

R-Type

Generated by Doxygen 1.9.2

1 Hierarchical Index	1
1.1 Class Hierarchy	1
2 Class Index	3
2.1 Class List	3
3 File Index	5
3.1 File List	5
4 Class Documentation	7
4.1 network::client Class Reference	7
4.2 Core Class Reference	7
4.3 DyLib Class Reference	7
4.3.1 Constructor & Destructor Documentation	8
4.3.1.1 DyLib() [1/3]	8
4.3.1.2 DyLib() [2/3]	8
4.3.1.3 DyLib() [3/3]	9
4.3.2 Member Function Documentation	9
4.3.2.1 close()	9
4.3.2.2 getFunction()	9
4.3.2.3 getVariable()	9
4.3.2.4 open()	11
4.4 DyLib::exception Class Reference	11
4.4.1 Detailed Description	12
4.5 ecs::exception Class Reference	12
4.6 Game Class Reference	13
4.6.1 Member Function Documentation	13
4.6.1.1 parseData()	13
4.6.1.2 removeKilled()	13
4.7 DyLib::handle_error Class Reference	15
4.7.1 Detailed Description	15
4.8 network::header Struct Reference	15
4.9 Inputbox Class Reference	16
4.9.1 Member Function Documentation	16
4.9.1.1 draw()	17
4.9.1.2 getInputbox()	17
4.9.1.3 getSelected()	17
4.9.1.4 getText() [1/2]	17
4.9.1.5 getText() [2/2]	18
4.9.1.6 inputLogic()	18
4.9.1.7 setFont()	18
4.9.1.8 setLimit()	18
4.9.1.9 setPosition()	19

4.9.1.10 setSelected()	19
4.9.1.11 setString()	19
4.10 Lobby Class Reference	19
4.10.1 Member Function Documentation	20
4.10.1.1 parseData()	20
4.11 Menu Class Reference	20
4.12 Data::MissingAsset Class Reference	21
4.13 ecs::module Class Reference	22
4.13.1 Member Function Documentation	22
4.13.1.1 get_label()	22
4.13.1.2 get_system()	22
4.13.1.3 spawn()	22
4.14 network::packet Struct Reference	23
4.15 ecs::component::position Struct Reference	23
4.16 Profile Class Reference	23
4.16.1 Member Function Documentation	24
4.16.1.1 draw()	24
4.16.1.2 setPosition()	24
4.17 ecs::registry Class Reference	24
4.17.1 Member Function Documentation	25
4.17.1.1 add_system()	25
4.17.1.2 destroy()	25
4.17.1.3 emplace_component()	25
4.17.1.4 get_components() [1/2]	26
4.17.1.5 get_components() [2/2]	26
4.17.1.6 get_entities()	27
4.17.1.7 has_component()	27
4.17.1.8 insert_component()	27
4.17.1.9 kill_entity()	28
4.17.1.10 register_component()	28
4.17.1.11 remove_component()	28
4.17.1.12 run_systems()	28
4.17.1.13 spawn_entity()	29
4.18 network::server Class Reference	29
4.18.1 Member Function Documentation	29
4.18.1.1 allow_new_connections()	29
4.18.1.2 get_clients()	30
4.18.1.3 receive()	30
4.18.1.4 send()	30
4.19 network::socket Class Reference	30
4.19.1 Member Function Documentation	31
4.19.1.1 connect()	31

4.19.1.2 disconnect()	31
4.19.1.3 get_packets()	31
4.19.1.4 get_remote_ip()	32
4.19.1.5 get_remote_port()	32
4.19.1.6 pop_packets()	32
4.19.1.7 send()	32
4.20 ecs::sparse_array< T > Class Template Reference	33
4.20.1 Member Function Documentation	33
4.20.1.1 emplace_at()	33
4.20.1.2 erase()	34
4.20.1.3 get_index()	34
4.20.1.4 insert_at() [1/2]	34
4.20.1.5 insert_at() [2/2]	36
4.21 Spritesheet Class Reference	36
4.21.1 Member Function Documentation	37
4.21.1.1 draw()	38
4.21.1.2 getCurrentImage()	38
4.21.1.3 getNbSprite()	38
4.21.1.4 getPosition()	38
4.21.1.5 getRect()	39
4.21.1.6 getScale()	39
4.21.1.7 getSize()	39
4.21.1.8 getSpace()	39
4.21.1.9 getTexture()	40
4.21.1.10 mouselsInSprite()	40
4.21.1.11 playImage()	40
4.21.1.12 setNbSprite()	40
4.21.1.13 setPosition()	42
4.21.1.14 setRect()	42
4.21.1.15 setRectSize()	42
4.21.1.16 setScale()	43
4.21.1.17 setSizeImage()	43
4.21.1.18 setSpace()	43
4.21.1.19 setTexture()	43
4.22 DyLib::symbol_error Class Reference	44
4.22.1 Detailed Description	44
4.23 Textbox Class Reference	44
4.23.1 Member Function Documentation	45
4.23.1.1 draw()	45
4.23.1.2 getPosition()	46
4.23.1.3 getString()	46
4.23.1.4 setColor()	46

4.23.1.5 setFont()	46
4.23.1.6 setOutlineColor()	47
4.23.1.7 setPosition()	47
4.23.1.8 setSize()	47
4.23.1.9 setString()	47
4.24 thread_pool Class Reference	49
4.24.1 Member Function Documentation	49
4.24.1.1 add()	49
4.24.1.2 release()	49
4.24.1.3 remove()	50
4.25 vec2float Struct Reference	50
5 File Documentation	51
5.1 include/DyLib.hpp File Reference	51
5.1.1 Detailed Description	51
5.2 DyLib.hpp	52
5.3 Components.hpp	54
5.4 Entity.hpp	54
5.5 Exception.hpp	55
5.6 Module.hpp	55
5.7 Registry.hpp	55
5.8 SparseArray.hpp	57
5.9 Systems.hpp	58
5.10 Client.hpp	60
5.11 Header.hpp	60
5.12 Packet.hpp	60
5.13 Server.hpp	60
5.14 Socket.hpp	62
5.15 Type.hpp	64
5.16 Random.hpp	64
5.17 ThreadPool.hpp	64
5.18 Tools.hpp	65
5.19 TypeChecker.hpp	65
5.20 Core.hpp	65
5.21 Data.hpp	66
5.22 Game.hpp	67
5.23 Inputbox.hpp	68
5.24 Lobby.hpp	68
5.25 Menu.hpp	69
5.26 Profile.hpp	70
5.27 Spritesheet.hpp	70
5.28 SpriteSize.hpp	71

5.29 Status.hpp	71
5.30 Textbox.hpp	71
Index	73

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

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

network::client	7
Core	7
sf::Drawable	
Inputbox	16
Profile	23
Spritesheet	36
Textbox	44
DyLib	7
std::exception	
Data::MissingAsset	21
DyLib::exception	11
DyLib::handle_error	15
DyLib::symbol_error	44
ecs::exception	12
Game	13
network::header	15
Lobby	19
Menu	20
ecs::module	22
network::packet	23
ecs::component::position	23
ecs::registry	24
network::server	29
network::socket	30
ecs::sparse_array< T >	33
thread_pool	49
vec2float	50

Chapter 2

Class Index

2.1 Class List

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

network::client	7
Core	7
DyLib	7
DyLib::exception	11
ecs::exception	12
Game	13
DyLib::handle_error	15
network::header	15
Inputbox	16
Lobby	19
Menu	20
Data::MissingAsset	21
ecs::module	22
network::packet	23
ecs::component::position	23
Profile	23
ecs::registry	24
network::server	29
network::socket	30
ecs::sparse_array< T >	33
Spritesheet	36
DyLib::symbol_error	44
Textbox	44
thread_pool	49
vec2float	50

Chapter 3

File Index

3.1 File List

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

include/DyLib.hpp	
Cross-platform Dynamic Library Loader	51
include/Random.hpp	64
include/ThreadPool.hpp	64
include/Tools.hpp	65
include/TypeChecker.hpp	65
include/ECS/Components.hpp	54
include/ECS/Entity.hpp	54
include/ECS/Exception.hpp	55
include/ECS/Module.hpp	55
include/ECS/Registry.hpp	55
include/ECS/SparseArray.hpp	57
include/ECS/Systems.hpp	58
include/Network/Client.hpp	60
include/Network/Header.hpp	60
include/Network/Packet.hpp	60
include/Network/Server.hpp	60
include/Network/Socket.hpp	62
include/Network/Type.hpp	64
include/UI/Core.hpp	65
include/UI/Data.hpp	66
include/UI/Game.hpp	67
include/UI/Inputbox.hpp	68
include/UI/Lobby.hpp	68
include/UI/Menu.hpp	69
include/UI/Profile.hpp	70
include/UI/Spritesheet.hpp	70
include/UI/SpriteSize.hpp	71
include/UI/Status.hpp	71
include/UI/Textbox.hpp	71

Chapter 4

Class Documentation

4.1 network::client Class Reference

Public Member Functions

- **client** (asio::ip::udp::endpoint &&endp)
- std::string **get_ip** () const
- unsigned short **get_port** ()
- const asio::ip::udp::endpoint & **get_endpoint** () const noexcept
- std::vector< [packet](#) > & **get_packets** () noexcept
- std::vector< [packet](#) > **pop_packets** () noexcept
- bool & **alive** ()

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

- include/Network/Client.hpp

4.2 Core Class Reference

Public Member Functions

- void **loop** ()
this function is the main loop that calls menu, lobby and game

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

- include/UI/Core.hpp

4.3 DyLib Class Reference

Classes

- class [exception](#)
- class [handle_error](#)
- class [symbol_error](#)

Public Member Functions

- constexpr [DyLib](#) () noexcept=default
- **DyLib** (const [DyLib](#) &)=delete
- [DyLib](#) & **operator=** (const [DyLib](#) &)=delete
- [DyLib](#) ([DyLib](#) &&other) noexcept
- [DyLib](#) & **operator=** ([DyLib](#) &&other) noexcept
- [DyLib](#) (const char *path)
- **DyLib** (const std::string &path)
- **DyLib** (std::string path, const char *ext)
- void [open](#) (const char *path)
- void **open** (const std::string &path)
- void **open** (std::string path, const char *ext)
- template<typename T >
std::function< T > [getFunction](#) (const char *name) const
- template<typename T >
std::function< T > **getFunction** (const std::string &name) const
- template<typename T >
T & [getVariable](#) (const char *name) const
- template<typename T >
T & **getVariable** (const std::string &name) const
- void [close](#) () noexcept

Static Public Attributes

- static constexpr auto **extension** = ".so"

4.3.1 Constructor & Destructor Documentation

4.3.1.1 [DyLib\(\)](#) [1/3]

```
constexpr DyLib::DyLib ( ) [constexpr], [default], [noexcept]
```

Creates a dynamic library object

4.3.1.2 [DyLib\(\)](#) [2/3]

```
DyLib::DyLib (
    DyLib && other ) [inline], [noexcept]
```

Move constructor : move a dynamic library instance to build this object

Parameters

<i>other</i>	ref on rvalue of the other DyLib (use std::move)
--------------	--

4.3.1.3 DyLib() [3/3]

```
DyLib::DyLib (  
    const char * path ) [inline], [explicit]
```

Creates a dynamic library instance

Parameters

<i>path</i>	path to the dynamic library to load
<i>ext</i>	use DyLib::extension to specify the os extension (optional parameter)

4.3.2 Member Function Documentation

4.3.2.1 close()

```
void DyLib::close ( ) [inline], [noexcept]
```

Close the dynamic library currently loaded in the object. This function will be automatically called by the class destructor

4.3.2.2 getFunction()

```
template<typename T >  
std::function< T > DyLib::getFunction (  
    const char * name ) const [inline]
```

Get a function from the dynamic library currently loaded in the object

Parameters

<i>T</i>	the template argument must be the function prototype. it must be the same pattern as the template of std::function
<i>name</i>	symbol name of the function to get from the dynamic library

Returns

std::function<T> that contains the function

4.3.2.3 getVariable()

```
template<typename T >  
T & DyLib::getVariable (  
    const char * name ) const [inline]
```

Get a global variable from the dynamic library currently loaded in the object

Parameters

<i>T</i>	type of the global variable
<i>name</i>	name of the global variable to get from the dynamic library

Returns

global variable of type <T>

4.3.2.4 open()

```
void DyLib::open (
    const char * path ) [inline]
```

Load a dynamic library into the object. If a dynamic library was already opened, it will be unload and replaced

Parameters

<i>path</i>	path to the dynamic library to load
<i>ext</i>	use DyLib::extension to specify the os extension (optional parameter)

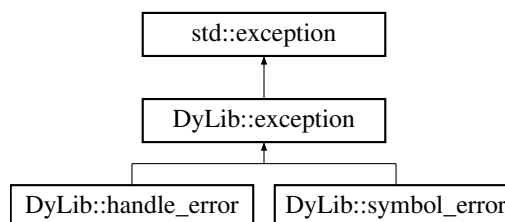
The documentation for this class was generated from the following file:

- [include/DyLib.hpp](#)

4.4 DyLib::exception Class Reference

```
#include <DyLib.hpp>
```

Inheritance diagram for DyLib::exception:



Public Member Functions

- **exception** (std::string &&message)
- const char * **what** () const noexcept override

Protected Attributes

- `const std::string m_error`

4.4.1 Detailed Description

This exception is thrown when the [DyLib](#) class encountered an error.

Returns

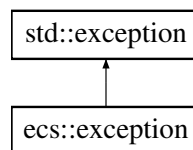
error message by calling `what()` member function

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

- `include/DyLib.hpp`

4.5 `ecs::exception` Class Reference

Inheritance diagram for `ecs::exception`:



Public Member Functions

- **exception** (`std::string err`)
- `template<typename ... T>`
exception (`T &&...err`)
- `const char * what ()` `const` `noexcept` override

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

- `include/ECS/Exception.hpp`

4.6 Game Class Reference

Public Member Functions

- **Game** (sf::RenderWindow &window, [network::socket](#) &socket, uint8_t &status)
- void **gameScreen** ()
this function is the game loop that calls all utils fonctions
- void **getServState** ()
Get the Serv State object.
- void [parseData](#) (const std::string &str)
this function is used to parse date received from serv updating game entities status
- void [removeKilled](#) (std::vector< size_t > id)
function that remove dead ids in m_display
- void **checkInput** ()
check user inputs and send them to serv
- void **gameEvents** ()
this function is managing all events during the game
- void **eventsButton** ()
this function is animating buttons when player is dead
- void **eventsPressedButton** ()
this function is sending request to serv when player click on a button
- void **animSprites** ()
this function is animating sprites
- void **displayGame** ()
this function is drawing the game on the screen

4.6.1 Member Function Documentation

4.6.1.1 [parseData\(\)](#)

```
void Game::parseData (
    const std::string & str )
```

this function is used to parse date received from serv updating game entities status

Parameters

<i>data</i>	const std::string & (received from serv)
-------------	--

4.6.1.2 [removeKilled\(\)](#)

```
void Game::removeKilled (
    std::vector< size_t > id )
```

function that remove dead ids in m_display

Parameters

<i>id</i>	std::vector<size_t>
-----------	---------------------

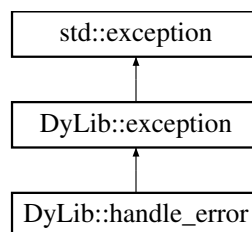
The documentation for this class was generated from the following file:

- include/UI/Game.hpp

4.7 DyLib::handle_error Class Reference

```
#include <DyLib.hpp>
```

Inheritance diagram for DyLib::handle_error:



Public Member Functions

- **handle_error** (std::string &&message)

Additional Inherited Members

4.7.1 Detailed Description

This exception is thrown when the library failed to load or the library encountered symbol resolution issues

Parameters

<i>message</i>	error message
----------------	---------------

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

- include/DyLib.hpp

4.8 network::header Struct Reference

Public Attributes

- uint32_t **magicValue**

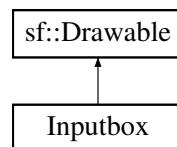
- uint8_t **type**
- uint32_t **size**

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

- include/Network/Header.hpp

4.9 Inputbox Class Reference

Inheritance diagram for Inputbox:



Public Member Functions

- **Inputbox** (int size=15, sf::Color color=sf::Color::White, bool select=false, size_t lim=0)
- void **inputLogic** (char c)
this function is adding char to the end of a sf::String
- void **setFont** (const sf::Font &font)
Set the Font object.
- void **setPosition** (sf::Vector2f pos)
Set the Position object.
- void **setLimit** (size_t lim)
Set the Limit object (set the maximum size of the String object)
- void **setSelected** (bool sel)
Set the Selected object.
- void **setString** (sf::String str)
Set the String object.
- sf::String & **getText** ()
Get the Text object.
- const sf::String & **getText** () const
Get the Text object.
- bool & **getSelected** ()
Get the Selected object.
- sf::Text & **getInputbox** ()
Get the Inputbox object.
- void **updateString** ()
this function is updating the sf::Text with the sf::String
- void **draw** (sf::RenderTarget &target, sf::RenderStates states) const override
Drawing the class on the window.

4.9.1 Member Function Documentation

4.9.1.1 draw()

```
void Inputbox::draw (
    sf::RenderTarget & target,
    sf::RenderStates states ) const [override]
```

Drawing the class on the window.

Parameters

<i>target</i>	sf::RenderTarget window
<i>states</i>	sf::RenderStates Define the states used for drawing to a RenderTarget

4.9.1.2 getInputbox()

```
sf::Text & Inputbox::getInputbox ( )
```

Get the [Inputbox](#) object.

Returns

sf::Text&

4.9.1.3 getSelected()

```
bool & Inputbox::getSelected ( )
```

Get the Selected object.

Returns

true

false

4.9.1.4 getText() [1/2]

```
sf::String & Inputbox::getText ( )
```

Get the Text object.

Returns

sf::String&

4.9.1.5 getText() [2/2]

```
const sf::String & Inputbox::getText ( ) const
```

Get the Text object.

Returns

const sf::String&

4.9.1.6 inputLogic()

```
void Inputbox::inputLogic (
    char c )
```

this function is adding char to the end of a sf::String

Parameters

<i>c</i>	char to add at the end of the string
----------	--------------------------------------

4.9.1.7 setFont()

```
void Inputbox::setFont (
    const sf::Font & font )
```

Set the Font object.

Parameters

<i>font</i>	sf::Font to set
-------------	-----------------

4.9.1.8 setLimit()

```
void Inputbox::setLimit (
    size_t lim )
```

Set the Limit object (set the maximum size of the String object)

Parameters

<i>lim</i>	size_t to set
------------	---------------

4.9.1.9 setPosition()

```
void Inputbox::setPosition (
    sf::Vector2f pos )
```

Set the Position object.

Parameters

<i>pos</i>	sf::Vector2f to set
------------	---------------------

4.9.1.10 setSelected()

```
void Inputbox::setSelected (
    bool sel )
```

Set the Selected object.

Parameters

<i>sel</i>	bool to set
------------	-------------

4.9.1.11 setString()

```
void Inputbox::setString (
    sf::String str )
```

Set the String object.

Parameters

<i>str</i>	sf::String to set
------------	-------------------

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

- include/UI/Inputbox.hpp

4.10 Lobby Class Reference

Public Member Functions

- **Lobby** (sf::RenderWindow &window, [network::socket](#) &socket, uint8_t &status)

- void **initLobby** ()
this function init all lobby requirements to start from 0
- void **endLobby** ()
function called at the end of lobby to stop music, reset the cursor...
- void **lobbyScreen** ()
this function is the lobby loop that calls utils functions
- void **eventsLobby** ()
this function is managing all lobby events
- void **eventsReady** ()
this function is managing ready button animations
- void **eventsDisconnect** ()
this function is managing disconnect button animations
- void **eventsPressButton** ()
this function is managing all buttons events
- void **reqLobby** ()
this function is communicating with the serv getting infos of all other clients connected to the serv: ready or not getting info when game start
- void **parseData** (const std::string &data)
this function is used to parse date received from serv setting a vector of profile (profile = client)
- void **displayLobby** ()
function that draw the lobby on the window

4.10.1 Member Function Documentation

4.10.1.1 parseData()

```
void Lobby::parseData (
    const std::string & data )
```

this function is used to parse date received from serv setting a vector of profile (profile = client)

Parameters

<i>data</i>	const std::string & (received from serv)
-------------	--

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

- include/UI/Lobby.hpp

4.11 Menu Class Reference

Public Member Functions

- **Menu** (sf::RenderWindow &window, [network::socket](#) &socket, uint8_t &status)

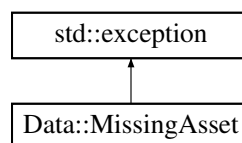
- void **initMenu** ()
this function init all menu requirements to start from 0
- void **loginScreen** ()
this function is the menu loop that calls all utils functions
- void **eventsInputbox** ()
this function is updating cursor texture also select the inputbox selected with cursor
- void **checkInputboxSelected** ()
this function select the next inputbox when user press tab
- void **eventsMenu** ()
this function is managing all menu events
- void **eventsLogin** ()
this function update login button texture
- void **eventsUser** ()
this function update user button texture
- void **tryLogin** ()
this function send to serv a login request set an error if connection failed go to username stage if connection is setup
- void **tryUsername** ()
this function send username entered by user to the serv set an error if username is already used go to lobby if username is good
- void **displayMenu** ()
this function draw the menu on window
- void **drawUser** ()
this function draw the username stage and errors

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

- include/UI/Menu.hpp

4.12 Data::MissingAsset Class Reference

Inheritance diagram for Data::MissingAsset:



Public Member Functions

- **MissingAsset** (const std::string &path)
- const char * **what** () const noexcept override

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

- include/UI/Data.hpp

4.13 ecs::module Class Reference

Public Member Functions

- virtual ecs::entity `spawn` (`registry` &)=0
- virtual std::function< void(`registry` &)> `get_system` ()=0
- virtual std::string `get_label` ()=0

4.13.1 Member Function Documentation

4.13.1.1 `get_label()`

```
virtual std::string ecs::module::get_label ( ) [pure virtual]
```

Returns

the entity's label

4.13.1.2 `get_system()`

```
virtual std::function< void(registry &)> ecs::module::get_system ( ) [pure virtual]
```

get the system to manage the entity

Parameters

<i>registry</i>	the registry to use with the system
-----------------	-------------------------------------

Returns

the system function as a std::function<void(`registry` &)>

4.13.1.3 `spawn()`

```
virtual ecs::entity ecs::module::spawn (  
    registry & ) [pure virtual]
```

spawn a new entity

Parameters

<i>registry</i>	the registry to put the new entity on
-----------------	---------------------------------------

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

- include/ECS/Module.hpp

4.14 network::packet Struct Reference

Public Attributes

- uint8_t **type**
- std::string **data**

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

- include/Network/Packet.hpp

4.15 ecs::component::position Struct Reference

Public Attributes

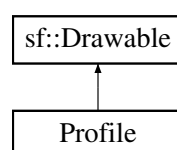
- int **x**
- int **y**
- int **vx**
- int **vy**
- int **cur**

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

- include/ECS/Components.hpp

4.16 Profile Class Reference

Inheritance diagram for Profile:



Public Member Functions

- **Profile** (std::string username, bool rdy, size_t nb)
- void [setPosition](#) (sf::Vector2f pos)
Set the Position object set position of all objects.
- void [draw](#) (sf::RenderTarget &target, sf::RenderStates states) const override
Drawing the class on the window.

4.16.1 Member Function Documentation

4.16.1.1 draw()

```
void Profile::draw (
    sf::RenderTarget & target,
    sf::RenderStates states ) const [override]
```

Drawing the class on the window.

Parameters

<i>target</i>	sf::RenderTarget window
<i>states</i>	sf::RenderStates Define the states used for drawing to a RenderTarget

4.16.1.2 setPosition()

```
void Profile::setPosition (
    sf::Vector2f pos )
```

Set the Position object set position of all objects.

Parameters

<i>pos</i>	sf::Vector2f pos
------------	------------------

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

- include/UI/Profile.hpp

4.17 ecs::registry Class Reference

Public Member Functions

- template<typename T >
[sparse_array](#)< T > & [register_component](#) ()

- `template<class T >`
`bool has_component ()`
- `template<typename T >`
`sparse_array< T > & get_components ()`
- `template<typename T >`
`const sparse_array< T > & get_components () const`
- `template<typename T >`
`sparse_array< T >::reference_type insert_component (const entity &ent, T &&comp)`
- `template<typename T , typename ... Params>`
`sparse_array< T >::reference_type emplace_component (const entity &ent, Params &&...params)`
- `template<typename T >`
`void remove_component (const entity &ent)`
- `entity spawn_entity ()`
- `void kill_entity (entity e)`
- `const std::vector< entity > & get_entities ()`
- `template<typename Function >`
`void add_system (Function &&f)`
- `void run_systems ()`
- `void destroy ()`

4.17.1 Member Function Documentation

4.17.1.1 add_system()

```
template<typename Function >
void ecs::registry::add_system (
    Function && f ) [inline]
```

This function is used to add a system

Parameters

<i>f</i>	function representing a system
----------	--------------------------------

4.17.1.2 destroy()

```
void ecs::registry::destroy ( ) [inline]
```

This function is used to destroy all data from the registry

4.17.1.3 emplace_component()

```
template<typename T , typename ... Params>
sparse_array< T >::reference_type ecs::registry::emplace_component (
    const entity & ent,
    Params &&... params ) [inline]
```

emplace a component at a position

Parameters

<i>T</i>	type of the component
<i>ent</i>	position to emplace the component
<i>params</i>	parameters to forward to build a component

Returns

[sparse_array](#) of components

4.17.1.4 get_components() [1/2]

```
template<typename T >
sparse\_array< T > & ecs::registry::get_components ( ) [inline]
```

get a [sparse_array](#) of components from the registry

Parameters

<i>T</i>	type of the component to get
----------	------------------------------

Returns

[sparse_array](#) of components

4.17.1.5 get_components() [2/2]

```
template<typename T >
const sparse\_array< T > & ecs::registry::get_components ( ) const [inline]
```

get a [sparse_array](#) of components from the registry

Parameters

<i>T</i>	type of the component to get
----------	------------------------------

Returns

[sparse_array](#) of components

4.17.1.6 get_entities()

```
const std::vector< entity > & ecs::registry::get_entities ( ) [inline]
```

get entities from the registry

Returns

the entity list

4.17.1.7 has_component()

```
template<class T >  
bool ecs::registry::has_component ( ) [inline]
```

check if the registry contains a component

Parameters

<i>T</i>	type of the component to check
----------	--------------------------------

Returns

true if the component is registered, false otherwise

4.17.1.8 insert_component()

```
template<typename T >  
sparse_array< T >::reference_type ecs::registry::insert_component (   
    const entity & ent,  
    T && comp ) [inline]
```

insert a component at a position

Parameters

<i>T</i>	type of the component
<i>ent</i>	position to insert the component
<i>comp</i>	component to be insert

Returns

[sparse_array](#) of components

4.17.1.9 kill_entity()

```
void ecs::registry::kill_entity (
    entity e ) [inline]
```

This function is used to kill an entity

Parameters

<i>e</i>	entity to kill
----------	----------------

4.17.1.10 register_component()

```
template<typename T >
sparse_array< T > & ecs::registry::register_component ( ) [inline]
```

register a new component to the registry

Parameters

<i>T</i>	type of the new component
----------	---------------------------

Returns

the new component as `sparse_array<T>`

4.17.1.11 remove_component()

```
template<typename T >
void ecs::registry::remove_component (
    const entity & ent ) [inline]
```

remove a component from a `sparse_array` of component

Parameters

<i>T</i>	type of the component
<i>ent</i>	position to remove the component

4.17.1.12 run_systems()

```
void ecs::registry::run_systems ( ) [inline]
```

This function is used to run all loaded systems

4.17.1.13 spawn_entity()

```
entity ecs::registry::spawn_entity ( ) [inline]
```

This function is used to spawn a new entity

Returns

a new entity

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

- include/ECS/Registry.hpp

4.18 network::server Class Reference

Public Member Functions

- **server** (unsigned short port, size_t max_clients, uint8_t login_handshake, uint8_t logout_handshake, uint8_t alive_handshake)
- **server** ([server](#) &)=delete
- [server](#) & **operator=** ([server](#) &)=delete
- **server** ([server](#) &&)=delete
- [server](#) & **operator=** ([server](#) &&)=delete
- void [send](#) (const [client](#) &c, uint8_t type, const std::string &buffer="", unsigned short repeat=1)
- void [receive](#) ()
- std::vector< [client](#) > & [get_clients](#) () noexcept
- void [allow_new_connections](#) (bool value)

4.18.1 Member Function Documentation

4.18.1.1 allow_new_connections()

```
void network::server::allow_new_connections (
    bool value ) [inline]
```

this function is used to lock or unlock new connections

Parameters

<i>value</i>	choice
--------------	--------

4.18.1.2 get_clients()

```
std::vector< client > & network::server::get_clients ( ) [inline], [noexcept]
```

this function is used to get the clients

Returns

client list

4.18.1.3 receive()

```
void network::server::receive ( ) [inline]
```

this function is used to receive pending packets from clients

4.18.1.4 send()

```
void network::server::send (
    const client & c,
    uint8_t type,
    const std::string & buffer = "",
    unsigned short repeat = 1 ) [inline]
```

this function is used to send packets to a client

Parameters

<i>c</i>	client to send the packet
<i>type</i>	type of the request to the client as a enum
<i>buffer</i>	data to send to client
<i>repeat</i>	number of times the packet will be send

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

- include/Network/Server.hpp

4.19 network::socket Class Reference

Public Member Functions

- **socket** (unsigned short port, uint8_t login_handshake, uint8_t logout_handshake, uint8_t alive_handshake)

- **socket** ([socket](#) &)=delete
- [socket](#) & **operator=** ([socket](#) &)=delete
- **socket** ([socket](#) &&)=delete
- [socket](#) & **operator=** ([socket](#) &&)=delete
- void [connect](#) (std::string ip, unsigned short port, long login_timeout=3)
- void [disconnect](#) () noexcept
- void [send](#) (uint8_t type, const std::string &buffer="", unsigned short repeat=1)
- std::string [get_remote_ip](#) () const
- unsigned short [get_remote_port](#) () const
- std::vector< [packet](#) > [get_packets](#) () noexcept
- std::vector< [packet](#) > [pop_packets](#) () noexcept

4.19.1 Member Function Documentation

4.19.1.1 connect()

```
void network::socket::connect (
    std::string ip,
    unsigned short port,
    long login_timeout = 3 ) [inline]
```

this function is used to send a request to the server to connect

Parameters

<i>ip</i>	the host ip
<i>port</i>	is the port for the ip
<i>login_timeout</i>	login timeout in sec, the function will throw after that delay

4.19.1.2 disconnect()

```
void network::socket::disconnect ( ) [inline], [noexcept]
```

this function is used to disconnect from the host and clear the buffer

4.19.1.3 get_packets()

```
std::vector< packet > network::socket::get_packets ( ) [inline], [noexcept]
```

this function is used to get the pending packets

Returns

the pending packets contained in the socket

4.19.1.4 get_remote_ip()

```
std::string network::socket::get_remote_ip ( ) const [inline]
```

this function is used to get the remote ip

Returns

the remote address as a std::string

4.19.1.5 get_remote_port()

```
unsigned short network::socket::get_remote_port ( ) const [inline]
```

this function is used to get the remote port

Returns

the remote port as a unsigned short

4.19.1.6 pop_packets()

```
std::vector< packet > network::socket::pop_packets ( ) [inline], [noexcept]
```

this function is used to move the pending packets and get them

Returns

the pending packets contained in the socket and empty them on the socket class

4.19.1.7 send()

```
void network::socket::send (
    uint8_t type,
    const std::string & buffer = "",
    unsigned short repeat = 1 ) [inline]
```

this function is used to send packets to the server

Parameters

<i>type</i>	type of the packet
<i>buffer</i>	data to send to the server
<i>repeat</i>	number of times to send the packets

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

- `include/Network/Socket.hpp`

4.20 `ecs::sparse_array< T >` Class Template Reference

Public Types

- using **value_type** = `std::optional< T >`
- using **reference_type** = `value_type &`
- using **const_reference_type** = `const value_type &`
- using **container_t** = `std::vector< value_type >`
- using **size_type** = `typename container_t::size_type`
- using **iterator** = `typename container_t::iterator`
- using **const_iterator** = `typename container_t::const_iterator`

Public Member Functions

- **sparse_array** (`sparse_array` const &)=default
- **sparse_array** (`sparse_array` &&) noexcept=default
- **sparse_array** & **operator=** (`sparse_array` const &)=default
- **sparse_array** & **operator=** (`sparse_array` &&) noexcept=default
- iterator **begin** ()
- const_iterator **begin** () const
- const_iterator **cbegin** () const
- iterator **end** ()
- const_iterator **end** () const
- const_iterator **cend** () const
- size_type **size** () const
- reference_type **operator[]** (size_t idx)
- const_reference_type **operator[]** (size_t idx) const
- reference_type **insert_at** (size_type pos, const T &comp)
- reference_type **insert_at** (size_type pos, T &&comp)
- template<class ... Params>
reference_type **emplace_at** (size_type pos, Params &&...params)
- void **erase** (size_type pos)
- size_type **get_index** (value_type const &val) const

4.20.1 Member Function Documentation

4.20.1.1 `emplace_at()`

```
template<typename T >
template<class ... Params>
reference_type ecs::sparse_array< T >::emplace_at (
    size_type pos,
    Params &&... params ) [inline]
```

emplace the component at the position pos

Parameters

<i>pos</i>	position to emplace the component
<i>params</i>	parameters to forward to build the component

Returns

the component just emplaced

4.20.1.2 erase()

```
template<typename T >
void ecs::sparse_array< T >::erase (
    size_type pos ) [inline]
```

erase a component at a position

Parameters

<i>pos</i>	position where to erease a component
------------	--------------------------------------

4.20.1.3 get_index()

```
template<typename T >
size_type ecs::sparse_array< T >::get_index (
    value_type const & val ) const [inline]
```

get the index from a value

Parameters

<i>val</i>	value
------------	-------

Returns

the index matching the value

4.20.1.4 insert_at() [1/2]

```
template<typename T >
reference_type ecs::sparse_array< T >::insert_at (
```

```
size_type pos,  
const T & comp ) [inline]
```

insert the component at the position `pos`

Parameters

<i>pos</i>	position to insert the component
<i>comp</i>	component to insert

Returns

the component just inserted

4.20.1.5 insert_at() [2/2]

```
template<typename T >
reference_type ecs::sparse_array< T >::insert_at (
    size_type pos,
    T && comp ) [inline]
```

insert the component at the position pos

Parameters

<i>pos</i>	position to insert the component
<i>comp</i>	component to insert

Returns

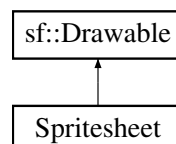
the component just inserted

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

- include/ECS/SparseArray.hpp

4.21 Spritesheet Class Reference

Inheritance diagram for Spritesheet:



Public Member Functions

- **Spritesheet** (sf::Texture &text, sf::Vector2f pos={0, 0}, size_t nb=1, sf::Vector2f dim={1920, 1080}, sf::IntRect rect={0, 0, 1920, 1080}, sf::Vector2f scale={1, 1}, int space=0)
- void **setNbSprite** (size_t nb)
Set the Nb Sprite object.
- void **setSizeImage** (sf::Vector2f size)
Set the Size Image object.
- void **setTexture** (const sf::Texture &text)
Set the Texture object.
- void **setPosition** (sf::Vector2f pos)
Set the Position object.
- void **setRectSize** (sf::IntRect rect)
Set the Rect object (only width and height)
- void **setRect** (sf::IntRect rect)
Set the Rect object.
- void **setSpace** (int space)
Set the Space object.
- void **setScale** (sf::Vector2f scale)
Set the Scale object.
- size_t & **getCurrentImage** ()
Get the Current Image object.
- size_t & **getNbSprite** ()
Get the Nb Sprite object.
- sf::Vector2f & **getScale** ()
Get the Scale object.
- sf::IntRect & **getRect** ()
Get the Rect object.
- sf::Vector2f & **getSize** ()
Get the Size object.
- int & **getSpace** ()
Get the Space object.
- sf::Vector2f & **getPosition** ()
Get the Position object.
- const sf::Texture & **getTexture** ()
Get the Texture object.
- bool **mouselsInSprite** (sf::Vector2i pmouse)
function that check if the cursor is in the sprite return true if the cursor is in the sprite, false otherwise
- void **playImage** (size_t nb)
function that play an exact sprite
- void **animSprite** ()
function that call play image with the next current sprite
- void **draw** (sf::RenderTarget &target, sf::RenderStates states) const override
Drawing the class on the window.

4.21.1 Member Function Documentation

4.21.1.1 draw()

```
void Spritesheet::draw (
    sf::RenderTarget & target,
    sf::RenderStates states ) const [override]
```

Drawing the class on the window.

Parameters

<i>target</i>	sf::RenderTarget window
<i>states</i>	sf::RenderStates Define the states used for drawing to a RenderTarget

4.21.1.2 getCurrentImage()

```
size_t & Spritesheet::getCurrentImage ( )
```

Get the Current Image object.

Returns

size_t&

4.21.1.3 getNbSprite()

```
size_t & Spritesheet::getNbSprite ( )
```

Get the Nb Sprite object.

Returns

size_t&

4.21.1.4 getPosition()

```
sf::Vector2f & Spritesheet::getPosition ( )
```

Get the Position object.

Returns

sf::Vector2f&

4.21.1.5 getRect()

```
sf::IntRect & Spritesheet::getRect ( )
```

Get the Rect object.

Returns

sf::IntRect&

4.21.1.6 getScale()

```
sf::Vector2f & Spritesheet::getScale ( )
```

Get the Scale object.

Returns

sf::Vector2f&

4.21.1.7 getSize()

```
sf::Vector2f & Spritesheet::getSize ( )
```

Get the Size object.

Returns

sf::Vector2f&

4.21.1.8 getSpace()

```
int & Spritesheet::getSpace ( )
```

Get the Space object.

Returns

int&

4.21.1.9 `getTexture()`

```
const sf::Texture & Spritesheet::getTexture ( )
```

Get the Texture object.

Returns

`sf::Texture&`

4.21.1.10 `mouseIsInSprite()`

```
bool Spritesheet::mouseIsInSprite (
    sf::Vector2i pmouse )
```

function that check if the cursor is in the sprite return true if the cursor is in the sprite, false otherwise

Parameters

<i>pmouse</i>	<code>sf::Vector2i</code> (pos of cursor on windows)
---------------	--

Returns

true

false

4.21.1.11 `playImage()`

```
void Spritesheet::playImage (
    size_t nb )
```

function that play an exact sprite

Parameters

<i>nb</i>	<code>size_t</code> (sprite nb wanted to display on window)
-----------	---

4.21.1.12 `setNbSprite()`

```
void Spritesheet::setNbSprite (
    size_t nb )
```


Set the Nb Sprite object.

Parameters

<i>nb</i>	size_t (nb of sprites on image)
-----------	---------------------------------

4.21.1.13 setPosition()

```
void Spritesheet::setPosition (
    sf::Vector2f pos )
```

Set the Position object.

Parameters

<i>pos</i>	sf::Vector2f (pos of texture on window)
------------	---

4.21.1.14 setRect()

```
void Spritesheet::setRect (
    sf::IntRect rect )
```

Set the Rect object.

Parameters

<i>rect</i>	sf::IntRect (rect of the sprite on image)
-------------	---

4.21.1.15 setRectSize()

```
void Spritesheet::setRectSize (
    sf::IntRect rect )
```

Set the Rect object (only width and height)

Parameters

<i>rect</i>	
-------------	--

4.21.1.16 setScale()

```
void Spritesheet::setScale (
    sf::Vector2f scale )
```

Set the Scale object.

Parameters

<i>scale</i>	sf::Vector2f (scale of sprite on window)
--------------	--

4.21.1.17 setSizeImage()

```
void Spritesheet::setSizeImage (
    sf::Vector2f size )
```

Set the Size Image object.

Parameters

<i>size</i>	sf::Vector2f (size of image)
-------------	------------------------------

4.21.1.18 setSpace()

```
void Spritesheet::setSpace (
    int space )
```

Set the Space object.

Parameters

<i>space</i>	int (space between all sprites on image)
--------------	--

4.21.1.19 setTexture()

```
void Spritesheet::setTexture (
    const sf::Texture & text )
```

Set the Texture object.

Parameters

<i>text</i>	const sf::Texture & (with a load texture)
-------------	---

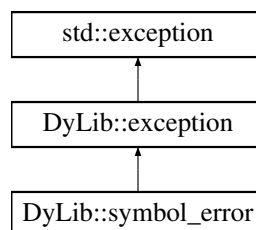
The documentation for this class was generated from the following file:

- include/UI/Spritesheet.hpp

4.22 DyLib::symbol_error Class Reference

```
#include <DyLib.hpp>
```

Inheritance diagram for DyLib::symbol_error:



Public Member Functions

- **symbol_error** (std::string &&message)

Additional Inherited Members

4.22.1 Detailed Description

This exception is thrown when the library failed to load a symbol. This usually happens when you forgot to mark a library function or variable as extern "C"

Parameters

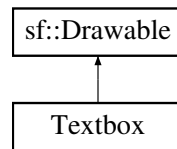
<i>message</i>	error message
----------------	---------------

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

- include/DyLib.hpp

4.23 Textbox Class Reference

Inheritance diagram for Textbox:



Public Member Functions

- **Textbox** (size_t size=20, sf::String str="", sf::Color color=sf::Color::White, sf::Vector2f pos={0, 0})
- void **setSize** (size_t size)
Set the Size object.
- void **setString** (sf::String str)
Set the String object.
- void **setColor** (sf::Color color)
Set the Color object.
- void **setPosition** (sf::Vector2f pos)
Set the Position object.
- void **setOutlineColor** (sf::Color color, float size)
Set the Outline Color object.
- void **setFont** (const sf::Font &font)
Set the Font object.
- sf::String & **getString** (void)
Get the String object.
- sf::Vector2f & **getPosition** (void)
Get the Position object.
- void **draw** (sf::RenderTarget &target, sf::RenderStates states) const override
Drawing the class on the window.

4.23.1 Member Function Documentation

4.23.1.1 draw()

```

void Textbox::draw (
    sf::RenderTarget & target,
    sf::RenderStates states ) const [override]
  
```

Drawing the class on the window.

Parameters

<i>target</i>	sf::RenderTarget window
<i>states</i>	sf::RenderStates Define the states used for drawing to a RenderTarget

4.23.1.2 getPosition()

```
sf::Vector2f & Textbox::getPosition (
    void )
```

Get the Position object.

Returns

sf::Vector2f&

4.23.1.3 getString()

```
sf::String & Textbox::getString (
    void )
```

Get the String object.

Returns

sf::String&

4.23.1.4 setColor()

```
void Textbox::setColor (
    sf::Color color )
```

Set the Color object.

Parameters

<i>color</i>	sf::Color (color of the text)
--------------	-------------------------------

4.23.1.5 setFont()

```
void Textbox::setFont (
    const sf::Font & font )
```

Set the Font object.

Parameters

<i>font</i>	const sf::Font & (font of text)
-------------	---------------------------------

4.23.1.6 setOutlineColor()

```
void Textbox::setOutlineColor (
    sf::Color color,
    float size )
```

Set the Outline Color object.

Parameters

<i>color</i>	sf::Color (outline color)
<i>size</i>	float (size of outline)

4.23.1.7 setPosition()

```
void Textbox::setPosition (
    sf::Vector2f pos )
```

Set the Position object.

Parameters

<i>pos</i>	sf::Vector2f (position of the text on window)
------------	---

4.23.1.8 setSize()

```
void Textbox::setSize (
    size_t size )
```

Set the Size object.

Parameters

<i>size</i>	size_t (set size of text)
-------------	---------------------------

4.23.1.9 setString()

```
void Textbox::setString (
    sf::String str )
```

Set the String object.

Parameters

<i>str</i>	sf::String (string to draw)
------------	-----------------------------

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

- include/UI/Textbox.hpp

4.24 thread_pool Class Reference

Public Member Functions

- **thread_pool** (const [thread_pool](#) &)=delete
- **thread_pool** ([thread_pool](#) &&)=delete
- [thread_pool](#) & **operator=** (const [thread_pool](#) &)=delete
- [thread_pool](#) & **operator=** ([thread_pool](#) &&)=delete
- template<typename Function , typename ... Args>
void [add](#) (Function &&fn, Args &&...args)
- void [remove](#) (size_t position)
- void [release](#) ()

4.24.1 Member Function Documentation

4.24.1.1 add()

```
template<typename Function , typename ... Args>
void thread_pool::add (
    Function && fn,
    Args &&... args ) [inline]
```

Start a function on another thread

Parameters

<i>fn</i>	function to be run on a new thread
<i>args</i>	arguments that will be forward to run the function

4.24.1.2 release()

```
void thread_pool::release ( ) [inline]
```

Release all threads from thread pool This function will be called by the class destructor

4.24.1.3 remove()

```
void thread_pool::remove (
    size_t position ) [inline]
```

Remove a thread from the thread pool

Parameters

<i>position</i>	position of the thread to be remove
-----------------	-------------------------------------

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

- include/ThreadPool.hpp

4.25 vec2float Struct Reference

Public Attributes

- float **x**
- float **y**

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

- include/UI/SpriteSize.hpp

Chapter 5

File Documentation

5.1 include/DyLib.hpp File Reference

Cross-platform Dynamic Library Loader.

```
#include <string>
#include <functional>
#include <exception>
#include <utility>
#include <dlfcn.h>
```

Classes

- class [DyLib](#)
- class [DyLib::exception](#)
- class [DyLib::handle_error](#)
- class [DyLib::symbol_error](#)

5.1.1 Detailed Description

Cross-platform Dynamic Library Loader.

Author

Martin Olivier

Version

1.6.1

MIT License Copyright (c) 2021 Martin Olivier

5.2 DyLib.hpp

[Go to the documentation of this file.](#)

```

1
11 #pragma once
12
13 #include <string>
14 #include <functional>
15 #include <exception>
16 #include <utility>
17 #if defined(_WIN32) || defined(_WIN64)
18 #define WIN32_LEAN_AND_MEAN
19 #include <windows.h>
20 #else
21 #include <dlfcn.h>
22 #endif
23
24 class DyLib
25 {
26 private:
27 #if defined(_WIN32) || defined(_WIN64)
28     HINSTANCE m_handle{nullptr};
29     static HINSTANCE openLib(const char *path) noexcept
30     {
31         return LoadLibraryA(path);
32     }
33     FARPROC getSymbol(const char *name) const noexcept
34     {
35         return GetProcAddress(m_handle, name);
36     }
37     void closeLib() noexcept
38     {
39         FreeLibrary(m_handle);
40     }
41     static char *getErrorMessage() noexcept
42     {
43         constexpr size_t bufSize = 512;
44         auto errorCode = GetLastError();
45         if (!errorCode)
46             return nullptr;
47         static char msg[bufSize];
48         auto lang = MAKELANGID(LANG_ENGLISH, SUBLANG_ENGLISH_US);
49         const DWORD len = FormatMessageA(FORMAT_MESSAGE_FROM_SYSTEM, nullptr, errorCode, lang, msg,
50 bufSize, nullptr);
51         if (len > 0)
52             return msg;
53         return nullptr;
54     }
55 #else
56     void *m_handle{nullptr};
57     static void *openLib(const char *path) noexcept
58     {
59         return dlopen(path, RTLD_NOW | RTLD_LOCAL);
60     }
61     void *getSymbol(const char *name) const noexcept
62     {
63         return dlsym(m_handle, name);
64     }
65     void closeLib() noexcept
66     {
67         dlclose(m_handle);
68     }
69     static char *getErrorMessage() noexcept
70     {
71         return dlerror();
72     }
73 #endif
74     static std::string getHandleError(const std::string &name)
75     {
76         auto err = getErrorMessage();
77         if (!err)
78             return "error while loading dynamic library \"" + name + "\"";
79         return err;
80     }
81     static std::string getSymbolError(const std::string &name)
82     {
83         auto err = getErrorMessage();
84         if (!err)
85             return "error while loading symbol \"" + name + "\"";
86         return err;
87     }
88 public:
89
90 #if defined(_WIN32) || defined(_WIN64)

```

```

91     static constexpr auto extension = ".dll";
92 #elif defined(__APPLE__)
93     static constexpr auto extension = ".dylib";
94 #else
95     static constexpr auto extension = ".so";
96 #endif
97
103     class exception : public std::exception
104     {
105     protected:
106         const std::string m_error;
107     public:
108         explicit exception(std::string &message) : m_error(std::move(message)) {}
109         const char *what() const noexcept override {return m_error.c_str();}
110     };
111
112     class handle_error : public exception
113     {
114     public:
115         explicit handle_error(std::string &message) : exception(std::move(message)) {}
116     };
117
118     class symbol_error : public exception
119     {
120     public:
121         explicit symbol_error(std::string &message) : exception(std::move(message)) {}
122     };
123
124     constexpr DyLib() noexcept = default;
125
126     DyLib(const DyLib&) = delete;
127     DyLib& operator=(const DyLib&) = delete;
128
129     DyLib(DyLib &&other) noexcept
130     {
131         m_handle = other.m_handle;
132         other.m_handle = nullptr;
133     }
134
135     DyLib& operator=(DyLib &&other) noexcept
136     {
137         if (this != &other) {
138             close();
139             m_handle = other.m_handle;
140             other.m_handle = nullptr;
141         }
142         return *this;
143     }
144
145     explicit DyLib(const char *path)
146     {
147         open(path);
148     }
149
150     explicit DyLib(const std::string &path)
151     {
152         open(path.c_str());
153     }
154
155     DyLib(std::string path, const char *ext)
156     {
157         open(std::move(path), ext);
158     }
159
160     ~DyLib()
161     {
162         close();
163     }
164
165     void open(const char *path)
166     {
167         close();
168         if (!path)
169             throw handle_error(getHandleError("(nullptr)"));
170         m_handle = openLib(path);
171         if (!m_handle)
172             throw handle_error(getHandleError(path));
173     }
174
175     void open(const std::string &path)
176     {
177         open(path.c_str());
178     }
179
180     void open(std::string path, const char *ext)
181     {
182         close();

```

```

216         if (!ext)
217             throw handle_error("bad extension : (nullptr)");
218         path += ext;
219         m_handle = openLib(path.c_str());
220         if (!m_handle)
221             throw handle_error(getHandleError(path));
222     }
223
224     template<typename T>
225     std::function<T> getFunction(const char *name) const
226     {
227         if (!m_handle)
228             throw handle_error("error : no dynamic library loaded");
229         if (!name)
230             throw symbol_error(getSymbolError("(nullptr)"));
231         auto sym = getSymbol(name);
232         if (!sym)
233             throw symbol_error(getSymbolError(name));
234         return reinterpret_cast<T *>(sym);
235     }
236
237     template<typename T>
238     std::function<T> getFunction(const std::string &name) const
239     {
240         return getFunction<T>(name.c_str());
241     }
242
243     template<typename T>
244     T &getVariable(const char *name) const
245     {
246         if (!m_handle)
247             throw handle_error("error : no dynamic library loaded");
248         if (!name)
249             throw symbol_error(getSymbolError("(nullptr)"));
250         auto sym = getSymbol(name);
251         if (!sym)
252             throw symbol_error(getSymbolError(name));
253         return *reinterpret_cast<T *>(sym);
254     }
255
256     template<typename T>
257     T &getVariable(const std::string &name) const
258     {
259         return getVariable<T>(name.c_str());
260     }
261
262     void close() noexcept
263     {
264         if (m_handle) {
265             closeLib();
266             m_handle = nullptr;
267         }
268     }
269 };
270

```

5.3 Components.hpp

```

1 #pragma once
2
3 #include <string>
4
5 namespace ecs {
6     namespace component {
7         struct position {
8             int x, y, vx, vy, cur;
9         };
10         using pv = int;
11         using label = std::string;
12     }
13 }

```

5.4 Entity.hpp

```

1 #pragma once
2
3 #include <cstdint>
4
5 namespace ecs {
6     using entity = size_t;
7 }

```

5.5 Exception.hpp

```

1 #pragma once
2
3 #include <exception>
4 #include <string>
5 #include <utility>
6
7 namespace ecs {
8 class exception : public std::exception
9 {
10 private:
11     std::string m_err;
12 public:
13     explicit exception(std::string err) : m_err(std::move(err)) {}
14     template<typename ...T>
15     exception(T &&...err) : m_err() {m_err.append(std::forward<T>(err)...);}
16     ~exception() override = default;
17     [[nodiscard]] inline const char *what() const noexcept override {return m_err.c_str();}
18 };
19 }

```

5.6 Module.hpp

```

1 #pragma once
2
3 #include <string>
4 #include "Registry.hpp"
5
6 namespace ecs {
7 class module
8 {
9 public:
10     virtual ~module() = default;
11     virtual ecs::entity spawn(registry &) = 0;
12     virtual std::function<void(registry &)> get_system() = 0;
13     virtual std::string get_label() = 0;
14 };
15 }

```

5.7 Registry.hpp

```

1 #pragma once
2
3 #include <map>
4 #include <any>
5 #include <typeindex>
6 #include <functional>
7
8 #include "Entity.hpp"
9 #include "SparseArray.hpp"
10 #include "Exception.hpp"
11
12 namespace ecs {
13 class registry {
14 public:
15
16     template<typename T>
17     sparse_array<T> &register_component() {
18         if (!has_component<T>())
19             m_map[std::type_index(typeid(T))] = std::make_any<sparse_array<T>();
20         m_deleter.push_back([&](entity ent) {
21             std::any_cast<sparse_array<T> &>(m_map[std::type_index(typeid(T))]).erase(ent);
22         });
23         return std::any_cast<sparse_array<T> &>(m_map[std::type_index(typeid(T))]);
24     }
25
26     template<class T>
27     bool has_component() {
28         if (m_map.contains(std::type_index(typeid(T))))
29             return true;
30         return false;
31     }
32
33     template<typename T>
34     sparse_array<T> &get_components() {
35         if (!has_component<T>())
36             throw exception("registry::get_components");
37         return std::any_cast<sparse_array<T> &>(m_map[std::type_index(typeid(T))]);
38     }
39 }

```

```

56     }
57
58     template <typename T>
59     const sparse_array<T> &get_components() const {
60         if (!has_component<T>())
61             throw exception("registry::get_components");
62         return std::any_cast<const sparse_array<T> &>(m_map.at(std::type_index(typeid(T))));
63     }
64
65     template <typename T>
66     typename sparse_array<T>::reference_type insert_component(const entity &ent, T &&comp) {
67         if (!has_component<T>())
68             throw exception("registry::insert_component");
69         return std::any_cast<sparse_array<T> &>(m_map[std::type_index(typeid(T))]).insert_at(ent,
70 std::forward<T>(comp));
71     }
72
73     template <typename T, typename ...Params>
74     typename sparse_array<T>::reference_type emplace_component(const entity &ent, Params && ...params) {
75         if (!has_component<T>())
76             throw exception("registry::emplace_component");
77         return std::any_cast<sparse_array<T> &>(m_map[std::type_index(typeid(T))]).emplace_at(ent,
78 std::forward<Params>(params)...);
79     }
80
81     template <typename T>
82     void remove_component(const entity &ent) {
83         if (!has_component<T>())
84             throw exception("registry::remove_component");
85         std::any_cast<sparse_array<T> &>(m_map[std::type_index(typeid(T))]).erase(ent);
86     }
87
88     entity spawn_entity() {
89         if (m_ents_available.empty()) {
90             m_ent_iter++;
91             m_pool.push_back(m_ent_iter);
92             return m_ent_iter;
93         }
94         auto it = m_ents_available.front();
95         m_ents_available.erase(m_ents_available.begin());
96         m_pool.push_back(it);
97         return it;
98     }
99
100     void kill_entity(entity e) {
101         auto res = std::find(m_pool.begin(), m_pool.end(), e);
102         if (res == std::end(m_pool))
103             throw exception("entity_pool::kill_entity");
104         m_pool.erase(res);
105         m_ents_available.push_back(e);
106         for (auto &del : m_deleter)
107             del(e);
108     }
109
110     const std::vector<entity> &get_entities() {
111         return m_pool;
112     }
113
114     template <typename Function>
115     void add_system(Function && f) {
116         m_systems.emplace_back([&] () {
117             std::forward<Function>(f) (*this);
118         });
119     }
120
121     void run_systems() {
122         for (auto &it : m_systems)
123             it();
124     }
125
126     void destroy() {
127         m_map.clear();
128         m_pool.clear();
129         m_ents_available.clear();
130         m_deleter.clear();
131         m_systems.clear();
132         m_ent_iter = 0;
133     }
134
135 private:
136     std::map<std::type_index, std::any> m_map{};
137     std::vector<entity> m_pool{};
138     std::vector<entity> m_ents_available{};
139     std::vector<std::function<void(entity)>> m_deleter{};
140     std::vector<std::function<void()>> m_systems{};
141     entity m_ent_iter{};

```



```
199 };
200 }
```

5.8 SparseArray.hpp

```
1 #pragma once
2
3 #include <optional>
4 #include <vector>
5 #include <utility>
6
7 namespace ecs {
8 template <typename T>
9 class sparse_array {
10 public:
11     using value_type = std::optional<T>;
12     using reference_type = value_type &;
13     using const_reference_type = const value_type &;
14     using container_t = std::vector<value_type>;
15     using size_type = typename container_t::size_type;
16     using iterator = typename container_t::iterator;
17     using const_iterator = typename container_t::const_iterator;
18
19     sparse_array() = default;
20     sparse_array(sparse_array const &) = default;
21     sparse_array(sparse_array &&) noexcept = default;
22     ~sparse_array() = default;
23     sparse_array &operator=(sparse_array const &) = default;
24     sparse_array &operator=(sparse_array &&) noexcept = default;
25     iterator begin() {return m_data.begin();}
26     const_iterator begin() const {return m_data.begin();}
27     const_iterator cbegin() const {return m_data.cbegin();}
28     iterator end() {return m_data.end();}
29     const_iterator end() const {return m_data.end();}
30     const_iterator cend() const {return m_data.cend();}
31     size_type size() const {return m_data.size();}
32     reference_type operator[](size_t idx) {
33         if (idx >= size())
34             m_data.resize(idx + 1);
35         return m_data[idx];
36     }
37     const_reference_type operator[](size_t idx) const {
38         if (idx >= size())
39             m_data.resize(idx + 1);
40         return m_data[idx];
41     }
42
43     reference_type insert_at(size_type pos, const T &comp) {
44         if (pos >= size())
45             m_data.resize(pos + 1);
46         m_data[pos] = std::make_optional<T>(comp);
47         return m_data[pos];
48     }
49
50     reference_type insert_at(size_type pos, T &&comp) {
51         if (pos >= size())
52             m_data.resize(pos + 1);
53         m_data[pos] = std::make_optional<T>(std::forward<T>(comp));
54         return m_data[pos];
55     }
56
57     template <class ...Params>
58     reference_type emplace_at(size_type pos, Params &&...params) {
59         if (pos >= size())
60             m_data.resize(pos + 1);
61         m_data[pos] = std::make_optional<T>(std::forward<T>(params)...);
62         return m_data[pos];
63     }
64
65     void erase(size_type pos) {
66         if (pos < m_data.size())
67             m_data[pos] = std::nullopt;
68     }
69
70     size_type get_index(value_type const &val) const {
71         for (size_t i = 0; i < m_data.size(); i++) {
72             if (m_data[i] == val)
73                 return i;
74         }
75         return -1;
76     }
77 private:
78     mutable container_t m_data{};
79 }
```

```
116 };
117 }
```

5.9 Systems.hpp

```
1 #pragma once
2
3 #include "Registry.hpp"
4 #include "Components.hpp"
5 #include "UI/SpriteSize.hpp"
6 #include <iostream>
7
8 namespace ecs {
9 namespace system {
10     inline void create_players(registry &reg, size_t client_size) {
11         for (size_t i = 0; i < client_size; i++) {
12             auto ent = reg.spawn_entity();
13             reg.insert_component<ecs::component::label>(ent, "player" + std::to_string(i));
14             reg.insert_component<ecs::component::position>(ent, {0, 100 + static_cast<int>(100 * i), 0,
15                 0, 0});
16             reg.insert_component<ecs::component::pv>(ent, 1);
17         }
18     }
19
20     inline void move_player(registry &reg, size_t client_pos, int x, int y) {
21         auto &positions = reg.get_components<ecs::component::position>();
22         auto &tags = reg.get_components<ecs::component::label>();
23         for (auto &e : reg.get_entities()) {
24             if (tags[e].value() == "player" + std::to_string(client_pos)) {
25                 auto &val = positions[e].value();
26                 if (y > 0)
27                     val.vy = 1;
28                 else if (y < 0)
29                     val.vy = -1;
30                 else
31                     val.vy = 0;
32                 val.x += x * 15;
33                 val.y += y * 15;
34                 if (val.x < 0)
35                     val.x = 0;
36                 if (val.y < 0)
37                     val.y = 0;
38                 if (val.x > 1856)
39                     val.x = 1856;
40                 if (val.y > 1052)
41                     val.y = 1052;
42                 return;
43             }
44         }
45     }
46
47     inline void create_friend_shoot(registry &reg, size_t client_pos) {
48         auto &tags = reg.get_components<ecs::component::label>();
49         auto &positions = reg.get_components<ecs::component::position>();
50         for (auto player : reg.get_entities()) {
51             if (tags[player].value() == "player" + std::to_string(client_pos)) {
52                 auto ent = reg.spawn_entity();
53                 auto player_pos = positions[player].value();
54                 reg.insert_component<ecs::component::label>(ent, "friend_shoot");
55                 reg.insert_component<ecs::component::position>(ent,
56                     {player_pos.x + static_cast<int>(Data::sprite_data["player"].first.x *
57                         Data::sprite_data["player"].second.x),
58                     player_pos.y + static_cast<int>((Data::sprite_data["player"].first.y *
59                         Data::sprite_data["player"].second.y) / 2), 20, 0, 0});
60                 reg.insert_component<ecs::component::pv>(ent, 1);
61                 return;
62             }
63         }
64     }
65
66     inline void create_enemy_shoot(registry &reg, size_t enemy_ent, int speed = -5) {
67         auto &label = reg.get_components<ecs::component::label>();
68         auto &positions = reg.get_components<ecs::component::position>();
69
70         auto ent = reg.spawn_entity();
71         auto enemy_pos = positions[enemy_ent].value();
72         reg.insert_component<ecs::component::label>(ent, "enemy_shoot");
73         reg.insert_component<ecs::component::position>(ent,
74             {enemy_pos.x, enemy_pos.y + static_cast<int>((Data::sprite_data[label[enemy_ent].value()].first.y
75                 * Data::sprite_data[label[enemy_ent].value()].second.y) / 2), speed, 0, 0});
76         reg.insert_component<ecs::component::pv>(ent, 1);
77     }
78 }
```

```

76     inline void update_velocity(registry &reg) {
77         auto &positions = reg.get_components<ecs::component::position>();
78         auto &tags = reg.get_components<ecs::component::label>();
79         auto &pvs = reg.get_components<ecs::component::pv>();
80         for (auto &e : reg.get_entities()) {
81             if (tags[e].value().find("player") != 0) {
82                 positions[e].value().x += positions[e].value().vx;
83                 positions[e].value().y += positions[e].value().vy;
84                 positions[e].value().cur += positions[e].value().vy;
85                 if (positions[e].value().x > 1920 or positions[e].value().x < -100)
86                     pvs[e].value() = 0;
87             }
88         }
89     }
90
91     inline void entity_killer(registry &reg) {
92         auto &pvs = reg.get_components<ecs::component::pv>();
93         for (auto &e : reg.get_entities()) {
94             if (pvs[e].value() <= 0) {
95                 reg.kill_entity(e);
96                 return;
97             }
98         }
99     }
100
101     bool is_collision(registry &reg, entity e1, entity e2) {
102         auto &label = reg.get_components<ecs::component::label>();
103         auto &l1 = label[e1].value();
104         auto &l2 = label[e2].value();
105         auto &positions = reg.get_components<ecs::component::position>();
106         auto &pos1 = positions[e1].value();
107         auto &pos2 = positions[e2].value();
108         float rect1[2] = {Data::sprite_data[l1].first.x * Data::sprite_data[l1].second.x,
109                          Data::sprite_data[l1].first.y * Data::sprite_data[l1].second.y};
110         float rect2[2] = {Data::sprite_data[l2].first.x * Data::sprite_data[l2].second.x,
111                          Data::sprite_data[l2].first.y * Data::sprite_data[l2].second.y};
112
113         if (pos1.x >= pos2.x && pos1.x <= pos2.x + rect2[0] && pos1.y >= pos2.y && pos1.y <= pos2.y +
114             rect2[1])
115             return true;
116         if (pos1.x + rect1[0] >= pos2.x && pos1.x + rect1[0] <= pos2.x + rect2[0] && pos1.y >= pos2.y &&
117             pos1.y <= pos2.y + rect2[1])
118             return true;
119         if (pos1.x >= pos2.x && pos1.x <= pos2.x + rect2[0] && pos1.y + rect1[1] >= pos2.y && pos1.y +
120             rect1[1] <= pos2.y + rect2[1])
121             return true;
122         if (pos1.x + rect1[0] >= pos2.x && pos1.x + rect1[0] <= pos2.x + rect2[0] && pos1.y + rect1[1]
123             >= pos2.y + rect1[1] <= pos2.y + rect2[1])
124             return true;
125         if (pos2.x >= pos1.x && pos2.x <= pos1.x + rect1[0] && pos2.y >= pos1.y && pos2.y <= pos1.y +
126             rect1[1])
127             return true;
128         if (pos2.x + rect2[0] >= pos1.x && pos2.x + rect2[0] <= pos1.x + rect1[0] && pos2.y >= pos1.y &&
129             pos2.y <= pos1.y + rect1[1])
130             return true;
131         if (pos2.x >= pos1.x && pos2.x <= pos1.x + rect1[0] && pos2.y + rect2[1] >= pos1.y && pos2.y +
132             rect2[1] <= pos1.y + rect1[1])
133             return true;
134         if (pos2.x + rect2[0] >= pos1.x && pos2.x + rect2[0] <= pos1.x + rect1[0] && pos2.y + rect2[1]
135             >= pos1.y + rect2[1] <= pos1.y + rect1[1])
136             return true;
137         return false;
138     }
139
140     inline void collision_system(registry &reg) {
141         auto &labels = reg.get_components<ecs::component::label>();
142         auto &pvs = reg.get_components<ecs::component::pv>();
143         for (auto e1 : reg.get_entities()) {
144             for (auto e2 : reg.get_entities()) {
145                 if (e1 != e2 and is_collision(reg, e1, e2)) {
146                     if ((labels[e1].value().find("enemy") == 0 and labels[e2].value().find("player") ==
147                         0) and
148                         labels[e1].value().find("enemy_shoot") != 0) {
149                         pvs[e1].value() -= 1;
150                         pvs[e2].value() -= 1;
151                     }
152                     if ((labels[e1].value().find("enemy_shoot") == 0 and
153                         labels[e2].value().find("player") == 0) and
154                         labels[e2].value().find("player_shoot") != 0) {
155                         pvs[e1].value() -= 1;
156                         pvs[e2].value() -= 1;
157                     }
158                     if ((labels[e1].value().find("friend_shoot") == 0 and
159                         labels[e2].value().find("enemy") == 0) and
160                         labels[e2].value().find("enemy_shoot") != 0) {
161                         pvs[e1].value() -= 1;
162                         pvs[e2].value() -= 1;
163                     }
164                 }
165             }
166         }
167     }

```

```

152         }
153     }
154 }
155 }
156 }
157 }
158 }

```

5.10 Client.hpp

```

1 #pragma once
2
3 #include <asio.hpp>
4 #include "Packet.hpp"
5
6 namespace network {
7     class client {
8     private:
9         asio::ip::udp::endpoint endpoint;
10        std::vector<packet> buffer{};
11        bool m_alive = true;
12    public:
13        client(asio::ip::udp::endpoint &endp) : endpoint(std::move(endp)) {}
14        ~client() = default;
15        std::string get_ip() const {return endpoint.address().to_string();}
16        unsigned short get_port() {return endpoint.port();}
17        const asio::ip::udp::endpoint &get_endpoint() const noexcept {return endpoint;}
18        std::vector<packet> &get_packets() noexcept {return buffer;}
19        std::vector<packet> pop_packets() noexcept {return std::move(buffer);}
20        bool &alive() {return m_alive;}
21    };
22 }

```

5.11 Header.hpp

```

1 #pragma once
2
3 #include <cstdint>
4
5 namespace network {
6     struct header {
7         uint32_t magicValue;
8         uint8_t type;
9         uint32_t size;
10    };
11 }

```

5.12 Packet.hpp

```

1 #pragma once
2
3 #include <cstdint>
4 #include <string>
5
6 namespace network {
7     struct packet {
8         uint8_t type;
9         std::string data;
10    };
11 }

```

5.13 Server.hpp

```

1 #pragma once
2
3 #include <asio.hpp>
4 #include <iostream>
5 #include <memory>
6 #include <vector>
7 #include <chrono>
8

```

```

9 #include "Header.hpp"
10 #include "Packet.hpp"
11 #include "Client.hpp"
12
13 namespace network {
14 class server {
15 private:
16     asio::io_context m_io_context{};
17     asio::ip::udp::socket m_socket;
18     std::vector<client> m_clients{};
19     uint8_t m_login_handshake;
20     uint8_t m_logout_handshake;
21     uint8_t m_alive_handshake;
22     size_t m_max_clients;
23     bool m_allow_new_clients = true;
24     std::chrono::system_clock::time_point m_check_alive = std::chrono::system_clock::now();
25 public:
26     server(unsigned short port, size_t max_clients, uint8_t login_handshake, uint8_t logout_handshake,
27         uint8_t alive_handshake)
28         : m_socket(m_io_context, asio::ip::udp::endpoint(asio::ip::udp::v4(), port)),
29         m_login_handshake(login_handshake), m_logout_handshake(logout_handshake),
30         m_alive_handshake(alive_handshake), m_max_clients(max_clients) {
31         m_socket.non_blocking(true);
32     }
33     ~server() = default;
34     server(server &) = delete;
35     server &operator=(server &) = delete;
36     server(server &&) = delete;
37     server &operator=(server &&) = delete;
38
39     void send(const client &c, uint8_t type, const std::string &buffer = "", unsigned short repeat = 1) {
40         size_t size = buffer.size();
41         auto p = reinterpret_cast<header*> (::operator new (sizeof(header) + size));
42         p->magicValue = 0x42dead42;
43         p->type = type;
44         p->size = buffer.size();
45         std::memcpy(reinterpret_cast<uint8_t*>(p) + sizeof(header), buffer.data(), size);
46         try {
47             while (repeat != 0) {
48                 m_socket.send_to(asio::buffer(reinterpret_cast<const uint8_t*>(p), sizeof(header) +
49 size), c.get_endpoint());
50                 repeat--;
51             }
52         } catch (...) {}
53         delete p;
54     }
55     void receive() {
56         if (std::chrono::system_clock::now() > m_check_alive + std::chrono::seconds(10)) {
57             m_check_alive = std::chrono::system_clock::now();
58             for (size_t i = 0; i < m_clients.size(); i++) {
59                 if (m_clients[i].alive() == false) {
60                     std::cout << "[USER] " << m_clients[i].get_ip() << " : DISCONNECTED" << std::endl;
61                     m_clients.erase(m_clients.begin() + i);
62                     i--;
63                 } else {
64                     m_clients[i].alive() = false;
65                 }
66             }
67         }
68         while (true) {
69             asio::ip::udp::endpoint endpoint;
70             uint8_t recv_str[4800];
71             asio::error_code error;
72             auto len = m_socket.receive_from(asio::buffer(recv_str, 4800), endpoint, 0, error);
73             if (error == asio::error::would_block)
74                 return;
75             std::string data{};
76             auto ret = reinterpret_cast<const header*>(recv_str);
77             if (ret->magicValue != 0x42dead42 or len != sizeof(header) + (ret->size))
78                 return;
79             for (size_t i = 0; i < ret->size; i++)
80                 data.push_back(((reinterpret_cast<const uint8_t*>(ret) + sizeof(header))[i]);
81             for (size_t i = 0; i < m_clients.size(); i++) {
82                 if (m_clients[i].get_ip() == endpoint.address().to_string() and m_clients[i].get_port()
83 == endpoint.port()) {
84                     if (ret->type == m_alive_handshake) {
85                         m_clients[i].alive() = true;
86                         continue;
87                     } else if (ret->type == m_logout_handshake) {
88                         std::cout << "[USER] " << m_clients[i].get_ip() << " : DISCONNECTED" << std::endl;
89                         m_clients.erase(m_clients.begin() + i);
90                         return;
91                     }
92                     m_clients[i].get_packets().push_back({ret->type, std::move(data)});
93                     return;
94                 }
95             }
96         }
97     }
98 }
99
100 }
101
102 }
103
104

```

```

105         if (ret->type == m_login_handshake and m_clients.size() < m_max_clients and
m_allow_new_clients == true) {
106             m_clients.push_back(std::move(endpoint));
107             std::cout << "[USER] " << m_clients.back().get_ip() << " : CONNECTED" << std::endl;
108             m_check_alive = std::chrono::system_clock::now();
109             send(m_clients.back(), m_login_handshake, "", 10);
110         }
111     }
112 }
119 std::vector<client> &get_clients() noexcept {
120     return m_clients;
121 }
127 void allow_new_connections(bool value) {m_allow_new_clients = value;}
128 };
129 }

```

5.14 Socket.hpp

```

1 #pragma once
2
3 #include <asio.hpp>
4 #include <thread>
5 #include <memory>
6 #include <vector>
7 #include <chrono>
8 #include <mutex>
9 #include <exception>
10
11 #include "Header.hpp"
12 #include "Packet.hpp"
13
14 namespace network {
15 class socket {
16 private:
17     asio::io_context m_io_context{};
18     asio::ip::udp::socket m_socket;
19     std::unique_ptr<asio::ip::udp::endpoint> m_server{};
20     std::unique_ptr<std::thread> m_receiver{};
21     std::mutex m_mutex{};
22     std::vector<packet> m_buffer{};
23     uint8_t m_login_handshake;
24     uint8_t m_logout_handshake;
25     uint8_t m_alive_handshake;
26     bool m_connected{};
27 public:
28     socket(unsigned short port, uint8_t login_handshake, uint8_t logout_handshake, uint8_t
alive_handshake)
29         : m_socket(m_io_context, asio::ip::udp::endpoint(asio::ip::udp::v4(), port)),
30           m_login_handshake(login_handshake), m_logout_handshake(logout_handshake),
31           m_alive_handshake(alive_handshake) {
32         m_socket.non_blocking(true);
33     }
34     ~socket() {disconnect();}
35     socket(socket &) = delete;
36     socket &operator=(socket &) = delete;
37     socket(socket &&) = delete;
38     socket &operator=(socket &&) = delete;
39
40     void connect(std::string ip, unsigned short port, long login_timeout = 3) {
41         disconnect();
42         std::this_thread::sleep_for(std::chrono::seconds(1));
43         try {
44             std::chrono::system_clock::time_point deadline = std::chrono::system_clock::now() +
std::chrono::seconds(login_timeout);
45             m_server = std::make_unique<asio::ip::udp::endpoint>(asio::ip::make_address(std::move(ip)),
port);
46             m_connected = true;
47             m_receiver = std::make_unique<std::thread>(receive, std::ref(*this));
48             send(m_login_handshake, "", 10);
49             while (std::chrono::system_clock::now() < deadline) {
50                 auto packets = pop_packets();
51                 for (auto &p : packets) {
52                     if (p.type == m_login_handshake)
53                         return;
54                 }
55                 std::this_thread::yield();
56             }
57             throw std::exception();
58         } catch (...) {
59             disconnect();
60             throw std::runtime_error("network::client::connect : could not connect to " + ip + ":" +
std::to_string(port));
61         }
62     }
63 }
64 }

```

```

68     }
69
70     void disconnect() noexcept {
71         if (m_receiver) {
72             m_connected = false;
73             m_receiver->join();
74         }
75         try {
76             send(m_logout_handshake, "", 10);
77         } catch (...) {}
78         m_server = nullptr;
79         m_receiver = nullptr;
80         m_buffer.clear();
81     }
82
83     void send(uint8_t type, const std::string &buffer = "", unsigned short repeat = 1) {
84         if (!m_server)
85             throw std::runtime_error("network::client::send : not connected to any host");
86         size_t size = buffer.size();
87         auto p = reinterpret_cast<header*> (::operator new (sizeof(header) + size));
88         p->magicValue = 0x42dead42;
89         p->type = type;
90         p->size = buffer.size();
91         std::memcpy(reinterpret_cast<uint8_t*>(p) + sizeof(header), buffer.data(), size);
92         try {
93             while (repeat != 0) {
94                 m_socket.send_to(asio::buffer(reinterpret_cast<const uint8_t*>(p), sizeof(header) +
95 size), *m_server);
96                 repeat--;
97             }
98         } catch (...) {}
99         delete p;
100     }
101
102     std::string get_remote_ip() const {
103         if (!m_server)
104             return "";
105         return m_server->address().to_string();
106     }
107
108     unsigned short get_remote_port() const {
109         if (!m_server)
110             return 0;
111         return m_server->port();
112     }
113
114     std::vector<packet> get_packets() noexcept {
115         std::lock_guard lock(m_mutex);
116         return m_buffer;
117     }
118
119     std::vector<packet> pop_packets() noexcept {
120         std::lock_guard lock(m_mutex);
121         return std::move(m_buffer);
122     }
123
124 private:
125     static void receive(socket &self) {
126         std::chrono::system_clock::time_point chrono = std::chrono::system_clock::now();
127
128         if (!self.m_server)
129             throw std::runtime_error("network::client::receive : not connected to any host");
130
131         while (self.m_connected) {
132             if (std::chrono::system_clock::now() > chrono + std::chrono::seconds(1)) {
133                 chrono = std::chrono::system_clock::now();
134                 self.send(self.m_alive_handshake, "", 1);
135             }
136             asio::ip::udp::endpoint endpoint;
137             uint8_t recv_str[4800];
138             asio::error_code error;
139             std::string data{};
140             auto len = self.m_socket.receive_from(asio::buffer(recv_str, 4800), endpoint, 0, error);
141             if (error == asio::error::would_block)
142                 continue;
143             if (self.get_remote_ip() != endpoint.address().to_string() or self.get_remote_port() !=
144 endpoint.port())
145                 continue;
146             auto ret = reinterpret_cast<const header*>(recv_str);
147             if (ret->magicValue == 0x42dead42 and len == sizeof(header) + (ret->size)) {
148                 for (size_t i = 0; i < ret->size; i++)
149                     data.push_back(((reinterpret_cast<const uint8_t*>(ret) + sizeof(header))[i]);
150                 self.m_mutex.lock();
151                 self.m_buffer.push_back({ret->type, std::move(data)});
152                 self.m_mutex.unlock();
153             }
154             std::this_thread::yield();
155         }
156     }

```

```

189     }
190 };
191 }

```

5.15 Type.hpp

```

1 #pragma once
2
3 #include <cstdint>
4
5 namespace network {
11 enum type : uint8_t {
12     Login,
13     Logout,
14     PingAlive,
15     UsernameOK,
16     UsernameKO,
17     Lobby,
18     Ready,
19     NotReady,
20     GameStart,
21     GameUpdate,
22     PlayerInput,
23     PlayerDead,
24 };
25 }

```

5.16 Random.hpp

```

1 #pragma once
2
3 namespace tools {
10     inline int random(int max) {
11         srand(time(NULL));
12         return rand() % max;
13     }
14 }

```

5.17 ThreadPool.hpp

```

1 #pragma once
2
3 #include <thread>
4 #include <vector>
5 #include <utility>
6
7 class thread_pool
8 {
9 private:
10     std::vector<std::thread> m_threads{};
11 public:
12     thread_pool() = default;
13     thread_pool(const thread_pool&) = delete;
14     thread_pool(thread_pool&&) = delete;
15     thread_pool &operator=(const thread_pool&) = delete;
16     thread_pool &operator=(thread_pool&&) = delete;
17     ~thread_pool() {
18         release();
19     };
20     template<typename Function, typename ...Args>
21     void add(Function &&fn, Args &&...args) {
22         m_threads.emplace_back(std::forward<Function>(fn), std::forward<Args>(args)...);
23     }
24     void remove(size_t position) {
25         if (position >= m_threads.size())
26             throw std::out_of_range("thread_pool::remove");
27         m_threads[position].join();
28         m_threads.erase(m_threads.begin() + position);
29     }
30     void release() {
31         for (auto &thread : m_threads)
32             thread.join();
33         m_threads.clear();
34     }
35 };

```


5.18 Tools.hpp

```

1 #pragma once
2
3 #include <string>
4 #include <vector>
5 #include <SFML/Graphics.hpp>
6 #include "UI/Spritesheet.hpp"
7 #include "UI/Inputbox.hpp"
8
9 namespace tools {
10     inline std::vector<std::string> string_to_vector(const std::string &str, char separator)
11     {
12         std::vector<std::string> array{};
13         std::string temp{};
14         size_t len = str.size();
15
16         for (size_t i = 0; i < len; i++) {
17             if (str[i] == separator) {
18                 array.push_back(temp);
19                 temp.clear();
20             }
21             else
22                 temp.push_back(str[i]);
23         }
24         if (temp.size() != 0) {
25             array.push_back(temp);
26         }
27         return array;
28     }
29     inline bool mouse_is_in_shape(sf::Vector2i pmouse, sf::RectangleShape shape)
30     {
31         if (pmouse.x >= shape.getPosition().x && pmouse.x <= shape.getPosition().x + shape.getSize().x
32             && pmouse.y >= shape.getPosition().y && pmouse.y <= shape.getPosition().y + shape.getSize().y) {
33             return true;
34         }
35         return (false);
36     }
37     inline void state_button_login(const std::vector<Inputbox> &box, Spritesheet &sprite)
38     {
39         for (auto &c : box) {
40             if (c.getText().getSize() == 0) {
41                 sprite.playImage(0);
42                 return;
43             }
44         }
45         sprite.playImage(1);
46     }
47 }

```

5.19 TypeChecker.hpp

```

1 #pragma once
2
3 #include <type_traits>
4
5 template<typename T, typename ...List>
6 concept TypeChecker = (std::is_same_v<T, List> || ...);

```

5.20 Core.hpp

```

1 #pragma once
2
3 #include <SFML/Graphics.hpp>
4 #include "Network/Type.hpp"
5 #include "Network/Socket.hpp"
6 #include "UI/Status.hpp"
7 #include "UI/Menu.hpp"
8 #include "UI/Lobby.hpp"
9 #include "UI/Game.hpp"
10
11 class Core {
12 public:
13     Core();
14     ~Core() = default;
15
16     void loop();
17 }

```

```

23     private:
24         uint8_t m_status = MENU;
25         sf::RenderWindow m_window{};
26         network::socket m_socket;
27         Menu m_menu;
28         Lobby m_lobby;
29         Game m_game;
30 };

```

5.21 Data.hpp

```

1  #pragma once
2
3  #include <SFML/Graphics.hpp>
4  #include <SFML/Window.hpp>
5  #include <SFML/System.hpp>
6  #include <SFML/Audio.hpp>
7  #include <exception>
8  #include <map>
9  #include <functional>
10 #include "UI/Spritesheet.hpp"
11
12 namespace Data {
13     class MissingAsset : public std::exception {
14     private:
15         const std::string m_error{};
16     public:
17         explicit MissingAsset(const std::string &path) : m_error("missing asset : " + path) {}
18         ~MissingAsset() override = default;
19         [[nodiscard]] inline const char *what() const noexcept override {return m_error.c_str();}
20     };
21
22     inline sf::Texture backgroundTexture{};
23     inline sf::Font font{};
24
25     inline sf::Music menuMusic{};
26     inline sf::Music gameMusic{};
27
28     inline sf::Texture tmenu{};
29     inline sf::Texture tloginbutton{};
30     inline sf::Texture tprofile{};
31     inline sf::Texture tready{};
32     inline sf::Texture treadybutton{};
33     inline sf::Texture tdisconnectbutton{};
34     inline sf::Texture tloading{};
35     inline sf::Texture tmap{};
36     inline sf::Texture tmapdead{};
37     inline sf::Texture tplayer0{};
38     inline sf::Texture tplayer1{};
39     inline sf::Texture tplayer2{};
40     inline sf::Texture tplayer3{};
41     inline sf::Texture tenemy0{};
42     inline sf::Texture tenemy1{};
43     inline sf::Texture tenemy2{};
44     inline sf::Texture tplayershot{};
45     inline sf::Texture tenemyshot{};
46     inline sf::Texture tlobbybutton{};
47     inline sf::Texture tdisconnectbuttongame{};
48
49     inline sf::Sound splayershot{};
50     inline sf::SoundBuffer bplayershot{};
51
52
53     inline std::map<std::string, std::function<Spritesheet()>> factorytexturemap {
54         {"player0", {[](){return Spritesheet(tplayer0, {0, 0}, 5, {160, 14}, {0, 0, 32, 14}, {2.5, 2.5},
55             32);}}},
56         {"player1", {[](){return Spritesheet(tplayer1, {0, 0}, 5, {160, 14}, {0, 0, 32, 14}, {2.5, 2.5},
57             32);}}},
58         {"player2", {[](){return Spritesheet(tplayer2, {0, 0}, 5, {160, 14}, {0, 0, 32, 14}, {2.5, 2.5},
59             32);}}},
60         {"player3", {[](){return Spritesheet(tplayer3, {0, 0}, 5, {160, 14}, {0, 0, 32, 14}, {2.5, 2.5},
61             32);}}},
62         {"enemy_plane", {[](){return Spritesheet(tenemy0, {0, 0}, 8, {168, 24}, {0, 0, 21, 24}, {2.5,
63             2.5}, 21);}}},
64         {"enemy_cyborg", {[](){return Spritesheet(tenemy1, {0, 0}, 4, {128, 31}, {0, 0, 32, 31}, {3, 3},
65             32);}}},
66         {"enemy2", {[](){return Spritesheet(tenemy1, {0, 0}, 4, {256, 22}, {0, 0, 32, 22}, {2, 2},
67             32);}}},
68         {"friend_shoot", {[](){return Spritesheet(tplayershot, {0, 0}, 1, {16, 4}, {0, 0, 16, 4}, {2,
69             2);}}},
70         {"enemy_shoot", {[](){return Spritesheet(tenemyshot, {0, 0}, 4, {28, 6}, {0, 0, 7, 6}, {2, 2},
71             7);}}},
72     };
73 }

```

```

64
65     void load();
66
67     inline void loadAsset(sf::Music &asset, const std::string &path) {
68         if (!asset.openFromFile(path))
69             throw MissingAsset(path);
70     }
71
72     template<typename T>
73     inline void loadAsset(T &asset, const std::string &path) {
74         if (!asset.loadFromFile(path))
75             throw MissingAsset(path);
76     }
77 };

```

5.22 Game.hpp

```

1  #pragma once
2
3  #include <SFML/Graphics.hpp>
4  #include "ECS/SparseArray.hpp"
5  #include "Network/Socket.hpp"
6  #include "UI/Spritesheet.hpp"
7  #include "UI/Textbox.hpp"
8  #include "UI/Status.hpp"
9  #include "UI/Data.hpp"
10
11 class Game {
12 public:
13     Game(sf::RenderWindow &window, network::socket &socket, uint8_t &status);
14     ~Game() = default;
15
16     void gameScreen();
17
18     void getServState();
19
20     void parseData(const std::string &str);
21
22     void removeKilled(std::vector<size_t> id);
23
24     void checkInput();
25
26     void gameEvents();
27
28     void eventsButton();
29
30     void eventsPressedButton();
31
32     void animSprites();
33
34     void displayGame();
35
36 private:
37     sf::RenderWindow &m_window;
38     network::socket &m_socket;
39     uint8_t &m_status;
40     Spritesheet m_map{Data::tmap, {0, 0}, 1023, {6016, 1080}, {0, 0, 1920, 1080}, {1, 1}, 4};
41     sf::Event m_event{};
42     sf::Clock m_clock{};
43     sf::Clock m_clockinputs{};
44     sf::Clock m_clockshots{};
45     sf::Clock m_clockanim{};
46     sf::Cursor m_cursor{};
47     Spritesheet m_lobbybutton{Data::tlobbybutton, {571, 880}, 2, {540, 135}, {0, 0, 270, 135}, {0.7,
48 0.5}, 270};
49     Spritesheet m_disconnectbutton{Data::tdisconnectbuttongame, {1160, 880}, 2, {540, 135}, {0, 0,
50 270, 135}, {0.7, 0.5}, 270};
51     sf::Color m_grey{169, 169, 169, 255};
52     sf::Color m_yellow{252, 245, 148, 255};
53     sf::Color m_red{255, 80, 37, 255};
54     Textbox m_lobbytext{30, "Lobby", m_yellow, {621, 893}};
55     Textbox m_disconnecttext{24, "Disconnect", m_red, {1190, 898}};
56     Textbox m_deadtext{40, "You are dead!", m_red, {825, 214}};
57     sf::RectangleShape m_deadrectangle{{400, 80}};
58     bool m_isdead{};
59     ecs::sparse_array<Spritesheet> m_display{};
60 };

```

5.23 Inputbox.hpp

```

1 #pragma once
2
3 #include <iostream>
4 #include <SFML/Graphics.hpp>
5
6 constexpr auto DELETE_KEY = 8;
7 constexpr auto ESCAPE_KEY = 27;
8
9 class Inputbox : public sf::Drawable {
10 public:
11     Inputbox(int size = 15, sf::Color color = sf::Color::White, bool select = false, size_t lim = 0);
12     ~Inputbox() = default;
13
14     void inputLogic(char c);
15
16     void setFont(const sf::Font &font);
17
18     void setPosition(sf::Vector2f pos);
19
20     void setLimit(size_t lim);
21
22     void setSelected(bool sel);
23
24     void setString(sf::String str);
25
26     sf::String &getText();
27
28     const sf::String &getText() const;
29
30     bool &getSelected();
31
32     sf::Text &getInputbox();
33
34     void updateString();
35
36     void draw(sf::RenderTarget &target, sf::RenderStates states) const override;
37
38 private:
39     sf::Text m_inputbox{};
40     sf::String m_text{};
41     sf::Vector2f m_pos{};
42     bool m_selected = false;
43     size_t m_limit = 0;
44 };

```

5.24 Lobby.hpp

```

1 #pragma once
2
3 #include <mutex>
4 #include <SFML/Graphics.hpp>
5 #include "UI/Spritesheet.hpp"
6 #include "UI/Profile.hpp"
7 #include "UI/Data.hpp"
8 #include "Network/Socket.hpp"
9
10 class Lobby {
11 public:
12     Lobby(sf::RenderWindow &window, network::socket &socket, uint8_t &status);
13     ~Lobby();
14
15     void initLobby();
16
17     void endLobby();
18
19     void lobbyScreen();
20
21     void eventsLobby();
22
23     void eventsReady();
24
25     void eventsDisconnect();
26
27     void eventsPressButton();
28
29     void reqLobby();
30
31     void parseData(const std::string &data);
32
33     void displayLobby();
34 };

```

```

79     private:
80         sf::RenderWindow &m_window;
81         network::socket &m_socket;
82         uint8_t &m_status;
83         Spritesheet m_menu{Data::tmenu, {0, 0}, 125, {3600, 10125}, {0, 0, 720, 405}, {2.667, 2.667},
720};
84         Spritesheet m_readybutton{Data::treadybutton, {870, 760}, 4, {540, 270}, {0, 0, 270, 135}, {0.7,
0.5}, 270};
85         Spritesheet m_disconnectbutton{Data::tdisconnectbutton, {540, 760}, 2, {540, 135}, {0, 0, 270,
135}, {0.7, 0.5}, 270};
86         sf::Color m_red{255, 80, 37, 255};
87         sf::Color m_green{59, 184, 115, 255};
88         Textbox m_lobbytext{60, "Lobby", sf::Color::Black, {880, 150}};
89         Textbox m_readytext{30, "Ready", m_red, {920, 773}};
90         Textbox m_disconnecttext{24, "Disconnect", sf::Color::Black, {570, 778}};
91         sf::RectangleShape m_lobbyrectangle{{1000, 750}};
92         sf::Color m_grey{180, 180, 180, 240};
93         sf::Event m_event{};
94         sf::Cursor m_cursor{};
95         sf::Clock m_clock{};
96         std::vector<Profile> m_profiles{};
97         std::unique_ptr<std::thread> m_thread{};
98         std::mutex m_lobbylock{};
99         bool m_ready{};
100         bool m_lobbyloop = true;
101 };

```

5.25 Menu.hpp

```

1  #pragma once
2
3  #include <SFML/Graphics.hpp>
4  #include <SFML/Audio.hpp>
5  #include "Network/Socket.hpp"
6  #include "UI/Inputbox.hpp"
7  #include "UI/Textbox.hpp"
8  #include "UI/Spritesheet.hpp"
9  #include "UI/Data.hpp"
10
11 class Menu {
12     public:
13         Menu(sf::RenderWindow &window, network::socket &socket, uint8_t &status);
14         ~Menu() = default;
15
16         void initMenu();
17
18         void loginScreen();
19
20         void eventsInputbox();
21
22         void checkInputboxSelected();
23
24         void eventsMenu();
25
26         void eventsLogin();
27
28         void eventsUser();
29
30         void tryLogin();
31
32         void tryUsername();
33
34         void displayMenu();
35
36         void drawUser();
37
38     private:
39         sf::RenderWindow &m_window;
40         network::socket &m_socket;
41         uint8_t &m_status;
42         Inputbox m_ip{20, sf::Color::Black, true, 15};
43         Inputbox m_port{20, sf::Color::Black, false, 7};
44         Inputbox m_user{20, sf::Color::Black, false, 25};
45         sf::Color m_black{0, 0, 0, 200};
46         sf::Color m_grey{180, 180, 180, 230};
47         sf::RectangleShape m_loginrectangle{{400, 450}};
48         sf::RectangleShape m_userrectangle{{400, 250}};
49         sf::RectangleShape m_ipbox{{330, 25}};
50         sf::RectangleShape m_portbox{{330, 25}};
51         sf::RectangleShape m_userbox{{330, 25}};
52         sf::RectangleShape m_errorbox{{400, 60}};
53         Textbox m_texttitle{80, "R-Type", sf::Color::White, {110, 280}};
54         Textbox m_textlogin{20, "Server login", sf::Color::White, {90, 410}};

```

```

104     Textbox m_textip{17, "IP address", sf::Color::White, {80, 470}};
105     Textbox m_textport{17, "Port", sf::Color::White, {80, 545}};
106     Textbox m_textuser{30, "Enter a username", sf::Color::Black, {840, 450}};
107     Textbox m_texterror{20, "Error: IP address or Port is incorrect", sf::Color::Yellow, {850,
900}};
108     Spritesheet m_menu{Data::tmenu, {0, 0}, 125, {3600, 10125}, {0, 0, 720, 405}, {2.667, 2.667},
720}};
109     Spritesheet m_loginbutton{Data::tloginbutton, {300, 640}, 3, {83, 75}, {0, 0, 83, 25}, {1.3,
1.3}, 83}};
110     Spritesheet m_userbutton{Data::tloginbutton, {910, 600}, 3, {83, 75}, {0, 0, 83, 25}, {1.3,
1.3}, 83}};
111     Spritesheet m_loading{Data::tloading, {0, 0}, 12, {612, 51}, {0, 0, 51, 51}, {0.4, 0.4}, 51}};
112     sf::Event m_event{};
113     sf::Font m_font{};
114     sf::Clock m_clock{};
115     sf::Cursor m_cursor{};
116     uint8_t m_logstatus = 0;
117     bool m_isloading{};
118 };

```

5.26 Profile.hpp

```

1 #pragma once
2
3 #include <SFML/Graphics.hpp>
4 #include "UI/Spritesheet.hpp"
5 #include "UI/Textbox.hpp"
6 #include "UI/Data.hpp"
7
8 class Profile : public sf::Drawable {
9     public:
10         Profile(std::string username, bool rdy, size_t nb);
11         ~Profile() = default;
12
13         void setPosition(sf::Vector2f pos);
14
15         void draw(sf::RenderTarget &target, sf::RenderStates states) const override;
16
17     private:
18         Spritesheet m_profilepic{Data::tprofile, {0, 0}, 1, {300, 300}, {0, 0, 300, 300}, {0.2, 0.2}};
19         Spritesheet m_ready{Data::tready, {0, 0}, 2, {248, 124}, {0, 0, 124, 124}, {0.5, 0.5}, 124}};
20         Textbox m_usernamebox{30, "", sf::Color::Black};
21         sf::Color m_grey{230, 230, 230, 255};
22         sf::RectangleShape m_profilerectangle{{800, 70}};
23         sf::Vector2f m_pos{};
24 };

```

5.27 Spritesheet.hpp

```

1 #pragma once
2
3 #include <SFML/Graphics.hpp>
4
5 class Spritesheet : public sf::Drawable {
6     public:
7         Spritesheet(sf::Texture &text, sf::Vector2f pos = {0, 0}, size_t nb = 1, sf::Vector2f dim = {1920,
1080}, sf::IntRect rect = {0, 0, 1920, 1080}, sf::Vector2f scale = {1, 1}, int space = 0);
8         ~Spritesheet() = default;
9
10         void setNbSprite(size_t nb);
11
12         void setSizeImage(sf::Vector2f size);
13
14         void setTexture(const sf::Texture &text);
15
16         void setPosition(sf::Vector2f pos);
17
18         void setRectSize(sf::IntRect rect);
19
20         void setRect(sf::IntRect rect);
21
22         void setSpace(int space);
23
24         void setScale(sf::Vector2f scale);
25
26         size_t &getCurrentImage();
27
28         size_t &getNbSprite();
29

```

```

85         sf::Vector2f &getScale();
86
92         sf::IntRect &getRect();
93
99         sf::Vector2f &getSize();
100
106         int &getSpace();
107
113         sf::Vector2f &getPosition();
114
120         const sf::Texture &getTexture();
121
130         bool mouseIsInSprite(sf::Vector2i pmouse);
131
137         void playImage(size_t nb);
138
143         void animSprite();
144
151         void draw(sf::RenderTarget &target, sf::RenderStates states) const override;
152
153     private:
154         sf::Sprite m_sprite{};
155         sf::Vector2f m_pos{};
156         sf::IntRect m_rect{};
157         sf::Vector2f m_scale{};
158         sf::Vector2f m_size{};
159         int m_space = 0;
160         size_t m_nb = 0;
161         size_t m_current = 0;
162
163 };

```

5.28 SpriteSize.hpp

```

1 #pragma once
2
3 #include <map>
4 #include <string>
5 #include <utility>
6
7 struct vec2float
8 {
9     float x, y;
10 };
11
12 namespace Data {
13     inline std::map<std::string, std::pair<vec2float, vec2float> > sprite_data {
14         {"player", {{32, 14}, {2.5, 2.5}}},
15         {"player0", {{32, 14}, {2.5, 2.5}}},
16         {"player1", {{32, 14}, {2.5, 2.5}}},
17         {"player2", {{32, 14}, {2.5, 2.5}}},
18         {"player3", {{32, 14}, {2.5, 2.5}}},
19         {"enemy_plane", {{21, 24}, {2.5, 2.5}}},
20         {"enemy_cyborg", {{32, 31}, {3, 3}}},
21         {"enemy2", {{32, 22}, {2, 2}}},
22         {"friend_shoot", {{16, 4}, {2, 2}}},
23         {"enemy_shoot", {{7, 6}, {2, 2}}}
24     };
25 }

```

5.29 Status.hpp

```

1 #pragma once
2
3 #include <cstdint>
4
5 enum status : uint8_t {
6     MENU,
7     LOBBY,
8     GAME,
9     EXIT
10 };

```

5.30 Textbox.hpp

```

1 #pragma once

```

```
2
3 #include <SFML/Graphics.hpp>
4
5 class Textbox : public sf::Drawable {
6     public:
7         Textbox(size_t size = 20, sf::String str = "", sf::Color color = sf::Color::White, sf::Vector2f
            pos = {0, 0});
8         ~Textbox() = default;
9
10        void setSize(size_t size);
11
12        void setString(sf::String str);
13
14        void setColor(sf::Color color);
15
16        void setPosition(sf::Vector2f pos);
17
18        void setOutlineColor(sf::Color color, float size);
19
20        void setFont(const sf::Font &font);
21
22        sf::String &getString(void);
23
24        sf::Vector2f &getPosition(void);
25
26        void draw(sf::RenderTarget &target, sf::RenderStates states) const override;
27
28    private:
29        sf::Text m_text{};
30        sf::String m_str{};
31        sf::Vector2f m_pos{};
32};
```


Index

- add
 - thread_pool, [49](#)
- add_system
 - ecs::registry, [25](#)
- allow_new_connections
 - network::server, [29](#)
- close
 - DyLib, [9](#)
- connect
 - network::socket, [31](#)
- Core, [7](#)
- Data::MissingAsset, [21](#)
- destroy
 - ecs::registry, [25](#)
- disconnect
 - network::socket, [31](#)
- draw
 - Inputbox, [16](#)
 - Profile, [24](#)
 - Spritesheet, [37](#)
 - Textbox, [45](#)
- DyLib, [7](#)
 - close, [9](#)
 - DyLib, [8](#)
 - getFunction, [9](#)
 - getVariable, [9](#)
 - open, [11](#)
- DyLib::exception, [11](#)
- DyLib::handle_error, [15](#)
- DyLib::symbol_error, [44](#)
- ecs::component::position, [23](#)
- ecs::exception, [12](#)
- ecs::module, [22](#)
 - get_label, [22](#)
 - get_system, [22](#)
 - spawn, [22](#)
- ecs::registry, [24](#)
 - add_system, [25](#)
 - destroy, [25](#)
 - emplace_component, [25](#)
 - get_components, [26](#)
 - get_entities, [26](#)
 - has_component, [27](#)
 - insert_component, [27](#)
 - kill_entity, [27](#)
 - register_component, [28](#)
 - remove_component, [28](#)
 - run_systems, [28](#)
 - spawn_entity, [29](#)
- ecs::sparse_array< T >, [33](#)
 - emplace_at, [33](#)
 - erase, [34](#)
 - get_index, [34](#)
 - insert_at, [34, 36](#)
- emplace_at
 - ecs::sparse_array< T >, [33](#)
- emplace_component
 - ecs::registry, [25](#)
- erase
 - ecs::sparse_array< T >, [34](#)
- Game, [13](#)
 - parseData, [13](#)
 - removeKilled, [13](#)
- get_clients
 - network::server, [30](#)
- get_components
 - ecs::registry, [26](#)
- get_entities
 - ecs::registry, [26](#)
- get_index
 - ecs::sparse_array< T >, [34](#)
- get_label
 - ecs::module, [22](#)
- get_packets
 - network::socket, [31](#)
- get_remote_ip
 - network::socket, [31](#)
- get_remote_port
 - network::socket, [32](#)
- get_system
 - ecs::module, [22](#)
- getCurrentImage
 - Spritesheet, [38](#)
- getFunction
 - DyLib, [9](#)
- getInputbox
 - Inputbox, [17](#)
- getNbSprite
 - Spritesheet, [38](#)
- getPosition
 - Spritesheet, [38](#)
 - Textbox, [45](#)
- getRect
 - Spritesheet, [38](#)
- getScale
 - Spritesheet, [39](#)

- getSelected
 - Inputbox, 17
- getSize
 - Spritesheet, 39
- getSpace
 - Spritesheet, 39
- getString
 - Textbox, 46
- getText
 - Inputbox, 17
- getTexture
 - Spritesheet, 39
- getVariable
 - DyLib, 9
- has_component
 - ecs::registry, 27
- include/DyLib.hpp, 51, 52
- include/ECS/Components.hpp, 54
- include/ECS/Entity.hpp, 54
- include/ECS/Exception.hpp, 55
- include/ECS/Module.hpp, 55
- include/ECS/Registry.hpp, 55
- include/ECS/SparseArray.hpp, 57
- include/ECS/Systems.hpp, 58
- include/Network/Client.hpp, 60
- include/Network/Header.hpp, 60
- include/Network/Packet.hpp, 60
- include/Network/Server.hpp, 60
- include/Network/Socket.hpp, 62
- include/Network/Type.hpp, 64
- include/Random.hpp, 64
- include/ThreadPool.hpp, 64
- include/Tools.hpp, 65
- include/TypeChecker.hpp, 65
- include/UI/Core.hpp, 65
- include/UI/Data.hpp, 66
- include/UI/Game.hpp, 67
- include/UI/Inputbox.hpp, 68
- include/UI/Lobby.hpp, 68
- include/UI/Menu.hpp, 69
- include/UI/Profile.hpp, 70
- include/UI/Spritesheet.hpp, 70
- include/UI/SpriteSize.hpp, 71
- include/UI/Status.hpp, 71
- include/UI/Textbox.hpp, 71
- Inputbox, 16
 - draw, 16
 - getInputbox, 17
 - getSelected, 17
 - getText, 17
 - inputLogic, 18
 - setFont, 18
 - setLimit, 18
 - setPosition, 19
 - setSelected, 19
 - setString, 19
- inputLogic
 - Inputbox, 18
- insert_at
 - ecs::sparse_array< T >, 34, 36
- insert_component
 - ecs::registry, 27
- kill_entity
 - ecs::registry, 27
- Lobby, 19
 - parseData, 20
- Menu, 20
 - mouelsInSprite
 - Spritesheet, 40
- network::client, 7
- network::header, 15
- network::packet, 23
- network::server, 29
 - allow_new_connections, 29
 - get_clients, 30
 - receive, 30
 - send, 30
- network::socket, 30
 - connect, 31
 - disconnect, 31
 - get_packets, 31
 - get_remote_ip, 31
 - get_remote_port, 32
 - pop_packets, 32
 - send, 32
- open
 - DyLib, 11
- parseData
 - Game, 13
 - Lobby, 20
- playImage
 - Spritesheet, 40
- pop_packets
 - network::socket, 32
- Profile, 23
 - draw, 24
 - setPosition, 24
- receive
 - network::server, 30
- register_component
 - ecs::registry, 28
- release
 - thread_pool, 49
- remove
 - thread_pool, 49
- remove_component
 - ecs::registry, 28
- removeKilled
 - Game, 13
- run_systems

- ecs::registry, 28
- send
 - network::server, 30
 - network::socket, 32
- setColor
 - Textbox, 46
- setFont
 - Inputbox, 18
 - Textbox, 46
- setLimit
 - Inputbox, 18
- setNbSprite
 - Spritesheet, 40
- setOutlineColor
 - Textbox, 47
- setPosition
 - Inputbox, 19
 - Profile, 24
 - Spritesheet, 42
 - Textbox, 47
- setRect
 - Spritesheet, 42
- setRectSize
 - Spritesheet, 42
- setScale
 - Spritesheet, 42
- setSelected
 - Inputbox, 19
- setSize
 - Textbox, 47
- setSizeImage
 - Spritesheet, 43
- setSpace
 - Spritesheet, 43
- setString
 - Inputbox, 19
 - Textbox, 47
- setTexture
 - Spritesheet, 43
- spawn
 - ecs::module, 22
- spawn_entity
 - ecs::registry, 29
- Spritesheet, 36
 - draw, 37
 - getCurrentImage, 38
 - getNbSprite, 38
 - getPosition, 38
 - getRect, 38
 - getScale, 39
 - getSize, 39
 - getSpace, 39
 - getTexture, 39
 - mouselsInSprite, 40
 - playImage, 40
 - setNbSprite, 40
 - setPosition, 42
 - setRect, 42
 - setRectSize, 42
 - setScale, 42
 - setSizeImage, 43
 - setSpace, 43
 - setTexture, 43
- Textbox, 44
 - draw, 45
 - getPosition, 45
 - getString, 46
 - setColor, 46
 - setFont, 46
 - setOutlineColor, 47
 - setPosition, 47
 - setSize, 47
 - setString, 47
- thread_pool, 49
 - add, 49
 - release, 49
 - remove, 49
- vec2float, 50