00004 ** File description:
00005 ** Tile
00006 */
00007
00008 #pragma once
00009
00010 #include "GUI/Inventory/Inventory.hpp"
00011 #include "GUI/Position.hpp"
00012
00013 namespace gui {
00014
00015     class Tile {
00016
00017         public:
00018
00019             Tile(Inventory inventory, const Position& position) : m_inventory(std::move(inventory)),
      m_position(position) {};
00020             Tile(const Tile& tile) = default;
00021             ~Tile() = default;
00022
00023             Tile() : m_inventory(Inventory({Resource::Type::FOOD, 0}, {Resource::Type::LINEMATE, 0},
00024                                            {Resource::Type::DERAUMERE, 0}, {Resource::Type::SIBUR, 0},
00025                                            {Resource::Type::MENDIANE, 0}, {Resource::Type::PHIRAS, 0},
00026                                            {Resource::Type::THYSTAME, 0})),
00027                      m_position(Position(0, 0)){};
00028
00029             [[nodiscard]] Inventory getInventory() const { return m_inventory; };
00030             void setInventory(Inventory inventory) { m_inventory = std::move(inventory); };
00031             [[nodiscard]] Position getPosition() const { return m_position; };
00032             void setPosition(Position position) { m_position = position; };
00033
00034         private:
00035
00036             Inventory m_inventory;
```

```
00037             Position m_position;
00038
00039     }; // class Tile
00040
00041 } // namespace gui
```

5.12 Parser.hpp

```
00001 /*
00002 ** EPITECH PROJECT, 2024
00003 ** zappy_gui
00004 ** File description:
00005 ** Parser
00006 */
00007
00008 #pragma once
00009
00010 #include "GUI/Gui.hpp"
00011
00012 namespace gui {
00013
00014     class Parser {
00015
00016         public:
00017
00018             static Argument getOptions(int argc, char* const argv[], const std::string &optString);
00019
00020             static uint16_t parsePort(const char* port);
00021             static std::string parseMachineName(const char* machineName);
00022
00023             static void processData(const std::vector<std::string>& data, Gui &gui);
00024             static Tile parseTileContent(std::string &tileContent);
00025             static Player::Orientation parseOrientation(const std::string &orientation);
00026
00027             class ParserException : public std::exception
00028             {
00029                 public:
00030
00031                     explicit ParserException(std::string msg) : m_msg{std::move(msg)} {};
00032                     ~ParserException() override = default;
00033
00034                     ParserException(const ParserException &) = delete;
00035                     ParserException &operator=(const ParserException &) = delete;
00036                     ParserException(const ParserException &&) = delete;
00037                     ParserException &operator=(const ParserException &&) = delete;
00038
00039                     [[nodiscard]] const char *what() const noexcept override { return m_msg.c_str();
    };
00040
00041                 private:
00042
00043                     std::string m_msg{0};
00044
00045             }; // class ParserException
00046
00047     }; // class Parser
00048
00049 } // namespace gui
```

5.13 Player.hpp

```
00001 /*
00002 ** EPITECH PROJECT, 2024
00003 ** zappy_gui
00004 ** File description:
00005 ** Player.hpp
00006 */
00007
00008 #pragma once
00009
00010 #include "GUI/Inventory/Inventory.hpp"
00011 #include "GUI/Position.hpp"
00012
00013 namespace gui {
00014
00015     class Player {
00016
00017         public:
00018
```

```
00019             enum class Action {
00020                 MOVE,
00021                 FEED,
00022                 ELEVATE,
00023                 NONE
00024             };
00025
00026             enum Orientation {
00027                 NORTH = 1,
00028                 EAST = 2,
00029                 SOUTH = 3,
00030                 WEST = 4
00031             };
00032
00033             Player() = default;
00034             ~Player() = default;
00035
00036             [[nodiscard]] Action getAction() const { return m_action; };
00037             [[nodiscard]] Orientation getOrientation() const { return m_orientation; };
00038             [[nodiscard]] Inventory& getInventory() { return m_inventory; };
00039             [[nodiscard]] Position& getPosition() { return m_position; };
00040             [[nodiscard]] unsigned int getLevel() const { return m_level; };
00041             [[nodiscard]] unsigned int getId() const { return m_id; };
00042             [[nodiscard]] std::string getTeamName() const { return m_teamName; };
00043
00044             void setAction(const Action action) { m_action = action; };
00045             void setOrientation(const Orientation orientation) { m_orientation = orientation; };
00046             void setId(const unsigned int id) { m_id = id; };
00047             void setTeamName(const std::string &teamName) { m_teamName = teamName; };
00048             void setLevel(const unsigned int level) { m_level = level; };
00049
00050             void levelUp() { m_level++; };
00051
00052         private:
00053
00054             Action m_action{Action::NONE};
00055             Inventory m_inventory;
00056             Position m_position;
00057             Orientation m_orientation{Orientation::NORTH};
00058             std::string m_teamName{""};
00059             unsigned int m_id{0};
00060             unsigned int m_level{1};
00061             bool isAlive{true};
00062
00063     }; // class Player
00064
00065 } // namespace gui
```

5.14 PluginLoader.hpp

```
00001 /*
00002 ** EPITECH PROJECT, 2024
00003 ** Zappy_gui
00004 ** File description:
00005 ** PluginLoader.hpp
00006 */
00007
00008 #include <unordered_map>
00009 #include <vector>
00010
00011 #include "GUI/Abstraction/IRenderer.hpp"
00012
00013 namespace gui {
00014
00015     class PluginLoader {
00016
00017         public:
00018
00019             using PluginCreator = std::unique_ptr<IPlugin> (*)();
00020
00021             ~PluginLoader() = default;
00022
00023
00024             static PluginLoader &getInstance() {
00025                 static PluginLoader instance;
00026                 return instance;
00027             }
00028
00029             template <typename T>
00030             std::unique_ptr<T> getPlugin(const std::string &pluginName);
00031
00032             void closePlugins();
00033
```

```
00034             class PluginLoaderException : public std::exception{
00035
00036                 public:
00037
00038                     explicit PluginLoaderException(std::string msg) : m_msg(std::move(msg)) {};
00039                     [[nodiscard]] const char* what() const noexcept override { return m_msg.data(); };
00040
00041                 private:
00042
00043                     std::string m_msg;
00044
00045             }; // class PluginLoaderException
00046
00047         private:
00048
00049             PluginLoader() { loadPlugins(); };
00050
00051             void loadPlugins();
00052
00053             std::unordered_map<std::string, PluginCreator> m_plugins{0};
00054             std::vector<void*> m_handles{nullptr};
00055
00056     }; // class PluginLoader
00057
00058 } // namespace gui
```

5.15 Position.hpp

```
00001 /*
00002 ** EPITECH PROJECT, 2024
00003 ** zappy_gui
00004 ** File description:
00005 ** Position
00006 */
00007
00008 #pragma once
00009
00010 namespace gui {
00011
00012     class Position {
00013
00014         public:
00015
00016             Position() = default;
00017
00018             Position(unsigned int cx, unsigned int cy) : x(cx), y(cy) {};
00019             ~Position() = default;
00020
00021             unsigned int x;
00022             unsigned int y;
00023
00024     }; // class Position
00025
00026 } // namespace
```

5.16 Protocol.hpp

```
00001 /*
00002 ** EPITECH PROJECT, 2024
00003 ** zappy_gui
00004 ** File description:
00005 ** Protocol
00006 */
00007
00008 #pragma once
00009
00010 #include <unordered_map>
00011 #include <functional>
00012
00013 #include "GUI/Parser.hpp"
00014
00015 namespace gui {
00016
00017     class Protocol {
00018
00019         public:
00020
00021             static const std::unordered_map<std::string, std::function<void(gui::Gui&, std::string)»
      ProtocolMap;
```

```
00022
00023             [[nodiscard]] static std::vector<std::string> parseCommand(const std::string &data) {
00024                 std::vector<std::string> dataVector;
00025                 std::string tmp;
00026                 for (const auto &c : data) {
00027                     if (c == '\n' || c == ' ') {
00028                         dataVector.push_back(tmp);
00029                         tmp.clear();
00030                     } else {
00031                         tmp += c;
00032                     }
00033                 }
00034                 return dataVector;
00035             };
00036
00037     }; // class Protocol
00038
00039 } // namespace gui
```

5.17 RunTimeException.hpp

```
00001 /*
00002 ** EPITECH PROJECT, 2024
00003 ** Zappy_gui
00004 ** File description:
00005 ** RunTimeException.hpp
00006 */
00007
00008 #pragma once
00009
00010 #include <string>
00011
00012 namespace gui {
00013
00014     class RunTimeException : public std::exception
00015     {
00016         public:
00017
00018             explicit RunTimeException(std::string msg) : m_msg{std::move(msg)} {};
00019             ~RunTimeException() override = default;
00020
00021             RunTimeException(const RunTimeException &) = delete;
00022             RunTimeException &operator=(const RunTimeException &) = delete;
00023             RunTimeException(const RunTimeException &&) = delete;
00024             RunTimeException &operator=(const RunTimeException &&) = delete;
00025
00026             [[nodiscard]] const char *what() const noexcept override { return m_msg.c_str(); };
00027
00028         private:
00029
00030             std::string m_msg{0};
00031
00032     }; // class RunTimeException
00033
00034 } // namespace gui
```

5.18 SFML.hpp

```
00001 /*
00002 ** EPITECH PROJECT, 2024
00003 ** zappy_gui
00004 ** File description:
00005 ** SFML
00006 */
00007
00008 #pragma once
00009
00010 #include <SFML/Graphics.hpp>
00011 #include <array>
00012
00013 #include "GUI/Abstraction/IRenderer.hpp"
00014 #include "GUI/SFMLClient.hpp"
00015 #include "GUI/KeyBoard.hpp"
00016 #include "GUI/Map/Map.hpp"
00017
00018 namespace gui {
00019
00020 class SFML : public IRenderer {
00021
```

```
00022         public:
00023
00024             SFML() = default;
00025             ~SFML() override = default;
00026
00027             void setFPS(const unsigned int FPS) override { m_window.setFramerateLimit(FPS); };
00028
00029             [[nodiscard]] std::string getPluginName() const override { return
      PLUGIN_RENDERER_SFML.data(); };
00030             [[nodiscard]] IClient& getClient() override { return m_client; };
00031             [[nodiscard]] KeyBoard::Key getEvents() override;
00032             [[nodiscard]] bool isRunning() override { return m_window.isOpen() &&
      checkConnection(m_timeoutClock); };
00033
00034             void init(const std::string &name, std::pair<const unsigned int,const unsigned int>
      resolution, unsigned int bitsPerPixel) override;
00035             void close() override { m_window.close(); getClient().disconnect(); };
00036             void render(Map &map) override;
00037
00038             [[nodiscard]] static KeyBoard::Key getKeyboardEvent(const sf::Event &event);
00039             [[nodiscard]] bool checkConnection(sf::Clock clock);
00040
00041             [[nodiscard]] std::vector<std::pair<sf::Sprite, std::string» &getSprites() { return
      m_sprites; };
00042             [[nodiscard]] std::vector<std::pair<sf::Texture, std::string» &getTextures() { return
      m_textures; };
00043
00044             void addSprite(const sf::Sprite &sprite, const std::string &name) {
      m_sprites.push_back({sprite, name}); };
00045             void addTexture(const sf::Texture &texture, const std::string &name) {
      m_textures.push_back({texture, name}); };
00046
00047         private:
00048
00049             sf::RenderWindow m_window;
00050             SFMLClient m_client;
00051             sf::Clock m_timeoutClock;
00052
00053             std::vector<std::pair<sf::Sprite, std::string» m_sprites;
00054             std::vector<std::pair<sf::Texture, std::string» m_textures;
00055
00056             static std::array<gui::KeyBoard::Key, sf::Keyboard::KeyCount> KEY_CODE_ARRAY;
00057
00058     }; // class SFML
00059
00060 } // namespace sfml
```

5.19 SFMLClient.hpp

```
00001 /*
00002 ** EPITECH PROJECT, 2024
00003 ** zappy_gui
00004 ** File description:
00005 ** Client.hpp
00006 */
00007
00008 #pragma once
00009
00010 #include <SFML/Network.hpp>
00011
00012 #include "GUI/Abstraction/IClient.hpp"
00013 #include "GUI/Constant.hpp"
00014
00015 namespace gui {
00016
00017     class SFMLClient : public IClient {
00018
00019         public:
00020
00021             ~SFMLClient() override = default;
00022
00023             [[nodiscard]] bool connect(uint16_t port, const std::string &machineName) override;
00024             void disconnect() override { m_socket.disconnect(); };
00025
00026             [[nodiscard]] bool sendCommand(const std::string &cmd) override;
00027             [[nodiscard]] bool getResponse(const std::string &cmd) override;
00028             [[nodiscard]] std::string getResponse() override;
00029
00030             [[nodiscard]] bool isConnected() override;
00031
00032         private:
00033
00034             sf::TcpSocket m_socket;
```

```
00035
00036     }; // class Client
00037
00038 } // namespace gui
```

5.20 Clock.hpp

```
00001 /*
00002 ** EPITECH PROJECT, 2024
00003 ** myLib | Clock
00004 ** File description:
00005 ** Clock.hpp
00006 */
00007
00008 /*
00009 ** @file Clock.hpp
00010 ** @brief Clock class for time management
00011 ** @namespace myLib
00012 */
00013
00014 #pragma once
00015
00016 #include <chrono>
00017
00018 #include "myLib/Clock/Time.hpp"
00019
00020 /*
00021 ** @brief TimePoint is a type alias for a time point which is a very long and complicated type in the
      standard library
00022 */
00023 using TimePoint = std::chrono::time_point<std::chrono::high_resolution_clock>;
00024
00025 namespace myLib {
00026
00027     /*
00028     ** @brief Class for time management
00029     */
00030     class Clock {
00031
00032         public:
00033
00034             Clock() : m_start(std::chrono::high_resolution_clock::now()) {};
00035
00036             ~Clock() = default;
00037
00038             /*
00039             ** @brief Restart the clock
00040             */
00041             void restart() { m_start = std::chrono::high_resolution_clock::now(); };
00042
00043             /*
00044             ** @brief Pause the clock
00045             */
00046             void pause();
00047
00048             /*
00049             ** @brief Resume the clock
00050             */
00051             void resume();
00052
00053             /*
00054             ** @brief Get the elapsed time since the last restart
00055             ** @return Time The elapsed time
00056             */
00057             [[nodiscard]] Time getElapsedTime() const;
00058
00059         private:
00060
00061             /*
00062             ** @property The start time
00063             */
00064             TimePoint m_start;
00065
00066             /*
00067             ** @property The pause time
00068             */
00069             TimePoint m_pause;
00070
00071             /*
00072             ** @property The "is in pause" boolean variable
00073             */
00074             bool m_paused{false};
00075
```

```
00076     }; // Clock
00077
00078 } // namespace myLib
```

5.21 Time.hpp

```
00001 /*
00002 ** EPITECH PROJECT, 2024
00003 ** myLib | Clock
00004 ** File description:
00005 ** Time.hpp
00006 */
00007
00008 /*
00009 ** @file Time.hpp
00010 ** @brief Class for time management
00011 ** @namespace myLib
00012 */
00013
00014 #pragma once
00015
00016 namespace myLib {
00017
00018     /*
00019     ** @class Time
00020     ** @brief Class used for time management
00021     */
00022     class Time {
00023
00024         public:
00025
00026             /*
00027             ** @brief Construct a new Time object
00028             */
00029             explicit Time(const double seconds) : m_seconds(seconds) {};
00030
00031             /*
00032             ** @brief Transform the time to seconds
00033             ** @return int The time in seconds
00034             */
00035             [[nodiscard]] int asSeconds() const { return static_cast<int>(m_seconds); };
00036
00037             /*
00038             ** @brief Transform the time to milliseconds
00039             ** @return int The time in milliseconds
00040             */
00041             [[nodiscard]] int asMilliseconds() const { return static_cast<int>(m_seconds * 1000); }
00042
00043             /*
00044             ** @brief Transform the time to microseconds
00045             ** @return int The time in microseconds
00046             */
00047             [[nodiscard]] int asMicroseconds() const { return static_cast<int>(m_seconds * 1000000);
    };
00048
00049         private:
00050
00051             /*
00052             ** @property The time in seconds
00053             */
00054             double m_seconds{0.0F};
00055
00056     }; // Time
00057
00058 } // namespace myLib
```

5.22 Random.hpp

```
00001 /*
00002 ** EPITECH PROJECT, 2024
00003 ** myLib
00004 ** File description:
00005 ** Random.hpp
00006 */
00007
00008 /*
00009 ** @file Random.hpp
00010 ** @brief Class for random number generation
00011 ** @namespace myLib
```

```
00012 */
00013
00014 #pragma once
00015
00016 #include <random>
00017
00018 namespace myLib {
00019
00020     /*
00021     ** @class Random
00022     ** @brief Class for random number generation
00023     */
00024     class Random {
00025
00026         public:
00027
00028             /*
00029             ** @brief Generate a random integer between min and max
00030             ** @param min The minimum value
00031             ** @param max The maximum value
00032             ** @return int The random integer
00033             */
00034             static int randomInt(int min, int max);
00035             static int randomInt() { return randomInt(-1000, 1000); };
00036
00037             /*
00038             ** @param min The minimum value
00039             ** @param max The maximum value
00040             ** @return float The random float
00041             */
00042             static float randomFloat(float min, float max);
00043             static float randomFloat() { return randomFloat(-1.0f, 1.0f); };
00044
00045     }; // class Random
00046
00047 } // namespace myLib
```

Index