R-Type

Generated by Doxygen 1.9.1

Chapter 1

Namespace Index

1.1 Namespace List

Here is a list of all namespaces with brief descriptions:

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

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

Chapter 3

Class Index

3.1 Class List

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

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

Chapter 5

Namespace Documentation

5.1 r_type Namespace Reference

Namespaces

- net

5.2 r_type::net Namespace Reference

Classes

- class AClient
- class Client
- class IClient
- class AServer

  AServer class template for managing server operations.
- class Server

Chapter 6

Class Documentation

6.1 AbstractScenes Class Reference

An abstract class that provides a base for managing different scenes in a game.

```
#include <a_scenes.hpp>
```

6.1.1 Detailed Description

An abstract class that provides a base for managing different scenes in a game.

This abstract class implements the ScenesInterface and provides some common functionality.

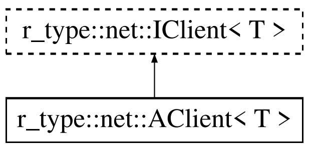
The documentation for this class was generated from the following file:

- /home/runner/work/R-Type/R-Type/ECS/Interface/Include/a_scenes.hpp

6.2 r_type::net::AClient< T > Class Template Reference

```
#include <a_client.hpp>
```

Inheritance diagram for r_type::net::AClient< T >:



Public Member Functions

- AClient ()
- virtual ∼AClient ()
- bool Connect (const std::string &host, const uint16_t port)

    Connects to a remote host using UDP protocol.
- void Disconnect ()

    Disconnects the client from the server.
- bool IsConnected ()

    Checks if the client is connected to the server.
- void Send (const Message< T > &msg)

    Send message to server.
- ThreadSafeQueue< OwnedMessage< T > > & Incoming ()

    get incoming messages
- const std::unique_ptr< Connection< T > > & getConnection ()
- void setPlayerId (int id)
- uint32_t getPlayerId ()
- void addEntity (EntityInformation entity, ComponentManager &componentManager, TextureManager &textureManager)
- void removeEntity (int entityId, ComponentManager &componentManager)
- void updateEntity (EntityInformation entity, ComponentManager &componentManager)

Protected Attributes

- asio::io_context m_context
- std::thread thrContext
- std::unique_ptr< Connection< T > > m_connection

Private Attributes

- ThreadSafeQueue< OwnedMessage< T > > m_qMessagesIn
- uint32_t playerId = 0

6.2.1 Constructor & Destructor Documentation

6.2.1.1 AClient()

```
template<typename T >
r_type::net::AClient< T >::AClient ( )  [inline]
```

6.2.1.2 ∼AClient()

```
template<typename T >
virtual r_type::net::AClient< T >::∼AClient ( )  [inline], [virtual]
```

6.2.2 Member Function Documentation

6.2.2.1 addEntity()

```
template<typename T >
void r_type::net::AClient< T >::addEntity (
            EntityInformation entity,
            ComponentManager & componentManager,
            TextureManager & textureManager )
```

6.2.2.2 Connect()

```
template<typename T >
bool r_type::net::AClient< T >::Connect (
            const std::string & host,
            const uint16_t port )  [inline], [virtual]
```

Connects to a remote host using UDP protocol.

Parameters

host	The IP address or hostname of the remote host.
port	The port number of the remote host.

Returns

true if the connection is successful, false otherwise.

Implements r_type::net::IClient< T >.

6.2.2.3 Disconnect()

```
template<typename T >
void r_type::net::AClient< T >::Disconnect ( )  [inline], [virtual]
```

Disconnects the client from the server.

This function disconnects the client from the server if it is currently connected. It stops the context and joins the context thread. It also releases the connection resource.

Implements r_type::net::IClient< T >.

6.2.2.4 getConnection()

```
template<typename T >
const std::unique_ptr<Connection<T> >& r_type::net::AClient< T >::getConnection ( )  [inline]
```

6.2.2.5 getPlayerId()

```
template<typename T >
uint32_t r_type::net::AClient< T >::getPlayerId ( )  [inline]
```

6.2.2.6 Incoming()

```
template<typename T >
ThreadSafeQueue<OwnedMessage<T> >& r_type::net::AClient< T >::Incoming ( )  [inline], [virtual]
```

get incoming messages

Returns

ThreadSafeQueue<OwnedMessage<T>>&

Implements r_type::net::IClient< T >.

6.2.2.7 IsConnected()

```
template<typename T >
bool r_type::net::AClient< T >::IsConnected ( )  [inline], [virtual]
```

Checks if the client is connected to the server.

Returns

true

false

Implements r_type::net::IClient< T >.

6.2.2.8 removeEntity()

```
template<typename T >
void r_type::net::AClient< T >::removeEntity (
             int entityId,
             ComponentManager & componentManager )
```

6.2.2.9 Send()

```
template<typename T >
void r_type::net::AClient< T >::Send (
             const Message< T > & msg )  [inline], [virtual]
```

Send message to server.

Parameters

msg	

Implements r_type::net::IClient< T >.

6.2.2.10 setPlayerId()

```
template<typename T >
void r_type::net::AClient< T >::setPlayerId (
             int id )  [inline]
```

6.2.2.11 updateEntity()

```
template<typename T >
void r_type::net::AClient< T >::updateEntity (
             EntityInformation entity,
             ComponentManager & componentManager )
```

6.2.3 Member Data Documentation

6.2.3.1 m_connection

```
template<typename T >
std::unique_ptr<Connection<T> > r_type::net::AClient< T >::m_connection  [protected]
```

6.2.3.2 m_context

```
template<typename T >
asio::io_context r_type::net::AClient< T >::m_context  [protected]
```

6.2.3.3 m_qMessagesIn

```
template<typename T >
ThreadSafeQueue<OwnedMessage<T> > r_type::net::AClient< T >::m_qMessagesIn  [private]
```

6.2.3.4 playerId

```
template<typename T >
uint32_t r_type::net::AClient< T >::playerId = 0  [private]
```

6.2.3.5 thrContext

```
template<typename T >
std::thread r_type::net::AClient< T >::thrContext  [protected]
```

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

- /home/runner/work/R-Type/R-Type/Client/Interface/Include/Net/a_client.hpp

6.3 AllyComponent Struct Reference

```
#include <ally_component.hpp>
```

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

- /home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/ally_component.hpp

6.4 AllyMissileComponent Struct Reference

```
#include <ally_missile_component.hpp>
```

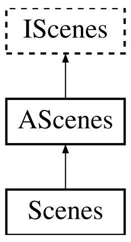
The documentation for this struct was generated from the following file:

- /home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/ally_missile_component.hpp

6.5 AScenes Class Reference

```
#include <a_scenes.hpp>
```

Inheritance diagram for AScenes:



Public Types

- enum class Scene {
  MAIN_MENU , GAME_LOOP , SETTINGS_MENU , IN_GAME_MENU ,
  EXIT }

  Represents the different scenes in the R-Type client application.
- enum class GameMode { EASY , MEDIUM , HARD }

  Enumeration to represent different game difficulty levels.
- enum class DaltonismMode { NORMAL , TRITANOPIA , DEUTERANOPIA , PROTANOPIA }

  Enum representing different modes of color blindness (Daltonism).
- enum class Actions {
  UP , DOWN , LEFT , RIGHT ,
  FIRE , PAUSE , QUIT }

  Enumeration representing possible actions in the game.

Public Member Functions

- AScenes (sf::RenderWindow ∗window)
- ∼AScenes ()=default
- void setScene (Scene scene)

  Set the Scene object.
- AScenes::Scene getPreviousScene ()

  Get the Previous Scene object.
- void setDaltonism (DaltonismMode const mode)

  Set the Daltonism object.
- void setGameMode (GameMode const mode)

  Set the Game Mode object.
- void setDisplayDaltonismChoice (bool const displayDaltonismChoice)
- bool getDisplayDaltonismChoice () const
- void setDisplayGameModeChoice (bool const displayGameModeChoice)
- bool getDisplayGameModeChoice () const
- void setDisplayKeyBindsChoice (bool const displayKeyBindsChoice)
- bool getDisplayKeyBindsChoice () const

Public Attributes

- std::map< Actions, sf::Keyboard::Key > keyBinds

  A map that binds game actions to specific keyboard keys.

Protected Attributes

- sf::RenderWindow ∗ _window
- GameMode _currentGameMode = GameMode::MEDIUM
- DaltonismMode _currentDaltonismMode = DaltonismMode::NORMAL
- Scene _currentScene = Scene::MAIN_MENU
- Scene _previousScene = Scene::MAIN_MENU
- std::vector< std::shared_ptr< Entity > > buttons
- bool _displayDaltonismChoice = false
- bool _displayGameModeChoice = false
- bool _displayKeyBindsChoice = false

6.5.1 Member Enumeration Documentation

6.5.1.1 Actions

`enum AScenes::Actions [strong]`

Enumeration representing possible actions in the game.

This enumeration defines the various actions that can be performed by the player in the game. The actions include:

- UP: Move up
- DOWN: Move down
- LEFT: Move left
- RIGHT: Move right
- FIRE: Fire a weapon
- PAUSE: Pause the game
- QUIT: Quit the game

Enumerator

UP	
DOWN	
LEFT	
RIGHT	
FIRE	
PAUSE	
QUIT	

6.5.1.2 DaltonismMode

`enum AScenes::DaltonismMode [strong]`

Enum representing different modes of color blindness (Daltonism).

This enum is used to specify the type of color blindness mode that can be applied.

Enumerator

NORMAL	Represents normal vision without any color blindness.
TRITANOPIA	Represents Tritanopia, a type of color blindness where blue and yellow colors are confused.
DEUTERANOPIA	Represents Deuteranopia, a type of color blindness where green and red colors are confused.
PROTANOPIA	Represents Protanopia, a type of color blindness where red and green colors are confused.

6.5.1.3 GameMode

```
enum AScenes::GameMode  [strong]
```

Enumeration to represent different game difficulty levels.

This enumeration defines the various difficulty levels that can be selected in the game. The available modes are:

- EASY: Represents an easy difficulty level.
- MEDIUM: Represents a medium difficulty level.
- HARD: Represents a hard difficulty level.

Enumerator

EASY	
MEDIUM	
HARD	

6.5.1.4 Scene

```
enum AScenes::Scene  [strong]
```

Represents the different scenes in the R-Type client application.

This enumeration defines the various scenes that the client can be in during its lifecycle.

Enumerator

MAIN_MENU	Represents the main menu scene.
GAME_LOOP	Represents the game loop scene where the main gameplay occurs.
SETTINGS_MENU	Represents the settings menu scene where the user can adjust settings.
IN_GAME_MENU	Represents the in-game menu scene that can be accessed during gameplay.
EXIT	Represents the exit scene where the application is closing.

6.5.2 Constructor & Destructor Documentation

6.5.2.1 AScenes()

```
AScenes::AScenes (
             sf::RenderWindow * window )
```

6.5.2.2 ∼AScenes()

```
AScenes::∼AScenes ( )  [default]
```

6.5.3 Member Function Documentation

6.5.3.1 getDisplayDaltonismChoice()

```
bool AScenes::getDisplayDaltonismChoice ( ) const
```

6.5.3.2 getDisplayGameModeChoice()

```
bool AScenes::getDisplayGameModeChoice ( ) const
```

6.5.3.3 getDisplayKeyBindsChoice()

```
bool AScenes::getDisplayKeyBindsChoice ( ) const
```

6.5.3.4 getPreviousScene()

```
AScenes::Scene AScenes::getPreviousScene ( )
```

Get the Previous Scene object.

Returns

Scene

6.5.3.5 setDaltonism()

```
void AScenes::setDaltonism (
            DaltonismMode const mode )
```

Set the Daltonism object.

Parameters

mode	The daltonism mode to set

6.5.3.6 setDisplayDaltonismChoice()

```
void AScenes::setDisplayDaltonismChoice (
             bool const displayDaltonismChoice )
```

6.5.3.7 setDisplayGameModeChoice()

```
void AScenes::setDisplayGameModeChoice (
             bool const displayGameModeChoice )
```

6.5.3.8 setDisplayKeyBindsChoice()

```
void AScenes::setDisplayKeyBindsChoice (
             bool const displayKeyBindsChoice )
```

6.5.3.9 setGameMode()

```
void AScenes::setGameMode (
             GameMode const mode )
```

Set the Game Mode object.

Parameters

mode	

6.5.3.10 setScene()

```
void AScenes::setScene (
             AScenes::Scene scene )
```

Set the Scene object.

Parameters

scene	

6.5.4 Member Data Documentation

6.5.4.1 _currentDaltonismMode

```
DaltonismMode AScenes::_currentDaltonismMode = DaltonismMode::NORMAL  [protected]
```

6.5.4.2 _currentGameMode

```
GameMode AScenes::_currentGameMode = GameMode::MEDIUM  [protected]
```

6.5.4.3 _currentScene

```
Scene AScenes::_currentScene = Scene::MAIN_MENU  [protected]
```

6.5.4.4 _displayDaltonismChoice

```
bool AScenes::_displayDaltonismChoice = false  [protected]
```

6.5.4.5 _displayGameModeChoice

```
bool AScenes::_displayGameModeChoice = false  [protected]
```

6.5.4.6 _displayKeyBindsChoice

```
bool AScenes::_displayKeyBindsChoice = false  [protected]
```

6.5.4.7 _previousScene

```
Scene AScenes::_previousScene = Scene::MAIN_MENU  [protected]
```

6.5.4.8 _window

```
sf::RenderWindow* AScenes::_window  [protected]
```

6.5.4.9 buttons

```
std::vector<std::shared_ptr<Entity> > AScenes::buttons  [protected]
```

6.5.4.10 keyBinds

```
std::map<Actions, sf::Keyboard::Key> AScenes::keyBinds
```

Initial value:
```
= {{Actions::UP, sf::Keyboard::Key::Up},
        {Actions::DOWN, sf::Keyboard::Key::Down}, {Actions::LEFT, sf::Keyboard::Key::Left},
        {Actions::RIGHT, sf::Keyboard::Key::Right}, {Actions::FIRE, sf::Keyboard::Key::Space},
        {Actions::PAUSE, sf::Keyboard::Key::Escape}, {Actions::QUIT, sf::Keyboard::Key::Q}}
```

A map that binds game actions to specific keyboard keys.

This map associates each action defined in the Actions enum with a corresponding key from the sf::Keyboard::Key enumeration. It is used to handle user input by mapping key presses to game actions.

The key bindings are as follows:

- Actions::UP -> sf::Keyboard::Key::Up
- Actions::DOWN -> sf::Keyboard::Key::Down
- Actions::LEFT -> sf::Keyboard::Key::Left
- Actions::RIGHT -> sf::Keyboard::Key::Right
- Actions::FIRE -> sf::Keyboard::Key::Space
- Actions::PAUSE -> sf::Keyboard::Key::Escape
- Actions::QUIT -> sf::Keyboard::Key::Q

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

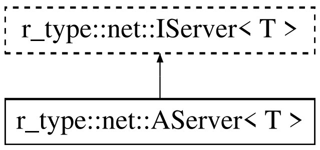
- /home/runner/work/R-Type/R-Type/ECS/Interface/Include/a_scenes.hpp
- /home/runner/work/R-Type/R-Type/ECS/Src/a_scenes.cpp

6.6 r_type::net::AServer< T > Class Template Reference

AServer class template for managing server operations.

```
#include <a_server.hpp>
```

Inheritance diagram for r_type::net::AServer< T >:



Public Member Functions

- AServer (uint16_t port)

  Constructs an AServer object with the specified port.
- ~AServer ()

  Destructor for the AServer class.
- bool Start ()

  Start the server.
- void Stop ()

  Stops the server.
- void WaitForClientMessage ()

  Waits for a client message asynchronously.
- void MessageClient (std::shared_ptr< Connection< T >> client, const Message< T > &msg)

  Sends a message to a specific client if the client is connected.
- void MessageAllClients (const Message< T > &msg, std::shared_ptr< Connection< T >> pIgnore↩ Client=nullptr)

  Sends a message to all connected clients, optionally ignoring a specified client.
- void Update (size_t nMaxMessages=-1, bool bWait=false)

  Updates the server state and processes incoming messages.
- void UpdateEntityPosition (r_type::net::Message< T > &msg, uint32_t clientId)

  Updates the position of an entity based on the message received and the client ID.
- uint32_t GetClientEntityId (uint32_t id)

  Retrieves the entity ID associated with a client ID.
- void RemovePlayer (uint32_t id)

  Removes a player from the game based on the client ID.
- void RemoveEntities (uint32_t id)

  Removes entities associated with a player.
- EntityInformation InitiatePlayers (int clientId)

  Initializes a new player entity and assigns a random position.
- EntityInformation InitiateMissile (int clientId)

  Initializes a missile entity associated with a player.
- EntityInformation InitiateBackground ()

  Initializes a background entity.
- virtual void InitListEntities (std::shared_ptr< r_type::net::Connection< T >> client, u_int32_t entityID)=0

  Sends a list of existing entities to a newly connected client for initialization.
- virtual void OnClientValidated (std::shared_ptr< Connection< T >> client)

  Callback function that is called when a client has been successfully validated.

Public Attributes

- ThreadSafeQueue< OwnedMessage< T > > m_qMessagesIn

    Thread-safe queue to store incoming messages.
- std::deque< std::shared_ptr< Connection< T > > > m_deqConnections

    A deque that holds shared pointers to Connection objects.
- asio::io_context m_asioContext

    The io_context object provides I/O services, such as sockets, that the server will use.
- std::thread m_threadContext

    Thread object for managing the server's context operations.
- asio::ip::udp::socket m_asioSocket

    A socket for sending and receiving UDP datagrams.
- asio::ip::udp::endpoint m_clientEndpoint

    Represents the endpoint of a client in a UDP connection.
- std::array< uint8_t, 1024 > m_tempBuffer

    Temporary buffer used for storing data.
- uint32_t nIDCounter = 10000

    Counter for generating unique network IDs.
- ComponentManager componentManager

    Manages and maintains the lifecycle of various components within the server.
- EntityManager entityManager

    Manages the lifecycle and operations of entities within the server.
- EntityFactory entityFactory

    An instance of EntityFactory used to create and manage game entities.
- std::unordered_map< uint32_t, uint32_t > clientPlayerID

    A container that maps client IDs to player IDs.
- int nbrOfPlayers = 0

    Number of players currently connected to the server.
- std::chrono::system_clock::time_point _clock = std::chrono::system_clock::now()

    Stores the current time point from the system clock.
- EntityInformation background

    Holds information about the background entity.

Protected Member Functions

- virtual bool OnClientConnect (std::shared_ptr< Connection< T >> client)

    on client connect event
- virtual void OnClientDisconnect (std::shared_ptr< Connection< T >> client)

    on client disconnect event
- virtual void OnMessage (std::shared_ptr< Connection< T >> client, Message< T > &msg)

    on message event

6.6.1 Detailed Description

template<typename T>
class r_type::net::AServer< T >

AServer class template for managing server operations.

This class template provides a framework for creating and managing a server that handles client connections, messages, and entity updates. It uses the ASIO library for asynchronous network communication and provides various functions for server operations such as starting, stopping, and updating the server, as well as handling client messages and connections.

Template Parameters

T	The type of data that the server handles.

6.6.2 Constructor & Destructor Documentation

6.6.2.1 AServer()

```
template<typename T >
r_type::net::AServer< T >::AServer (
            uint16_t port ) [inline]
```

Constructs an AServer object with the specified port.

This constructor initializes the server with the given port number and sets up the necessary components for the server to function. It initializes the ASIO socket with the provided port and creates instances of EntityManager, EntityFactory, and ComponentManager. Additionally, it initiates the background process and creates three basic monster entities using the entity factory.

Parameters

port	The port number on which the server will listen for incoming connections.

6.6.2.2 ∼AServer()

```
template<typename T >
r_type::net::AServer< T >::∼AServer ( ) [inline]
```

Destructor for the AServer class.

This destructor ensures that the server is properly stopped by calling the Stop() method when an instance of AServer is destroyed.

6.6.3 Member Function Documentation

6.6.3.1 GetClientEntityId()

```
template<typename T >
uint32_t r_type::net::AServer< T >::GetClientEntityId (
            uint32_t id ) [inline]
```

Retrieves the entity ID associated with a client ID.

Parameters

id	The client ID.

Returns

uint32_t The entity ID associated with the client.

6.6.3.2 InitiateBackground()

```
template<typename T >
EntityInformation r_type::net::AServer< T >::InitiateBackground ( )  [inline]
```

Initializes a background entity.

The function creates and returns information about the background entity.

Returns

EntityInformation The information of the background entity.

6.6.3.3 InitiateMissile()

```
template<typename T >
EntityInformation r_type::net::AServer< T >::InitiateMissile (
             int clientId )  [inline]
```

Initializes a missile entity associated with a player.

The function creates a missile entity associated with a player and assigns its position based on the player's current position.

Parameters

client↩ Id	The client ID of the player firing the missile.

Returns

EntityInformation The information of the newly created missile entity.

6.6.3.4 InitiatePlayers()

```
template<typename T >
EntityInformation r_type::net::AServer< T >::InitiatePlayers (
             int clientId )  [inline]
```

Initializes a new player entity and assigns a random position.

The function creates a new player entity, assigns it a random position, and ensures that it does not overlap with any other players.

Parameters

client↩ Id	The client ID of the player being initialized.

Returns

EntityInformation The information of the newly created player entity.

6.6.3.5 InitListEntities()

```
template<typename T >
virtual void r_type::net::AServer< T >::InitListEntities (
             std::shared_ptr< r_type::net::Connection< T >> client,
             u_int32_t entityID )  [pure virtual]
```

Sends a list of existing entities to a newly connected client for initialization.

The function iterates through all existing entities and sends their information to the newly connected client, excluding specific entities such as the client itself.

Parameters

client	The connection to the client.
entityID	The ID of the entity to exclude (usually the client's own entity).

Implemented in r_type::net::Server.

6.6.3.6 MessageAllClients()

```
template<typename T >
void r_type::net::AServer< T >::MessageAllClients (
             const Message< T > & msg,
             std::shared_ptr< Connection< T >> pIgnoreClient = nullptr )  [inline]
```

Sends a message to all connected clients, optionally ignoring a specified client.

This function iterates through all the connections in the server and sends the provided message to each connected client, except for the client specified by `pIgnoreClient`. If a client is found to be disconnected, it triggers the disconnection handler and removes the client from the list of connections.

Template Parameters

T	The type of the message.

Parameters

msg	The message to be sent to all clients.
pIgnoreClient	A shared pointer to a client connection that should be ignored. Defaults to nullptr.

6.6.3.7 MessageClient()

```
template<typename T >
void r_type::net::AServer< T >::MessageClient (
             std::shared_ptr< Connection< T >> client,
             const Message< T > & msg )  [inline]
```

Sends a message to a specific client if the client is connected.

If the client is not connected, it handles the client disconnection.

Template Parameters

T	The type of the message.

Parameters

client	A shared pointer to the client connection.
msg	The message to be sent to the client.

6.6.3.8 OnClientConnect()

```
template<typename T >
virtual bool r_type::net::AServer< T >::OnClientConnect (
             std::shared_ptr< Connection< T >> client )  [inline], [protected], [virtual]
```

on client connect event

Parameters

client	

Returns

true

false

6.6.3.9 OnClientDisconnect()

```
template<typename T >
virtual void r_type::net::AServer< T >::OnClientDisconnect (
             std::shared_ptr< Connection< T >> client )  [inline], [protected], [virtual]
```

on client disconnect event

Parameters

client	

6.6.3.10 OnClientValidated()

```
template<typename T >
virtual void r_type::net::AServer< T >::OnClientValidated (
             std::shared_ptr< Connection< T >> client )  [inline], [virtual]
```

Callback function that is called when a client has been successfully validated.

This function is intended to be overridden by derived classes to handle any specific actions that need to be taken when a client is validated.

Parameters

client	A shared pointer to the validated client connection.

6.6.3.11 OnMessage()

```
template<typename T >
virtual void r_type::net::AServer< T >::OnMessage (
             std::shared_ptr< Connection< T >> client,
             Message< T > & msg )  [inline], [protected], [virtual]
```

on message event

Parameters

client	
msg	

6.6.3.12 RemoveEntities()

```
template<typename T >
void r_type::net::AServer< T >::RemoveEntities (
             uint32_t id )  [inline]
```

Removes entities associated with a player.

Parameters

id	The ID of the player whose entities are to be removed.

6.6.3.13 RemovePlayer()

```
template<typename T >
void r_type::net::AServer< T >::RemovePlayer (
             uint32_t id )  [inline]
```

Removes a player from the game based on the client ID.

Parameters

id	The client ID of the player to be removed.

6.6.3.14 Start()

```
template<typename T >
bool r_type::net::AServer< T >::Start ( )  [inline]
```

Start the server.

Returns

true

false

6.6.3.15 Stop()

```
template<typename T >
void r_type::net::AServer< T >::Stop ( )  [inline]
```

Stops the server.

This function stops the server by stopping the ASIO context and joining the thread context. It also prints a message indicating that the server has been stopped.

6.6.3.16 Update()

```
template<typename T >
void r_type::net::AServer< T >::Update (
            size_t nMaxMessages = -1,
            bool bWait = false )  [inline]
```

Updates the server state and processes incoming messages.

This function updates the state of entities on the server and processes incoming messages. It can optionally wait for messages and limit the number of messages processed in one call.

Parameters

nMaxMessages	The maximum number of messages to process in one call. Default is -1 (no limit).
bWait	If true, the function will wait for messages to be available before processing.

The function performs the following tasks:

- Updates the positions of entities based on their components.
- Sends updated entity information to all connected clients.
- Checks for collisions between player missiles and monsters, and handles entity destruction.
- Processes incoming messages from clients.

6.6.3.17 UpdateEntityPosition()

```
template<typename T >
void r_type::net::AServer< T >::UpdateEntityPosition (
            r_type::net::Message< T > & msg,
            uint32_t clientId )  [inline]
```

Updates the position of an entity based on the message received and the client ID.

This function updates the position of an entity. If the entity is not touching any other player, it updates its position and sends a message to all clients about the new position. If it touches another player, a destroy message is sent to all clients.

Parameters

msg	The message containing the new position of the entity.
client↩ Id	The ID of the client sending the update.

6.6.3.18 WaitForClientMessage()

```
template<typename T >
void r_type::net::AServer< T >::WaitForClientMessage ( )  [inline]
```

Waits for a client message asynchronously.

This function waits for a client message by asynchronously receiving data from the socket. When a message is received, it checks if the client endpoint protocol is UDPv4. If the protocol is not UDPv4, it recursively calls itself to wait for another client message. If the protocol is UDPv4 and there are no errors, it prints the client endpoint and checks if a connection already exists. If a connection already exists, it returns without further processing. If a connection does not exist, it creates a new client socket, binds it to a local endpoint, and creates a new connection object. It then calls the OnClientConnect function to check if the client connection is approved. If the connection is approved, it adds the new connection to the list of connections, connects it to the client, and prints the connection ID. If the connection is denied, it prints a message indicating the connection was denied. If there is an error during the receive operation, it prints the error message../

6.6.4 Member Data Documentation

6.6.4.1 _clock

```
template<typename T >
std::chrono::system_clock::time_point r_type::net::AServer< T >::_clock = std::chrono::system↩
_clock::now()
```

Stores the current time point from the system clock.

This variable is initialized with the current time using std::chrono::system_clock::now() and represents a specific point in time according to the system clock.

6.6.4.2 background

```
template<typename T >
EntityInformation r_type::net::AServer< T >::background
```

Holds information about the background entity.

This member variable stores the details related to the background entity in the game. It includes properties such as position, texture, and other relevant attributes that define the background's appearance and behavior.

6.6.4.3 clientPlayerID

```
template<typename T >
std::unordered_map<uint32_t, uint32_t> r_type::net::AServer< T >::clientPlayerID
```

A container that maps client IDs to player IDs.

left: client ID right: player ID

This unordered map is used to associate client IDs with their corresponding player IDs. The keys are o}f type uint32_t representing the client IDs, and the values are also of type uint32_t representing the player IDs.

6.6.4.4 componentManager

```
template<typename T >
ComponentManager r_type::net::AServer< T >::componentManager
```

Manages and maintains the lifecycle of various components within the server.

The ComponentManager is responsible for creating, updating, and destroying components as needed. It ensures that all components are properly managed and that their states are consistent throughout the server's operation.

6.6.4.5 entityFactory

```
template<typename T >
EntityFactory r_type::net::AServer< T >::entityFactory
```

An instance of EntityFactory used to create and manage game entities.

6.6.4.6 entityManager

```
template<typename T >
EntityManager r_type::net::AServer< T >::entityManager
```

Manages the lifecycle and operations of entities within the server.

The EntityManager is responsible for creating, updating, and deleting entities. It ensures that entities are properly managed and synchronized within the server's environment.

6.6.4.7 m_asioContext

```
template<typename T >
asio::io_context r_type::net::AServer< T >::m_asioContext
```

The io_context object provides I/O services, such as sockets, that the server will use.

This member variable is responsible for managing asynchronous I/O operations. It is part of the ASIO library, which is used for network programming.

6.6.4.8 m_asioSocket

```
template<typename T >
asio::ip::udp::socket r_type::net::AServer< T >::m_asioSocket
```

A socket for sending and receiving UDP datagrams.

This member variable represents a UDP socket using the ASIO library. It is used for network communication in the server.

6.6.4.9 m_clientEndpoint

```
template<typename T >
asio::ip::udp::endpoint r_type::net::AServer< T >::m_clientEndpoint
```

Represents the endpoint of a client in a UDP connection.

This member variable holds the endpoint information (IP address and port) of a client in a UDP connection using the ASIO library.

6.6.4.10 m_deqConnections

```
template<typename T >
std::deque<std::shared_ptr<Connection<T> > > r_type::net::AServer< T >::m_deqConnections
```

A deque that holds shared pointers to Connection objects.

This member variable is used to manage a collection of active connections. The use of std::shared_ptr ensures that the Connection objects are reference-counted and automatically deallocated when no longer in use.

Template Parameters

T	The type of data that the Connection handles.

6.6.4.11 m_qMessagesIn

```
template<typename T >
ThreadSafeQueue<OwnedMessage<T> > r_type::net::AServer< T >::m_qMessagesIn
```

Thread-safe queue to store incoming messages.

This member variable is a thread-safe queue that holds messages of type OwnedMessage<T>. It ensures that messages can be safely accessed and modified by multiple threads concurrently.

6.6.4.12 m_tempBuffer

```
template<typename T >
std::array<uint8_t, 1024> r_type::net::AServer< T >::m_tempBuffer
```

Temporary buffer used for storing data.

This buffer is an array of 1024 bytes (uint8_t) used for temporary storage of data within the server's network interface.

6.6.4.13 m_threadContext

```
template<typename T >
std::thread r_type::net::AServer< T >::m_threadContext
```

Thread object for managing the server's context operations.

This member variable represents a thread that handles the server's context, allowing for concurrent execution of tasks related to the server's operation. It is used to ensure that the server can perform its duties without blocking the main execution flow.

6.6.4.14 nbrOfPlayers

```
template<typename T >
int r_type::net::AServer< T >::nbrOfPlayers = 0
```

Number of players currently connected to the server.

6.6.4.15 nIDCounter

```
template<typename T >
uint32_t r_type::net::AServer< T >::nIDCounter = 10000
```

Counter for generating unique network IDs.

This variable is used to keep track of the current ID to be assigned for network-related entities. It starts at 10000 and increments with each new ID generation.

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

- /home/runner/work/R-Type/R-Type/Server/Interface/Include/Net/a_server.hpp

6.7 BackgroundComponent Struct Reference

```
#include <background_component.hpp>
```

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

- /home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/background_component.hpp

6.8 BasicMonsterComponent Struct Reference

```
#include <basic_monster_component.hpp>
```

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

- /home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/basic_monster_component.hpp

6.9 BindComponent Struct Reference

```
#include <bind_component.hpp>
```

Public Member Functions

- BindComponent (std::function< IScenes ∗(AScenes ∗, AScenes::Actions)> bindFunction)

Public Attributes

- bool isHovered = false
- std::function< IScenes ∗(AScenes ∗, AScenes::Actions)> bind

6.9.1 Constructor & Destructor Documentation

6.9.1.1 BindComponent()

```
BindComponent::BindComponent (
            std::function< IScenes ∗(AScenes ∗, AScenes::Actions)> bindFunction )  [inline]
```

6.9.2 Member Data Documentation

6.9.2.1 bind

```
std::function<IScenes ∗(AScenes ∗, AScenes::Actions)> BindComponent::bind
```

6.9.2.2 isHovered

```
bool BindComponent::isHovered = false
```

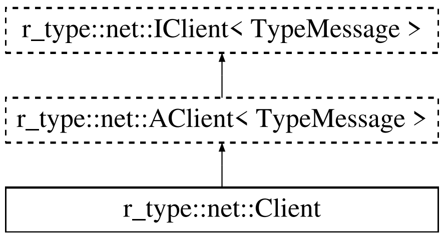
The documentation for this struct was generated from the following file:

- /home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/bind_component.hpp

6.10 r_type::net::Client Class Reference

```
#include <client.hpp>
```

Inheritance diagram for r_type::net::Client:


r_type::net::IClient< TypeMessage >
r_type::net::AClient< TypeMessage >
r_type::net::Client

Public Member Functions

- void PingServer ()

  Send a message to the server to get the ping.
- void MessageAll ()

  Send a message to the server to all other clients.
- void addEntity (EntityInformation entity, ComponentManager &componentManager, TextureManager &textureManager)
- void removeEntity (int entityId, ComponentManager &componentManager)
- void updateEntity (EntityInformation entity, ComponentManager &componentManager)

Additional Inherited Members

6.10.1 Member Function Documentation

6.10.1.1 addEntity()

```
void r_type::net::Client::addEntity (
            EntityInformation entity,
            ComponentManager & componentManager,
            TextureManager & textureManager )  [inline]
```

6.10.1.2 MessageAll()

```
void r_type::net::Client::MessageAll ( )  [inline]
```

Send a message to the server to all other clients.

6.10.1.3 PingServer()

```
void r_type::net::Client::PingServer ( )  [inline]
```

Send a message to the server to get the ping.

6.10.1.4 removeEntity()

```
void r_type::net::Client::removeEntity (
            int entityId,
            ComponentManager & componentManager )  [inline]
```

6.10.1.5 updateEntity()

```
void r_type::net::Client::updateEntity (
            EntityInformation entity,
            ComponentManager & componentManager )  [inline]
```

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

- /home/runner/work/R-Type/R-Type/Client/Interface/Include/Net/client.hpp

6.11 ComponentManager Class Reference

Manages the components of entities in an ECS system.

```
#include <component_manager.hpp>
```

Public Member Functions

- template<typename ComponentType , typename... Args>
  void addComponent (int entityId, Args &&...args)

  Adds a component to an entity.
- template<typename ComponentType >
  std::optional< ComponentType ∗ > getComponent (int entityId)

  Retrieves the component of the specified type associated with the given entity ID.
- template<typename ComponentType >
  std::optional< std::unordered_map< int, std::any > ∗ > getComponentMap ()

  Retrieves the component map for the specified component type.
- template<typename ComponentType >
  void removeEntityFromComponent (int entityId)

Private Attributes

- std::unordered_map< std::type_index, std::unordered_map< int, std::any > > components

  A component manager that stores components in an unordered map.

6.11.1 Detailed Description

Manages the components of entities in an ECS system.

The ComponentManager class provides functionality to add and retrieve components for entities in an ECS system. It uses an unordered map to store the components, where the key is the type of the component and the value is another unordered map that maps entity IDs to their corresponding component values.

6.11.2 Member Function Documentation

6.11.2.1 addComponent()

```
template<typename ComponentType , typename...  Args>
void ComponentManager::addComponent (
            int entityId,
            Args &&...  args )  [inline]
```

Adds a component to an entity.

Template Parameters

ComponentType	The type of the component to add.
Args	The types of the arguments to forward to the component's constructor.

Parameters

entity↩ Id	The ID of the entity to add the component to.
args	The arguments to forward to the component's constructor.

6.11.2.2 getComponent()

```
template<typename ComponentType >
std::optional<ComponentType *> ComponentManager::getComponent (
            int entityId )  [inline]
```

Retrieves the component of the specified type associated with the given entity ID.

Template Parameters

ComponentType	The type of the component to retrieve.

Parameters

entity↩ Id	The ID of the entity.

Returns

An optional pointer to the component if found, otherwise std::nullopt.

6.11.2.3 getComponentMap()

```
template<typename ComponentType >
std::optional<std::unordered_map<int, std::any> *> ComponentManager::getComponentMap ( )
[inline]
```

Retrieves the component map for the specified component type.

Template Parameters

ComponentType	The type of the component.

Returns

std::optional<std::unordered_map<int, std::any>*> The component map if found, otherwise std::nullopt.

6.11.2.4 removeEntityFromComponent()

```
template<typename ComponentType >
void ComponentManager::removeEntityFromComponent (
             int entityId )  [inline]
```

6.11.3 Member Data Documentation

6.11.3.1 components

```
std::unordered_map<std::type_index, std::unordered_map<int, std::any> > ComponentManager↩
::components [private]
```

A component manager that stores components in an unordered map.

This component manager uses an unordered map to store components. The keys of the outer map are of type std::type_index, which represents the type of the component. The values of the outer map are inner unordered maps, where the keys are of type int and represent the entity ID, and the values are of type std::any, which allows storing components of any type.

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

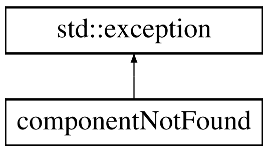
- /home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/component_manager.hpp

6.12 componentNotFound Class Reference

Exception class for when a component is not found.

```
#include <error_handling.hpp>
```

Inheritance diagram for componentNotFound:



Private Member Functions

- const char ∗ what () const noexcept override

6.12.1 Detailed Description

Exception class for when a component is not found.

This exception is thrown when a component is not found in the system. It inherits from std::exception and overrides the what() method to provide a custom error message.

6.12.2 Member Function Documentation

6.12.2.1 what()

```
const char* componentNotFound::what ( ) const  [inline], [override], [private], [noexcept]
```

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

- /home/runner/work/R-Type/R-Type/ECS/Interface/Include/error_handling.hpp

6.13 CreatableClientObject Class Reference

Enum class for the creatable client object.

```
#include <creatable_client_object.hpp>
```

6.13.1 Detailed Description

Enum class for the creatable client object.

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

- /home/runner/work/R-Type/R-Type/ECS/Interface/Include/creatable_client_object.hpp

6.14 EnemyComponent Struct Reference

```
#include <enemy_component.hpp>
```

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

- /home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/enemy_component.hpp

6.15 EnemyMissileComponent Struct Reference

```
#include <enemy_missile_component.hpp>
```

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

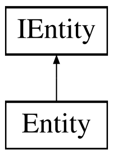
- /home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/enemy_missile_component.hpp

6.16 Entity Class Reference

Represents an entity in the ECS system.

```
#include <entity.hpp>
```

Inheritance diagram for Entity:



Public Member Functions

- Entity (int id)

    Constructs an Entity object with the given ID.
- int getId () const override

    Returns the ID of the entity.

Private Attributes

- int _id

6.16.1 Detailed Description

Represents an entity in the ECS system.

This class is a concrete implementation of the IEntity interface. It provides functionality to retrieve the ID of the entity.

6.16.2 Constructor & Destructor Documentation

6.16.2.1 Entity()

```
Entity::Entity (
              int id )  [inline], [explicit]
```

Constructs an Entity object with the given ID.

Parameters

id	The ID of the entity.

6.16.3 Member Function Documentation

6.16.3.1 getId()

```
int Entity::getId ( ) const  [inline], [override], [virtual]
```

Returns the ID of the entity.

Returns

The ID of the entity.

Implements IEntity.

6.16.4 Member Data Documentation

6.16.4.1 _id

```
int Entity::_id  [private]
```

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

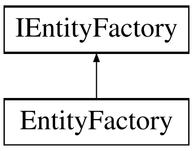
- /home/runner/work/R-Type/R-Type/ECS/Interface/Include/Entities/entity.hpp

6.17 EntityFactory Class Reference

A class responsible for creating different types of entities.

```
#include <entity_factory.hpp>
```

Inheritance diagram for EntityFactory:



Public Member Functions

- Entity createBackground (EntityManager &entityManager, ComponentManager &componentManager) override

  Creates a background entity.
- Entity createPlayer (EntityManager &entityManager, ComponentManager &componentManager, int nbrOf←Players) override

  Creates a player entity.
- Entity createAlly (EntityManager &entityManager, ComponentManager &componentManager) override

  Creates a player entity.
- Entity createBasicEnemy (EntityManager &entityManager, ComponentManager &componentManager) override

  Creates a basic enemy entity.
- Entity createBasicMonster (EntityManager &entityManager, ComponentManager &componentManager) override

  Creates a basic monster entity.
- Entity createPlayerMissile (EntityManager &entityManager, ComponentManager &componentManager, uint32_t entityId) override

  Creates a player missile entity.
- Entity createButton (EntityManager &entityManager, ComponentManager &componentManager, TextureManager &textureManager, std::string text, std::function< IScenes ∗(AScenes ∗)> ∗onClick, float x=0, float y=0)

  Creates a button entity.
- Entity createSmallButton (EntityManager &entityManager, ComponentManager &componentManager, TextureManager &textureManager, std::string text, std::function< IScenes ∗(AScenes ∗, AScenes::Actions)> ∗onClick, float x=0, float y=0)

  Creates a small button entity.
- Entity createAllyMissile (EntityManager &entityManager, ComponentManager &componentManager) override

  Creates an ally missile entity.
- Entity createEnemyMissile (EntityManager &entityManager, ComponentManager &componentManager, uint32_t entityId) override

  Creates an enemy missile entity.

6.17.1 Detailed Description

A class responsible for creating different types of entities.

6.17.2 Member Function Documentation

6.17.2.1 createAlly()

```
Entity EntityFactory::createAlly (
            EntityManager & entityManager,
            ComponentManager & componentManager )  [override], [virtual]
```

Creates a player entity.

This function creates a player entity using the provided entity manager and component manager.

Parameters

entityManager	The entity manager to use for creating the entity.
componentManager	The component manager to use for adding components to the entity.

Returns

The created player entity.

Implements IEntityFactory.

6.17.2.2 createAllyMissile()

```
Entity EntityFactory::createAllyMissile (
            EntityManager & entityManager,
            ComponentManager & componentManager )  [override], [virtual]
```

Creates an ally missile entity.

This function creates an ally missile entity using the provided entity manager and component manager.

Parameters

entityManager	The entity manager used to create the entity.
componentManager	The component manager used to manage the components of the entity.

Returns

The created ally missile entity.

Implements IEntityFactory.

6.17.2.3 createBackground()

```
Entity EntityFactory::createBackground (
            EntityManager & entityManager,
            ComponentManager & componentManager )  [override], [virtual]
```

Creates a background entity.

This function creates a background entity using the provided entity manager and component manager.

Parameters

entityManager	The entity manager to use for creating the entity.
componentManager	The component manager to use for adding components to the entity.

Returns

The created background entity.

Implements IEntityFactory.

6.17.2.4 createBasicEnemy()

```
Entity EntityFactory::createBasicEnemy (
            EntityManager & entityManager,
            ComponentManager & componentManager )  [override], [virtual]
```

Creates a basic enemy entity.

This function creates a basic enemy entity using the provided entity manager and component manager.

Parameters

entityManager	The entity manager used to create the entity.
componentManager	The component manager used to add components to the entity.

Returns

The created basic enemy entity.

Implements IEntityFactory.

6.17.2.5 createBasicMonster()

```
Entity EntityFactory::createBasicMonster (
            EntityManager & entityManager,
            ComponentManager & componentManager )  [override], [virtual]
```

Creates a basic monster entity.

This function creates a basic monster entity using the provided entity manager and component manager.

Parameters

entityManager	The entity manager used to create the entity.
componentManager	The component manager used to add components to the entity.

Returns

The created basic monster entity.

Implements IEntityFactory.

6.17.2.6 createButton()

```
Entity EntityFactory::createButton (
            EntityManager & entityManager,
            ComponentManager & componentManager,
            TextureManager & textureManager,
            std::string text,
            std::function< IScenes *(AScenes *)> * onClick,
            float x = 0,
            float y = 0 )  [virtual]
```

Creates a button entity.

This function creates a button entity with the specified parameters.

Parameters

entityManager	The entity manager to create the entity.
componentManager	The component manager to add components to the entity.
textureManager	The texture manager to load the button texture.
text	The text to display on the button.
onClick	The function to be called when the button is clicked.

Returns

The created button entity.

Implements IEntityFactory.

6.17.2.7 createEnemyMissile()

```
Entity EntityFactory::createEnemyMissile (
            EntityManager & entityManager,
            ComponentManager & componentManager,
            uint32_t entityId )  [override], [virtual]
```

Creates an enemy missile entity.

This function creates an enemy missile entity using the provided entity manager and component manager.

Parameters

entityManager	The entity manager used to create the entity.
componentManager	The component manager used to add components to the entity.
entityId	The id of the entity that shoot the missile

Returns

The created enemy missile entity.

Implements IEntityFactory.

6.17.2.8 createPlayer()

```
Entity EntityFactory::createPlayer (
            EntityManager & entityManager,
            ComponentManager & componentManager,
            int nbrOfPlayers ) [override], [virtual]
```

Creates a player entity.

This function creates a player entity using the provided entity manager and component manager.

Parameters

entityManager	The entity manager to use for creating the entity.
componentManager	The component manager to use for adding components to the entity.

Returns

The created player entity.

Implements IEntityFactory.

6.17.2.9 createPlayerMissile()

```
Entity EntityFactory::createPlayerMissile (
            EntityManager & entityManager,
            ComponentManager & componentManager,
            uint32_t entityId ) [override], [virtual]
```

Creates a player missile entity.

This function creates a player missile entity with the specified player ID and adds it to the entity manager. It also initializes the necessary components for the player missile entity using the component manager.

Parameters

entityManager	The entity manager to add the player missile entity to.
componentManager	The component manager to initialize the components for the player
entityId	The id of the entity that shoot the missile

Returns

The created player missile entity.

Implements IEntityFactory.

6.17.2.10 createSmallButton()

```
Entity EntityFactory::createSmallButton (
            EntityManager & entityManager,
            ComponentManager & componentManager,
            TextureManager & textureManager,
            std::string text,
            std::function< IScenes *(AScenes *, AScenes::Actions)> * onClick,
            float x = 0,
            float y = 0 )
```

Creates a small button entity.

This function creates a small button entity with the specified parameters.

Parameters

entityManager	The entity manager to create the entity.
componentManager	The component manager to add components to the entity.
textureManager	The texture manager to load the button texture.
text	The text to display on the button.
onClick	The function to be called when the button is clicked.

Returns

The created small button entity.

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

- /home/runner/work/R-Type/R-Type/ECS/Interface/Include/Entities/entity_factory.hpp
- /home/runner/work/R-Type/R-Type/ECS/Src/Entities/entity_factory.cpp

6.18 EntityInformation Struct Reference

Represents information about an entity.

`#include <entity_struct.hpp>`

Public Attributes

- uint32_t uniqueID = 0
- SpriteDataComponent spriteData
- vf2d vPos

6.18.1 Detailed Description

Represents information about an entity.

6.18.2 Member Data Documentation

6.18.2.1 spriteData

`SpriteDataComponent EntityInformation::spriteData`

6.18.2.2 uniqueID

`uint32_t EntityInformation::uniqueID = 0`

6.18.2.3 vPos

`vf2d EntityInformation::vPos`

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

- /home/runner/work/R-Type/R-Type/ECS/Interface/Include/entity_struct.hpp

6.19 EntityManager Class Reference

Class responsible for managing entities in the ECS system.

`#include <entity_manager.hpp>`

Public Member Functions

- Entity createEntity ()

    Create a Entity object.
- void removeEntity (int entityId)

    Remove an entity from the entity manager.
- Entity & getEntity (int entityId)

    Get an entity by its ID.
- const std::vector< Entity > & getAllEntities () const

    Get all entities in the entity manager.

Private Attributes

- int entityNb = 0

    The number of entities in the entity manager.
- std::vector< Entity > entities

6.19.1 Detailed Description

Class responsible for managing entities in the ECS system.

6.19.2 Member Function Documentation

6.19.2.1 createEntity()

```
Entity EntityManager::createEntity ( ) [inline]
```

Create a Entity object.

Returns

Entity

6.19.2.2 getAllEntities()

```
const std::vector<Entity>& EntityManager::getAllEntities ( ) const [inline]
```

Get all entities in the entity manager.

Returns

const std::vector<Entity>& A reference to the vector of entities.

This function returns a reference to the vector of entities in the entity manager.

6.19.2.3 getEntity()

```
Entity& EntityManager::getEntity (
             int entityId ) [inline]
```

Get an entity by its ID.

Parameters

entity↩ Id	The ID of the entity to retrieve.

Returns

Entity& A reference to the entity with the specified ID.

This function retrieves the entity with the specified ID from the entity manager. If the entity is not found, an entityNotFound exception is thrown.

6.19.2.4 removeEntity()

```
void EntityManager::removeEntity (
             int entityId )  [inline]
```

Remove an entity from the entity manager.

Parameters

entity↩ Id	The ID of the entity to remove.

This function removes the entity with the specified ID from the entity manager. If the entity is not found, an entityNotFound exception is thrown.

6.19.3 Member Data Documentation

6.19.3.1 entities

```
std::vector<Entity> EntityManager::entities  [private]
```

6.19.3.2 entityNb

```
int EntityManager::entityNb = 0  [private]
```

The number of entities in the entity manager.

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

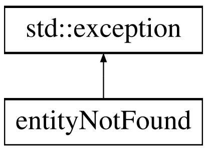
- /home/runner/work/R-Type/R-Type/ECS/Interface/Include/Entities/entity_manager.hpp

6.20 entityNotFound Class Reference

Exception class for entity not found error.

`#include <error_handling.hpp>`

Inheritance diagram for entityNotFound:



Private Member Functions

- const char ∗ what () const noexcept override

6.20.1 Detailed Description

Exception class for entity not found error.

This exception is thrown when an entity is not found. It is derived from the std::exception class. The `what()` function is overridden to provide a custom error message.

6.20.2 Member Function Documentation

6.20.2.1 what()

```
const char* entityNotFound::what ( ) const  [inline], [override], [private], [noexcept]
```

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

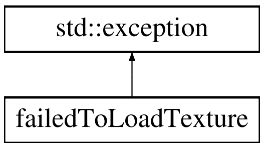
- /home/runner/work/R-Type/R-Type/ECS/Interface/Include/error_handling.hpp

6.21 failedToLoadTexture Class Reference

Exception class for failed texture loading.

`#include <error_handling.hpp>`

Inheritance diagram for failedToLoadTexture:



Private Member Functions

- const char ∗ what () const noexcept override

6.21.1 Detailed Description

Exception class for failed texture loading.

This exception is thrown when there is a failure to load a texture. It inherits from the std::exception class and overrides the what() method to provide a custom error message.

6.21.2 Member Function Documentation

6.21.2.1 what()

```
const char* failedToLoadTexture::what ( ) const  [inline], [override], [private], [noexcept]
```

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

- /home/runner/work/R-Type/R-Type/ECS/Interface/Include/error_handling.hpp

6.22 HealthComponent Struct Reference

```
#include <health_component.hpp>
```

Public Attributes

- int max_health
- int health

6.22.1 Member Data Documentation

6.22.1.1 health

```
int HealthComponent::health
```

6.22.1.2 max_health

```
int HealthComponent::max_health
```

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

- /home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/health_component.hpp

6.23 HitboxComponent Struct Reference

```
#include <hitbox_component.hpp>
```

Public Attributes

- int w
- int h

6.23.1 Member Data Documentation

6.23.1.1 h

```
int HitboxComponent::h
```

6.23.1.2 w

```
int HitboxComponent::w
```

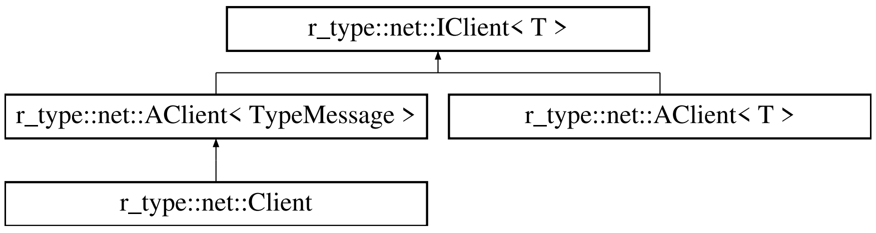
The documentation for this struct was generated from the following file:

- /home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/hitbox_component.hpp

6.24 r_type::net::IClient< T > Class Template Reference

```
#include <i_client.hpp>
```

Inheritance diagram for r_type::net::IClient< T >:


r_type::net::IClient< T >
r_type::net::AClient< TypeMessage >
r_type::net::AClient< T >
r_type::net::Client

Public Member Functions

- IClient ()
- virtual ∼IClient ()
- virtual bool Connect (const std::string &host, const uint16_t port)=0

  Connects to a remote host using UDP protocol.
- virtual void Disconnect ()=0

  Disconnects the client from the server.
- virtual bool IsConnected ()=0

  Checks if the client is connected to the server.
- virtual void Send (const Message< T > &msg)=0

  Send message to server.
- virtual ThreadSafeQueue< OwnedMessage< T > > & Incoming ()=0

  get incoming messages

6.24.1 Constructor & Destructor Documentation

6.24.1.1 IClient()

```
template<typename T >
r_type::net::IClient< T >::IClient ( )  [inline]
```

6.24.1.2 ∼IClient()

```
template<typename T >
virtual r_type::net::IClient< T >::∼IClient ( )  [inline], [virtual]
```

6.24.2 Member Function Documentation

6.24.2.1 Connect()

```
template<typename T >
virtual bool r_type::net::IClient< T >::Connect (
             const std::string & host,
             const uint16_t port )  [pure virtual]
```

Connects to a remote host using UDP protocol.

Parameters

host	The IP address or hostname of the remote host.
port	The port number of the remote host.

Returns

> true if the connection is successful
>
> false otherwise.

Implemented in r_type::net::AClient< T >, and r_type::net::AClient< TypeMessage >.

6.24.2.2 Disconnect()

```
template<typename T >
virtual void r_type::net::IClient< T >::Disconnect ( )  [pure virtual]
```

Disconnects the client from the server.

This function disconnects the client from the server if it is currently connected. It stops the context and joins the context thread. It also releases the connection resource.

Implemented in r_type::net::AClient< T >, and r_type::net::AClient< TypeMessage >.

6.24.2.3 Incoming()

```
template<typename T >
virtual ThreadSafeQueue<OwnedMessage<T> >& r_type::net::IClient< T >::Incoming ( )  [pure
virtual]
```

get incoming messages

Returns

> ThreadSafeQueue<OwnedMessage<T>>&

Implemented in r_type::net::AClient< T >, and r_type::net::AClient< TypeMessage >.

6.24.2.4 IsConnected()

```
template<typename T >
virtual bool r_type::net::IClient< T >::IsConnected ( )  [pure virtual]
```

Checks if the client is connected to the server.

Returns

> true
>
> false

Implemented in r_type::net::AClient< T >, and r_type::net::AClient< TypeMessage >.

6.24.2.5 Send()

```
template<typename T >
virtual void r_type::net::IClient< T >::Send (
            const Message< T > & msg )  [pure virtual]
```

Send message to server.

Parameters

msg	

Implemented in r_type::net::AClient< T >.

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

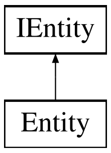
- /home/runner/work/R-Type/R-Type/Client/Interface/Include/Net/i_client.hpp

6.25 IEntity Class Reference

The IEntity class represents an entity in the system.

```
#include <i_entity.hpp>
```

Inheritance diagram for IEntity:



Public Member Functions

- virtual ∼IEntity ()=default

  Destructor for the IEntity class.
- virtual int getId () const =0

  Gets the ID of the entity.

6.25.1 Detailed Description

The IEntity class represents an entity in the system.

This class provides an interface for entities in the system. It defines a pure virtual function getId() which returns the ID of the entity.

Note

This class is meant to be inherited from and should not be instantiated directly.

6.25.2 Constructor & Destructor Documentation

6.25.2.1 ∼IEntity()

```
virtual IEntity::~IEntity ( ) [virtual], [default]
```

Destructor for the IEntity class.

6.25.3 Member Function Documentation

6.25.3.1 getId()

```
virtual int IEntity::getId ( ) const [pure virtual]
```

Gets the ID of the entity.

Returns

The ID of the entity.

Implemented in Entity.

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

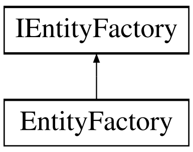
- /home/runner/work/R-Type/R-Type/ECS/Interface/Include/Entities/i_entity.hpp

6.26 IEntityFactory Class Reference

The interface for an entity factory.

```
#include <i_entity_factory.hpp>
```

Inheritance diagram for IEntityFactory:



Public Member Functions

- virtual ∼IEntityFactory ()=default

  Destroy the IEntityFactory object.
- virtual Entity createBackground (EntityManager &entityManager, ComponentManager &component↩ Manager)=0

  Creates a background entity.
- virtual Entity createPlayer (EntityManager &entityManager, ComponentManager &componentManager, int nbrOfPlayers)=0

  Creates a player entity.
- virtual Entity createAlly (EntityManager &entityManager, ComponentManager &componentManager)=0

  Creates an ally entity.
- virtual Entity createBasicEnemy (EntityManager &entityManager, ComponentManager &component↩ Manager)=0

  Creates a basic enemy entity.
- virtual Entity createBasicMonster (EntityManager &entityManager, ComponentManager &component↩ Manager)=0

  Creates a basic monster entity.
- virtual Entity createPlayerMissile (EntityManager &entityManager, ComponentManager &component↩ Manager, uint32_t entityId)=0

  Creates a player missile entity.
- virtual Entity createAllyMissile (EntityManager &entityManager, ComponentManager &component↩ Manager)=0

  Creates an ally missile entity.
- virtual Entity createEnemyMissile (EntityManager &entityManager, ComponentManager &component↩ Manager, uint32_t entityId)=0

  Creates an enemy missile entity.
- virtual Entity createButton (EntityManager &entityManager, ComponentManager &componentManager, TextureManager &textureManager, std::string text, std::function< IScenes ∗(AScenes ∗)> ∗onClick, float x, float y)=0

  Creates a button entity.

6.26.1 Detailed Description

The interface for an entity factory.

This interface defines the methods for creating different types of entities in the game. Each method takes references to the entity manager, component manager, and other necessary parameters, and returns an entity object.

Note

This is an abstract base class and cannot be instantiated directly.

6.26.2 Constructor & Destructor Documentation

6.26.2.1 ∼IEntityFactory()

```
virtual IEntityFactory::∼IEntityFactory ( )  [virtual], [default]
```

Destroy the IEntityFactory object.

6.26.3 Member Function Documentation

6.26.3.1 createAlly()

```
virtual Entity IEntityFactory::createAlly (
            EntityManager & entityManager,
            ComponentManager & componentManager )  [pure virtual]
```

Creates an ally entity.

This function creates an ally entity using the provided entity manager and component manager.

Parameters

entityManager	The entity manager used to create the entity.
componentManager	The component manager used to manage the components of the entity.

Returns

The created ally entity.

Implemented in EntityFactory.

6.26.3.2 createAllyMissile()

```
virtual Entity IEntityFactory::createAllyMissile (
            EntityManager & entityManager,
            ComponentManager & componentManager )  [pure virtual]
```

Creates an ally missile entity.

This function creates an ally missile entity using the provided entity manager and component manager.

Parameters

entityManager	The entity manager used to create the entity.
componentManager	The component manager used to manage the components of the entity.

Returns

The created ally missile entity.

Implemented in EntityFactory.

6.26.3.3 createBackground()

```
virtual Entity IEntityFactory::createBackground (
            EntityManager & entityManager,
            ComponentManager & componentManager )  [pure virtual]
```

Creates a background entity.

This function creates a background entity using the provided entity manager and component manager.

Parameters

entityManager	The entity manager to use for creating the entity.
componentManager	The component manager to use for adding components to the entity.

Returns

The created background entity.

Implemented in EntityFactory.

6.26.3.4 createBasicEnemy()

```
virtual Entity IEntityFactory::createBasicEnemy (
            EntityManager & entityManager,
            ComponentManager & componentManager )  [pure virtual]
```

Creates a basic enemy entity.

This function creates a basic enemy entity using the provided entity manager and component manager.

Parameters

entityManager	The entity manager used to create the entity.
componentManager	The component manager used to add components to the entity.

Returns

The created basic enemy entity.

Implemented in EntityFactory.

6.26.3.5 createBasicMonster()

```
virtual Entity IEntityFactory::createBasicMonster (
            EntityManager & entityManager,
            ComponentManager & componentManager )  [pure virtual]
```

Creates a basic monster entity.

This function creates a basic monster entity using the provided entity manager and component manager.

Parameters

entityManager	The entity manager used to create the entity.
componentManager	The component manager used to add components to the entity.

Returns

The created basic monster entity.

Implemented in EntityFactory.

6.26.3.6 createButton()

```
virtual Entity IEntityFactory::createButton (
            EntityManager & entityManager,
            ComponentManager & componentManager,
            TextureManager & textureManager,
            std::string text,
            std::function< IScenes *(AScenes *)> * onClick,
            float x,
            float y )  [pure virtual]
```

Creates a button entity.

This function creates a button entity using the provided entity manager, component manager, texture manager, text, and onClick function. The button entity represents a clickable button in the game.

Parameters

entityManager	The entity manager used to create the button entity.
componentManager	The component manager used to manage the components of the button entity.
textureManager	The texture manager used to load the textures for the button entity.
text	The text displayed on the button.
onClick	The function to be called when the button is clicked.

Returns

The created button entity.

Implemented in EntityFactory.

6.26.3.7 createEnemyMissile()

```
virtual Entity IEntityFactory::createEnemyMissile (
            EntityManager & entityManager,
            ComponentManager & componentManager,
            uint32_t entityId ) [pure virtual]
```

Creates an enemy missile entity.

This function creates an enemy missile entity using the provided entity manager and component manager.

Parameters

entityManager	The entity manager used to create the entity.
componentManager	The component manager used to add components to the entity.

Returns

The created enemy missile entity.

Implemented in EntityFactory.

6.26.3.8 createPlayer()

```
virtual Entity IEntityFactory::createPlayer (
            EntityManager & entityManager,
            ComponentManager & componentManager,
            int nbrOfPlayers ) [pure virtual]
```

Creates a player entity.

This function creates a player entity using the provided entity manager and component manager.

Parameters

entityManager	The entity manager used to create the entity.
componentManager	The component manager used to add components to the entity.

Returns

The created player entity.

Implemented in EntityFactory.

6.26.3.9 createPlayerMissile()

```
virtual Entity IEntityFactory::createPlayerMissile (
            EntityManager & entityManager,
            ComponentManager & componentManager,
            uint32_t entityId )  [pure virtual]
```

Creates a player missile entity.

This function creates a player missile entity with the specified player ID and adds it to the entity manager. It also initializes the necessary components for the player missile entity using the component manager.

Parameters

entityId	The ID of the entity that shoot the missile.
entityManager	The entity manager to add the player missile entity to.
componentManager	The component manager to initialize the components for the player missile entity.

Returns

The created player missile entity.

Implemented in EntityFactory.

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

- /home/runner/work/R-Type/R-Type/ECS/Interface/Include/Entities/i_entity_factory.hpp

6.27 InputComponent Struct Reference

```
#include <input_component.hpp>
```

Public Attributes

- InputType input

6.27.1 Member Data Documentation

6.27.1.1 input

```
InputType InputComponent::input
```

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

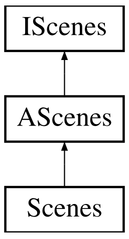
- /home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/input_component.hpp

6.28 IScenes Class Reference

Interface for managing different scenes in a game.

```
#include <i_scenes.hpp>
```

Inheritance diagram for IScenes:



Public Member Functions

- virtual ∼IScenes ()=default
- virtual void mainMenu ()=0

  Displays the main menu and creates necessary entities.
- virtual void gameLoop ()=0

  Displays the main game loop and creates necessary entities.
- virtual void settingsMenu ()=0

  Displays the settings menu and creates necessary entities.
- virtual void inGameMenu ()=0

  Displays the in-game menu and creates necessary entities.
- virtual void render ()=0

  Displays the current scene and manages its components.
- virtual bool shouldQuit ()=0

  Checks if the game should quit.
- virtual sf::RenderWindow ∗ getRenderWindow ()=0

  Gets the render window.

6.28.1 Detailed Description

Interface for managing different scenes in a game.

This interface declares the methods for displaying and managing various scenes in a game, such as the main menu, game loop, settings menu, and in-game menu.

6.28.2 Constructor & Destructor Documentation

6.28.2.1 ∼IScenes()

```
virtual IScenes::∼IScenes ( )  [virtual], [default]
```

6.28.3 Member Function Documentation

6.28.3.1 gameLoop()

```
virtual void IScenes::gameLoop ( )  [pure virtual]
```

Displays the main game loop and creates necessary entities.

Implemented in Scenes.

6.28.3.2 getRenderWindow()

```
virtual sf::RenderWindow* IScenes::getRenderWindow ( )  [pure virtual]
```

Gets the render window.

Returns

Pointer to the sf::RenderWindow.

Implemented in Scenes.

6.28.3.3 inGameMenu()

```
virtual void IScenes::inGameMenu ( )  [pure virtual]
```

Displays the in-game menu and creates necessary entities.

Implemented in Scenes.

6.28.3.4 mainMenu()

```
virtual void IScenes::mainMenu ( )  [pure virtual]
```

Displays the main menu and creates necessary entities.

Implemented in Scenes.

6.28.3.5 render()

```
virtual void IScenes::render ( ) [pure virtual]
```

Displays the current scene and manages its components.

Implemented in Scenes.

6.28.3.6 settingsMenu()

```
virtual void IScenes::settingsMenu ( ) [pure virtual]
```

Displays the settings menu and creates necessary entities.

Implemented in Scenes.

6.28.3.7 shouldQuit()

```
virtual bool IScenes::shouldQuit ( ) [pure virtual]
```

Checks if the game should quit.

Returns

True if the game should quit, false otherwise.

Implemented in Scenes.

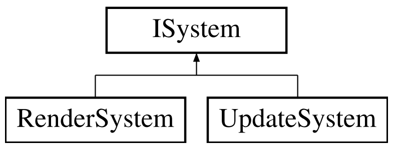
The documentation for this class was generated from the following file:

- /home/runner/work/R-Type/R-Type/ECS/Interface/Include/i_scenes.hpp

6.29 ISystem Class Reference

```
#include <i_system.hpp>
```

Inheritance diagram for ISystem:



Public Member Functions

- ISystem ()=default
- virtual ~ISystem ()=default
- virtual void update (float deltaTime)=0

6.29.1 Constructor & Destructor Documentation

6.29.1.1 ISystem()

```
ISystem::ISystem ( )  [default]
```

6.29.1.2 ~ISystem()

```
virtual ISystem::~ISystem ( )  [virtual], [default]
```

6.29.2 Member Function Documentation

6.29.2.1 update()

```
virtual void ISystem::update (
            float deltaTime )  [pure virtual]
```

Implemented in UpdateSystem, and RenderSystem.

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

- /home/runner/work/R-Type/R-Type/ECS/Interface/Include/Systems/i_system.hpp

6.30 labelComponent Struct Reference

```
#include <label_component.hpp>
```

Public Attributes

- std::string name
- int x
- int y

6.30.1 Member Data Documentation

6.30.1.1 name

`std::string labelComponent::name`

6.30.1.2 x

`int labelComponent::x`

6.30.1.3 y

`int labelComponent::y`

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

- /home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/label_component.hpp

6.31 OffsetComponent Struct Reference

`#include <offset_component.hpp>`

Public Attributes

- float offset

6.31.1 Member Data Documentation

6.31.1.1 offset

`float OffsetComponent::offset`

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

- /home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/offset_component.hpp

6.32 OnClickComponent Struct Reference

```
#include <on_click_component.hpp>
```

Public Member Functions

- OnClickComponent (std::function< IScenes ∗(AScenes ∗)> &onClickfunction)

Public Attributes

- bool isClicked = false
- std::function< IScenes ∗(AScenes ∗)> & onClick

6.32.1 Constructor & Destructor Documentation

6.32.1.1 OnClickComponent()

```
OnClickComponent::OnClickComponent (
            std::function< IScenes ∗(AScenes ∗)> & onClickfunction )  [inline]
```

6.32.2 Member Data Documentation

6.32.2.1 isClicked

```
bool OnClickComponent::isClicked = false
```

6.32.2.2 onClick

```
std::function<IScenes ∗(AScenes ∗)>& OnClickComponent::onClick
```

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

- /home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/on_click_component.hpp

6.33 PlayerComponent Struct Reference

```
#include <player_component.hpp>
```

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

- /home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/player_component.hpp

6.34 PlayerMissileComponent Struct Reference

```
#include <player_missile_component.hpp>
```

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

- /home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/player_missile_component.hpp

6.35 PositionComponent Struct Reference

```
#include <position_component.hpp>
```

Public Member Functions

- PositionComponent (float x, float y)

Public Attributes

- float x
- float y

6.35.1 Constructor & Destructor Documentation

6.35.1.1 PositionComponent()

```
PositionComponent::PositionComponent (
                float x,
                float y )  [inline]
```

6.35.2 Member Data Documentation

6.35.2.1 x

```
float PositionComponent::x
```

6.35.2.2 y

```
float PositionComponent::y
```

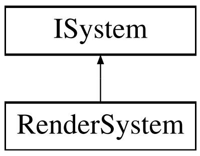
The documentation for this struct was generated from the following file:

- /home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/position_component.hpp

6.36 RenderSystem Class Reference

```
#include <render_system.hpp>
```

Inheritance diagram for RenderSystem:



Public Member Functions

- RenderSystem (sf::RenderWindow &window, ComponentManager &componentManager)
- void update (float deltaTime) override
- void render (ComponentManager &componentManager)

Private Attributes

- sf::RenderWindow & _window
- ComponentManager & _componentManager
- sf::Font _font

6.36.1 Constructor & Destructor Documentation

6.36.1.1 RenderSystem()

```
RenderSystem::RenderSystem (
            sf::RenderWindow & window,
            ComponentManager & componentManager )  [inline]
```

6.36.2 Member Function Documentation

6.36.2.1 render()

```
void RenderSystem::render (
            ComponentManager & componentManager )
```

6.36.2.2 update()

```
void RenderSystem::update (
            float deltaTime )  [inline], [override], [virtual]
```

Implements ISystem.

6.36.3 Member Data Documentation

6.36.3.1 _componentManager

```
ComponentManager& RenderSystem::_componentManager  [private]
```

6.36.3.2 _font

```
sf::Font RenderSystem::_font  [private]
```

6.36.3.3 _window

```
sf::RenderWindow& RenderSystem::_window  [private]
```

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

- /home/runner/work/R-Type/R-Type/ECS/Interface/Include/Systems/render_system.hpp
- /home/runner/work/R-Type/R-Type/ECS/Src/Systems/render_system.cpp

6.37 Rtype Class Reference

The Rtype class handles the main game logic, including initialization, main menu, game loop, event handling, server message processing, game updating, and rendering.

`#include <r_type_client.hpp>`

Public Member Functions

- Rtype ()

  Construct a new Rtype object This will init the player.
- void run ()

  If _mainMenu variable is true, call mainMenu.
- void mainMenu ()

  Open window.
- void gameLoop ()

  Open window.
- void handleEvents ()

  This is where I will handle the events for the window & player (key input, etc.).
- void processServerMessages ()

  This is where I will process the info from the server.
- void updateGame ()

  This is where I will update the time, position of sprites, etc.
- void renderGame ()

  This is where I will render the game.

Private Attributes

- Scenes ∗ _scenes

  Set the Game Mode object.
- sf::RenderWindow _window

  The main window for rendering graphics.

6.37.1 Detailed Description

The Rtype class handles the main game logic, including initialization, main menu, game loop, event handling, server message processing, game updating, and rendering.

The Rtype class is responsible for initializing the player, managing the main menu, running the game loop, handling events, processing server messages, updating the game state, and rendering the game.

6.37.2 Usage

To use the Rtype class, create an instance of it and call the run() method to start the game.

```
Rtype game;
game.run();
```

6.37.3 Methods

- Rtype(): Constructs a new Rtype object and initializes the player.
- void run(): Starts the main loop of the game, switching between the main menu and the game loop based on the _mainMenu variable.
- void mainMenu(): Displays the main menu with options to start the game, view help, toggle daltonic mode, and select speed. Handles user input for these options.
- void gameLoop(): Runs the main game loop, calling handleEvents, updateGame, processCommands, and render functions.
- void handleEvents(): Handles window and player events, such as key input, and sends updated player information to the server.
- void processServerMessages(): Processes messages received from the server.
- void updateGame(): Updates the game state, including time, position of sprites, and other game elements.
- void renderGame(): Renders the game, including clearing the window, drawing the background, rendering game elements, and displaying the window.

6.37.4 Members

- Scenes $*$_scenes: Pointer to the scenes object.
- sf::RenderWindow _window: The window object used for rendering the game.

6.37.5 Constructor & Destructor Documentation

6.37.5.1 Rtype()

```
Rtype::Rtype ( )
```

Construct a new Rtype object This will init the player.

Construct a new Rtype:: Rtype object.

Default easy mode and normal daltonism mode. Ex: renderSystem.addEntity(player), inputSystem.add↩Entity(player), collisionSystem.addEntity(player), etc.

6.37.6 Member Function Documentation

6.37.6.1 gameLoop()

```
void Rtype::gameLoop ( )
```

Open window.

This is where I will call the handleEvents, updateGame, processCommands, and render functions.

6.37.6.2 handleEvents()

```
void Rtype::handleEvents ( )
```

This is where I will handle the events for the window & player (key input, etc.).

When key is pressed, move player, and send new player info to server.

6.37.6.3 mainMenu()

```
void Rtype::mainMenu ( )
```

Open window.

(handleEvents). Display the main menu with start, help, daltonic mode, and speed selection buttons. On start, set _mainMenu to false, close window, and return. When active, daltonic_mode will be set to true, and draw a color filter over the screen until deactivated. Can set keybindings as well, either default or customized

6.37.6.4 processServerMessages()

```
void Rtype::processServerMessages ( )
```

This is where I will process the info from the server.

6.37.6.5 renderGame()

```
void Rtype::renderGame ( )
```

This is where I will render the game.

Ex: window.clear(), window.draw(background), renderSystem.render(window), window.display, etc.

6.37.6.6 run()

```
void Rtype::run ( )
```

If _mainMenu variable is true, call mainMenu.

While _mainMenu is false, call gameLoop.

6.37.6.7 updateGame()

```
void Rtype::updateGame ( )
```

This is where I will update the time, position of sprites, etc.

Ex: inputSystem.update(deltaTime.asSeconds()), renderSystem.update(deltaTime.asSeconds()), etc.

6.37.7 Member Data Documentation

6.37.7.1 _scenes

```
Scenes* Rtype::_scenes  [private]
```

Set the Game Mode object.

Parameters

mode	Pointer to the Scenes object.

This member variable holds a pointer to an instance of the Scenes class, which is used to manage and control different scenes within the client.

6.37.7.2 _window

```
sf::RenderWindow Rtype::_window  [private]
```

The main window for rendering graphics.

This member represents the window where all the graphical content of the application will be displayed. It is an instance of the sf::RenderWindow class from the SFML library.

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

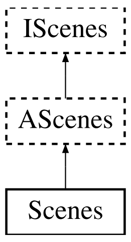
- /home/runner/work/R-Type/R-Type/Client/Interface/Include/r_type_client.hpp
- /home/runner/work/R-Type/R-Type/Client/Src/r-type_client.cpp

6.38 Scenes Class Reference

Represents a class that manages different scenes in a game.

```
#include <scenes.hpp>
```

Inheritance diagram for Scenes:



Public Member Functions

- Scenes (sf::RenderWindow ∗window)

  Construct a new Scenes object.
- ∼Scenes ()=default

  Destroy the Scenes object.
- void mainMenu ()

  displays the main menu, creates all the necessary entities
- void gameLoop ()

  displays the main game loop, creates all the necessary entities
- void settingsMenu ()

  displays the settings menu, creates all the necessary entities
- void inGameMenu ()

  displays the in game menu, creates all the necessary entities
- void render ()

  display what must be displayed (main menu, game loop, settings menu, in game menu), creates all the components needed and manages them
- bool shouldQuit ()

  check if game should stop running
- sf::RenderWindow ∗ getRenderWindow ()

  Get the RenderWindow object.

Additional Inherited Members

6.38.1 Detailed Description

Represents a class that manages different scenes in a game.

The Scenes class provides functionality to display and manage various scenes in a game, such as the main menu, game loop, settings menu, and in-game menu. It also allows setting the game mode and daltonism mode.

6.38.2 Constructor & Destructor Documentation

6.38.2.1 Scenes()

```
Scenes::Scenes (
            sf::RenderWindow ∗ window )
```

Construct a new Scenes object.

Parameters

window	

6.38.2.2 ∼Scenes()

```
Scenes::∼Scenes ( )  [default]
```

Destroy the Scenes object.

6.38.3 Member Function Documentation

6.38.3.1 gameLoop()

```
void Scenes::gameLoop ( )  [virtual]
```

displays the main game loop, creates all the necessary entities

This function handles the main game loop for the Scenes class.

It contains the logic for connecting to a server, updating entities, handling user input, and rendering the game.

The game loop performs the following steps:

1. Connects to a server using the r_type::net::Client class.
2. Initializes the ComponentManager, TextureManager, and EntityManager.
3. Creates a background entity and sets its sprite component.
4. Defines lambda functions for updating player position and firing missiles.
5. Enters the main loop, which continues until the window is closed.
6. Within the loop, it checks for user input events and handles them accordingly.
7. If the server is connected, it processes incoming messages and updates entities accordingly.
8. It then updates the entities using the UpdateSystem and renders them using the RenderSystem.

Note

This code assumes the presence of the r_type::net::Client, ComponentManager, TextureManager, EntityManager, UpdateSystem, and RenderSystem classes.

See also

r_type::net::Client
ComponentManager
TextureManager
EntityManager
UpdateSystem
RenderSystem

Implements IScenes.

6.38.3.2 getRenderWindow()

```
sf::RenderWindow* Scenes::getRenderWindow ( )  [inline], [virtual]
```

Get the RenderWindow object.

Returns

sf::RenderWindow∗

Implements IScenes.

6.38.3.3 inGameMenu()

```
void Scenes::inGameMenu ( )  [virtual]
```

displays the in game menu, creates all the necessary entities

Displays the in-game menu.

Implements IScenes.

6.38.3.4 mainMenu()

```
void Scenes::mainMenu ( )  [virtual]
```

displays the main menu, creates all the necessary entities

Displays the main menu scene.

This function creates the main menu scene, including the background, buttons, and event handling. The main menu scene allows the user to navigate to different scenes by clicking on the buttons. The buttons include "Play", "↩ Settings", and "Quit". The function continuously updates and renders the scene until the user closes the window or navigates to a different scene.

Returns

void

Implements IScenes.

6.38.3.5 render()

```
void Scenes::render ( )  [virtual]
```

display what must be displayed (main menu, game loop, settings menu, in game menu), creates all the components needed and manages them

Renders the current scene based on the value of currentScene.

The render function uses a switch statement to determine which scene to render. It calls the corresponding member function based on the value of currentScene.

Note

The currentScene variable must be set before calling this function.

Implements IScenes.

6.38.3.6 settingsMenu()

```
void Scenes::settingsMenu ( )  [virtual]
```

displays the settings menu, creates all the necessary entities

Displays the settings menu.

This function is responsible for displaying the settings menu in the game. It does not return any value.

Implements IScenes.

6.38.3.7 shouldQuit()

```
bool Scenes::shouldQuit ( )  [inline], [virtual]
```

check if game should stop running

Returns

true

false

Implements IScenes.

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

- /home/runner/work/R-Type/R-Type/Client/Interface/Include/scenes.hpp
- /home/runner/work/R-Type/R-Type/Client/Src/scenes.cpp

6.39 ScoreComponent Struct Reference

```
#include <score_component.hpp>
```

Public Attributes

- int score

6.39.1 Member Data Documentation

6.39.1.1 score

```
int ScoreComponent::score
```

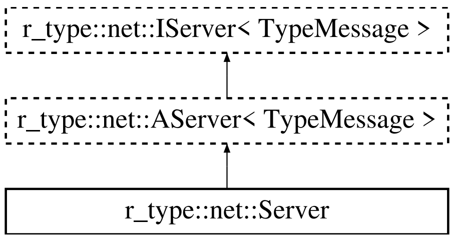
The documentation for this struct was generated from the following file:

- /home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/score_component.hpp

6.40 r_type::net::Server Class Reference

```
#include <server.hpp>
```

Inheritance diagram for r_type::net::Server:


r_type::net::IServer< TypeMessage >
r_type::net::AServer< TypeMessage >
r_type::net::Server

Public Member Functions

- Server (uint16_t nPort)
- ∼Server ()

Protected Member Functions

- bool OnClientConnect (std::shared_ptr< r_type::net::Connection< TypeMessage >> client)

  Called when a client is validated.
- void OnClientDisconnect (std::shared_ptr< r_type::net::Connection< TypeMessage >> client, r_type::net↩ ::Message< TypeMessage > &msg)

  Called when a client appears to have disconnected.
- void OnMessage (std::shared_ptr< r_type::net::Connection< TypeMessage >> client, r_type::net::↩ Message< TypeMessage > &msg)

  Called when a message is received from a client.
- void InitListEntities (std::shared_ptr< r_type::net::Connection< TypeMessage >> client, u_int32_t entityID)

  Sends a list of existing entities to a newly connected client for initialization.

Additional Inherited Members

6.40.1 Constructor & Destructor Documentation

6.40.1.1 Server()

```
r_type::net::Server::Server (
            uint16_t nPort )  [inline]
```

6.40.1.2 ∼Server()

```
r_type::net::Server::∼Server ( )  [inline]
```

6.40.2 Member Function Documentation

6.40.2.1 InitListEntities()

```
void r_type::net::Server::InitListEntities (
            std::shared_ptr< r_type::net::Connection< TypeMessage >> client,
            u_int32_t entityID )  [protected], [virtual]
```

Sends a list of existing entities to a newly connected client for initialization.

The function iterates through all existing entities and sends their information to the newly connected client, excluding specific entities such as the client itself.

Parameters

client	The connection to the client.
entityID	The ID of the entity to exclude (usually the client's own entity).

Implements r_type::net::AServer< TypeMessage >.

6.40.2.2 OnClientConnect()

```
bool r_type::net::Server::OnClientConnect (
             std::shared_ptr< r_type::net::Connection< TypeMessage >> client )  [protected]
```

Called when a client is validated.

Parameters

client	

Returns

true

false

6.40.2.3 OnClientDisconnect()

```
void r_type::net::Server::OnClientDisconnect (
             std::shared_ptr< r_type::net::Connection< TypeMessage >> client,
             r_type::net::Message< TypeMessage > & msg )  [protected]
```

Called when a client appears to have disconnected.

Parameters

client	

6.40.2.4 OnMessage()

```
void r_type::net::Server::OnMessage (
             std::shared_ptr< r_type::net::Connection< TypeMessage >> client,
             r_type::net::Message< TypeMessage > & msg )  [protected]
```

Called when a message is received from a client.

Parameters

client	
msg	

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

- /home/runner/work/R-Type/R-Type/Server/Interface/Include/Net/server.hpp
- /home/runner/work/R-Type/R-Type/Server/Src/server.cpp

6.41 SpriteComponent Struct Reference

```
#include <sprite_component.hpp>
```

Public Member Functions

- SpriteComponent (sf::Texture &texture, const float posX, float posY, const sf::Vector2f &scale)

Public Attributes

- sf::Sprite sprite

6.41.1 Constructor & Destructor Documentation

6.41.1.1 SpriteComponent()

```
SpriteComponent::SpriteComponent (
             sf::Texture & texture,
             const float posX,
             float posY,
             const sf::Vector2f & scale )  [inline]
```

6.41.2 Member Data Documentation

6.41.2.1 sprite

```
sf::Sprite SpriteComponent::sprite
```

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

- /home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/sprite_component.hpp

6.42 SpriteDataComponent Struct Reference

```
#include <sprite_data_component.hpp>
```

Public Attributes

- SpritePath spritePath
- Vector< uint32_t > offSet
- Vector< uint32_t > dimension
- Vector< float > scale

6.42.1 Member Data Documentation

6.42.1.1 dimension

```
Vector<uint32_t> SpriteDataComponent::dimension
```

6.42.1.2 offSet

```
Vector<uint32_t> SpriteDataComponent::offSet
```

6.42.1.3 scale

```
Vector<float> SpriteDataComponent::scale
```

6.42.1.4 spritePath

```
SpritePath SpriteDataComponent::spritePath
```

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

- /home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/sprite_data_component.hpp

6.43 SystemManager Class Reference

```
#include <system_manager.hpp>
```

Public Member Functions

- void addSystem (std::shared_ptr< ISystem > system)
- void updateSystems (float deltaTime)

Private Attributes

- std::vector< std::shared_ptr< ISystem > > systems

6.43.1 Member Function Documentation

6.43.1.1 addSystem()

```
void SystemManager::addSystem (
            std::shared_ptr< ISystem > system )  [inline]
```

6.43.1.2 updateSystems()

```
void SystemManager::updateSystems (
            float deltaTime )  [inline]
```

6.43.2 Member Data Documentation

6.43.2.1 systems

```
std::vector<std::shared_ptr<ISystem> > SystemManager::systems  [private]
```

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

- /home/runner/work/R-Type/R-Type/ECS/Interface/Include/Systems/system_manager.hpp

6.44 TextComponent Struct Reference

```
#include <text_component.hpp>
```

Public Member Functions

- TextComponent (std::string text)

Public Attributes

- std::string _text

6.44.1 Constructor & Destructor Documentation

6.44.1.1 TextComponent()

```
TextComponent::TextComponent (
             std::string text )  [inline]
```

6.44.2 Member Data Documentation

6.44.2.1 _text

```
std::string TextComponent::_text
```

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

- /home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/text_component.hpp

6.45 TextureManager Class Reference

```
#include <texture_manager.hpp>
```

Public Member Functions

- sf::Texture & getTexture (const std::string &filePath)

    Retrieves a texture from the texture manager.

Private Attributes

- std::unordered_map< std::string, sf::Texture > textures

    A container for storing textures with string keys.

6.45.1 Member Function Documentation

6.45.1.1 getTexture()

```
sf::Texture& TextureManager::getTexture (
            const std::string & filePath )  [inline]
```

Retrieves a texture from the texture manager.

This function attempts to find the texture associated with the given file path in the texture manager. If the texture is found, it is returned. Otherwise, a new texture is loaded from the file path and added to the texture manager before being returned.

Exceptions

failedToLoadTexture	If the texture fails to load from the file path.

Parameters

filePath	The file path of the texture to retrieve.

Returns

sf::Texture& A reference to the retrieved texture.

6.45.2 Member Data Documentation

6.45.2.1 textures

```
std::unordered_map<std::string, sf::Texture> TextureManager::textures  [private]
```

A container for storing textures with string keys.

This unordered map allows you to associate a string key with an sf::Texture object. It provides fast access to textures based on their keys.

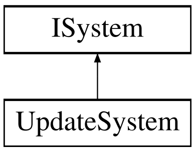
The documentation for this class was generated from the following file:

- /home/runner/work/R-Type/R-Type/ECS/Interface/Include/texture_manager.hpp

6.46 UpdateSystem Class Reference

```
#include <update_system.hpp>
```

Inheritance diagram for UpdateSystem:



Public Member Functions

- UpdateSystem (sf::RenderWindow &window, ComponentManager &componentManager, EntityManager &entityManager)
- void update (float deltaTime) override
- void updateSpritePositions (ComponentManager &componentManager, EntityManager &entityManager)

Private Attributes

- sf::RenderWindow & _window
- ComponentManager & _componentManager
- EntityManager & _entityManager

6.46.1 Constructor & Destructor Documentation

6.46.1.1 UpdateSystem()

```
UpdateSystem::UpdateSystem (
             sf::RenderWindow & window,
             ComponentManager & componentManager,
             EntityManager & entityManager )  [inline]
```

6.46.2 Member Function Documentation

6.46.2.1 update()

```
void UpdateSystem::update (
             float deltaTime )  [inline], [override], [virtual]
```

Implements ISystem.

6.46.2.2 updateSpritePositions()

```
void UpdateSystem::updateSpritePositions (
            ComponentManager & componentManager,
            EntityManager & entityManager )
```

6.46.3 Member Data Documentation

6.46.3.1 _componentManager

```
ComponentManager& UpdateSystem::_componentManager  [private]
```

6.46.3.2 _entityManager

```
EntityManager& UpdateSystem::_entityManager  [private]
```

6.46.3.3 _window

```
sf::RenderWindow& UpdateSystem::_window  [private]
```

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

- /home/runner/work/R-Type/R-Type/ECS/Interface/Include/Systems/update_system.hpp
- /home/runner/work/R-Type/R-Type/ECS/Src/Systems/update_system.cpp

6.47 Vector< T > Struct Template Reference

```
#include <sprite_data_component.hpp>
```

Public Attributes

- T x
- T y

6.47.1 Member Data Documentation

6.47.1.1 x

```
template<typename T >
T Vector< T >::x
```

6.47.1.2 y

```
template<typename T >
T Vector< T >::y
```

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

- /home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/sprite_data_component.hpp

6.48 VelocityComponent Struct Reference

```
#include <velocity_component.hpp>
```

Public Attributes

- float speed

6.48.1 Member Data Documentation

6.48.1.1 speed

```
float VelocityComponent::speed
```

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

- /home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/velocity_component.hpp

6.49 vf2d Struct Reference

Represents a 2D vector with x and y coordinates.

```
#include <entity_struct.hpp>
```

Public Attributes

- float x = 0
- float y = 0

6.49.1 Detailed Description

Represents a 2D vector with x and y coordinates.

6.49.2 Member Data Documentation

6.49.2.1 x

```
float vf2d::x = 0
```

6.49.2.2 y

```
float vf2d::y = 0
```

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

- /home/runner/work/R-Type/R-Type/ECS/Interface/Include/entity_struct.hpp

6.50 WeaponComponent Struct Reference

```
#include <weapon_component.hpp>
```

Public Attributes

- float damage
- float fire_rate
- float bullet_speed
- float bullet_lifetime

6.50.1 Member Data Documentation

6.50.1.1 bullet_lifetime

```
float WeaponComponent::bullet_lifetime
```

6.50.1.2 bullet_speed

```
float WeaponComponent::bullet_speed
```

6.50.1.3 damage

```
float WeaponComponent::damage
```

6.50.1.4 fire_rate

```
float WeaponComponent::fire_rate
```

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

- /home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/weapon_component.hpp

Chapter 7

File Documentation

7.1 /home/runner/work/R-Type/R-Type/Client/Interface/↩ Include/mainmenu.hpp File Reference

```
#include <SFML/Graphics.hpp>
#include <r_type_client.hpp>
```

Functions

- int MainMenu (sf::RenderWindow ∗window, Rtype ∗rtype)

7.1.1 Function Documentation

7.1.1.1 MainMenu()

```
int MainMenu (
            sf::RenderWindow ∗ window,
            Rtype ∗ rtype )
```

7.2 /home/runner/work/R-Type/R-Type/Client/Interface/Include/Net/a_↩ client.hpp File Reference

```
#include <Components/component_manager.hpp>
#include <Components/components.hpp>
#include <Net/i_client.hpp>
#include <entity_struct.hpp>
#include <texture_manager.hpp>
#include <unordered_map>
```

Classes

- class r_type::net::AClient< T >

Namespaces

- r_type
- r_type::net

7.3 /home/runner/work/R-Type/R-Type/Client/Interface/Include/↩ Net/client.hpp File Reference

```
#include <Net/a_client.hpp>
#include <SFML/Graphics.hpp>
#include <iostream>
```

Classes

- class r_type::net::Client

Namespaces

- r_type
- r_type::net

7.4 /home/runner/work/R-Type/R-Type/Client/Interface/Include/Net/i_↩ client.hpp File Reference

```
#include <Net/common.hpp>
#include <Net/connection.hpp>
#include <Net/thread_safe_queue.hpp>
```

Classes

- class r_type::net::IClient< T >

Namespaces

- r_type
- r_type::net

7.5 /home/runner/work/R-Type/R-Type/Client/Interface/Include/r_type_↩ client.hpp File Reference

```
#include "error_handling.hpp"
#include "scenes.hpp"
#include <SFML/Graphics.hpp>
#include <SFML/Window.hpp>
```

Classes

- class Rtype

  The Rtype class handles the main game logic, including initialization, main menu, game loop, event handling, server message processing, game updating, and rendering.

7.6 /home/runner/work/R-Type/R-Type/Client/Interface/↩ Include/scenes.hpp File Reference

```
#include "Entities/entity.hpp"
#include <SFML/Graphics.hpp>
#include <a_scenes.hpp>
#include <memory>
#include <vector>
```

Classes

- class Scenes

  Represents a class that manages different scenes in a game.

7.7 /home/runner/work/R-Type/R-Type/Client/Src/main.cpp File Reference

```
#include <r_type_client.hpp>
```

Functions

- int main ()

  The entry point of the program.

7.7.1 Function Documentation

7.7.1.1 main()

```
int main ( )
```

The entry point of the program.

This function initializes the Rtype object and runs the game.

Returns

0 indicating successful program execution.

int

7.8 /home/runner/work/R-Type/R-Type/Server/Src/main.cpp File Reference

```
#include <Net/server.hpp>
#include <iostream>
```

Functions

- int main ()

7.8.1 Function Documentation

7.8.1.1 main()

```
int main ( )
```

7.9 /home/runner/work/R-Type/R-Type/Client/Src/r-type_client.cpp File Reference

```
#include <Components/component_manager.hpp>
#include <Entities/entity_factory.hpp>
#include <Entities/entity_manager.hpp>
#include <Systems/systems.hpp>
#include <iostream>
#include <r_type_client.hpp>
#include <texture_manager.hpp>
```

7.10 /home/runner/work/R-Type/R-Type/Client/Src/scenes.cpp File Reference

```
#include <Components/component_manager.hpp>
#include <Components/components.hpp>
#include <Entities/entity_factory.hpp>
#include <Entities/entity_manager.hpp>
#include <Net/client.hpp>
#include <Systems/system_manager.hpp>
#include <Systems/systems.hpp>
#include <creatable_client_object.hpp>
#include <functional>
#include <iostream>
#include <r_type_client.hpp>
#include <scenes.hpp>
#include <texture_manager.hpp>
```

Functions

- void handleEvents (sf::Event event, ComponentManager &componentManager, sf::RenderWindow ∗_↩ window, std::vector< std::shared_ptr< Entity >> buttons, Scenes ∗scenes)

  Handles events for the scene, including window close and mouse button press events.
- void createDaltonismChoiceButtons (std::vector< std::shared_ptr< Entity >> &buttons, ComponentManager &componentManager, EntityManager &entityManager, TextureManager &textureManager, EntityFactory &entityFactory)
- void createGameModeChoiceButtons (std::vector< std::shared_ptr< Entity >> &buttons, ComponentManager &componentManager, EntityManager &entityManager, TextureManager &textureManager, EntityFactory &entityFactory)
- sf::Keyboard::Key waitForKey (sf::RenderWindow ∗_window)
- void createKeyBindingButtons (std::vector< std::shared_ptr< Entity >> &buttons, ComponentManager &componentManager, EntityManager &entityManager, TextureManager &textureManager, EntityFactory &entityFactory)

7.10.1 Function Documentation

7.10.1.1 createDaltonismChoiceButtons()

```
void createDaltonismChoiceButtons (
            std::vector< std::shared_ptr< Entity >> & buttons,
            ComponentManager & componentManager,
            EntityManager & entityManager,
            TextureManager & textureManager,
            EntityFactory & entityFactory )
```

7.10.1.2 createGameModeChoiceButtons()

```
void createGameModeChoiceButtons (
            std::vector< std::shared_ptr< Entity >> & buttons,
            ComponentManager & componentManager,
            EntityManager & entityManager,
            TextureManager & textureManager,
            EntityFactory & entityFactory )
```

7.10.1.3 createKeyBindingButtons()

```
void createKeyBindingButtons (
            std::vector< std::shared_ptr< Entity >> & buttons,
            ComponentManager & componentManager,
            EntityManager & entityManager,
            TextureManager & textureManager,
            EntityFactory & entityFactory )
```

7.10.1.4 handleEvents()

```
void handleEvents (
            sf::Event event,
            ComponentManager & componentManager,
            sf::RenderWindow * _window,
            std::vector< std::shared_ptr< Entity >> buttons,
            Scenes * scenes )
```

Handles events for the scene, including window close and mouse button press events.

This function processes events from the given RenderWindow and performs actions based on the type of event. It handles window close events and mouse button press events. For mouse button press events, it checks if the left mouse button was pressed and if the click occurred within the bounds of any button entities. If a button is clicked, it triggers the associated OnClickComponent or BindComponent actions.

Parameters

event	The event to handle.
componentManager	Reference to the ComponentManager to access components of entities.
_window	Pointer to the RenderWindow where events are polled from.
buttons	Vector of shared pointers to Entity objects representing buttons.

7.10.1.5 waitForKey()

```
sf::Keyboard::Key waitForKey (
            sf::RenderWindow * _window )
```

7.11 /home/runner/work/R-Type/R-Type/ECS/Interface/Include/a_↩ scenes.hpp File Reference

```
#include "Entities/entity.hpp"
#include "i_scenes.hpp"
#include <memory>
```

Classes

- class AScenes

7.12 /home/runner/work/R-Type/R-Type/ECS/Interface/Include/↩ Components/ally_component.hpp File Reference

Classes

- struct AllyComponent

7.13 /home/runner/work/R-Type/R-Type/ECS/Interface/Include/↩ Components/ally_missile_component.hpp File Reference

Classes

- struct AllyMissileComponent

7.14 /home/runner/work/R-Type/R-Type/ECS/Interface/Include/↩ Components/background_component.hpp File Reference

Classes

- struct BackgroundComponent

7.15 /home/runner/work/R-Type/R-Type/ECS/Interface/Include/↩ Components/basic_monster_component.hpp File Reference

Classes

- struct BasicMonsterComponent

7.16 /home/runner/work/R-Type/R-Type/ECS/Interface/Include/↩ Components/bind_component.hpp File Reference

```
#include "i_scenes.hpp"
#include "a_scenes.hpp"
#include <functional>
```

Classes

- struct BindComponent

7.17 /home/runner/work/R-Type/R-Type/ECS/Interface/Include/↩ Components/component_manager.hpp File Reference

```
#include "components.hpp"
#include "texture_manager.hpp"
#include <any>
#include <iostream>
#include <memory>
#include <optional>
#include <typeindex>
#include <unordered_map>
```

Classes

- class ComponentManager

  Manages the components of entities in an ECS system.

7.18 /home/runner/work/R-Type/R-Type/ECS/Interface/Include/↩ Components/components.hpp File Reference

```
#include "ally_component.hpp"
#include "ally_missile_component.hpp"
#include "background_component.hpp"
#include "basic_monster_component.hpp"
#include "bind_component.hpp"
#include "enemy_component.hpp"
#include "enemy_missile_component.hpp"
#include "health_component.hpp"
#include "hitbox_component.hpp"
#include "input_component.hpp"
#include "offset_component.hpp"
#include "on_click_component.hpp"
#include "player_component.hpp"
#include "player_missile_component.hpp"
#include "position_component.hpp"
```

```
#include "score_component.hpp"
#include "sprite_component.hpp"
#include "sprite_data_component.hpp"
#include "text_component.hpp"
#include "velocity_component.hpp"
#include "weapon_component.hpp"
```

7.19 /home/runner/work/R-Type/R-Type/ECS/Interface/Include/↩ Components/enemy_component.hpp File Reference

Classes

- struct EnemyComponent

7.20 /home/runner/work/R-Type/R-Type/ECS/Interface/Include/↩ Components/enemy_missile_component.hpp File Reference

Classes

- struct EnemyMissileComponent

7.21 /home/runner/work/R-Type/R-Type/ECS/Interface/Include/↩ Components/health_component.hpp File Reference

Classes

- struct HealthComponent

7.22 /home/runner/work/R-Type/R-Type/ECS/Interface/Include/↩ Components/hitbox_component.hpp File Reference

Classes

- struct HitboxComponent

7.23 /home/runner/work/R-Type/R-Type/ECS/Interface/Include/↩ Components/input_component.hpp File Reference

Classes

- struct InputComponent

Enumerations

- enum class InputType {
  UP , DOWN , LEFT , RIGHT ,
  SHOOT , QUIT , NONE }

7.23.1 Enumeration Type Documentation

7.23.1.1 InputType

```
enum InputType  [strong]
```

Enumerator

UP	
DOWN	
LEFT	
RIGHT	
SHOOT	
QUIT	
NONE	

7.24 /home/runner/work/R-Type/R-Type/ECS/Interface/Include/↩ Components/label_component.hpp File Reference

```
#include <iostream>
```

Classes

- struct labelComponent

7.25 /home/runner/work/R-Type/R-Type/ECS/Interface/Include/↩ Components/offset_component.hpp File Reference

Classes

- struct OffsetComponent

7.26 /home/runner/work/R-Type/R-Type/ECS/Interface/Include/↩ Components/on_click_component.hpp File Reference

```
#include <functional>
#include <i_scenes.hpp>
#include <a_scenes.hpp>
```

Classes

- struct OnClickComponent

7.27 /home/runner/work/R-Type/R-Type/ECS/Interface/Include/↩ Components/player_component.hpp File Reference

Classes

- struct PlayerComponent

7.28 /home/runner/work/R-Type/R-Type/ECS/Interface/Include/↩ Components/player_missile_component.hpp File Reference

Classes

- struct PlayerMissileComponent

7.29 /home/runner/work/R-Type/R-Type/ECS/Interface/Include/↩ Components/position_component.hpp File Reference

Classes

- struct PositionComponent

7.30 /home/runner/work/R-Type/R-Type/ECS/Interface/Include/↩ Components/score_component.hpp File Reference

Classes

- struct ScoreComponent

7.31 /home/runner/work/R-Type/R-Type/ECS/Interface/Include/↩ Components/sprite_component.hpp File Reference

```
#include "../error_handling.hpp"
#include "position_component.hpp"
#include <SFML/Graphics.hpp>
#include <string>
```

Classes

- struct SpriteComponent

7.32 /home/runner/work/R-Type/R-Type/ECS/Interface/Include/↩ Components/sprite_data_component.hpp File Reference

```
#include "../error_handling.hpp"
#include "../sprite_path.hpp"
#include "position_component.hpp"
#include <SFML/Graphics.hpp>
#include <cstdint>
#include <string>
```

Classes

- struct Vector< T >
- struct SpriteDataComponent

7.33 /home/runner/work/R-Type/R-Type/ECS/Interface/Include/↩ Components/text_component.hpp File Reference

```
#include <iostream>
```

Classes

- struct TextComponent

7.34 /home/runner/work/R-Type/R-Type/ECS/Interface/Include/↩ Components/velocity_component.hpp File Reference

Classes

- struct VelocityComponent

7.35 /home/runner/work/R-Type/R-Type/ECS/Interface/Include/↩ Components/weapon_component.hpp File Reference

Classes

- struct WeaponComponent

7.36 /home/runner/work/R-Type/R-Type/ECS/Interface/Include/creatable↩ _client_object.hpp File Reference

```
#include <cstdint>
```

Enumerations

- enum class CreatableClientObject : uint32_t { MISSILE , NONE }

7.36.1 Enumeration Type Documentation

7.36.1.1 CreatableClientObject

```
enum CreatableClientObject : uint32_t [strong]
```

Enumerator

MISSILE	
NONE	

7.37 /home/runner/work/R-Type/R-Type/ECS/Interface/Include/↩ Entities/entity.hpp File Reference

```
#include "i_entity.hpp"
```

Classes

- class Entity

  Represents an entity in the ECS system.

7.38 /home/runner/work/R-Type/R-Type/ECS/Interface/Include/↩ Entities/entity_factory.hpp File Reference

```
#include "i_entity_factory.hpp"
#include "i_scenes.hpp"
#include "a_scenes.hpp"
#include <functional>
```

Classes

- class EntityFactory

  A class responsible for creating different types of entities.

7.39 /home/runner/work/R-Type/R-Type/ECS/Interface/Include/↩ Entities/entity_manager.hpp File Reference

```
#include "../error_handling.hpp"
#include "entity.hpp"
#include <algorithm>
#include <vector>
```

Classes

- class EntityManager

  Class responsible for managing entities in the ECS system.

7.40 /home/runner/work/R-Type/R-Type/ECS/Interface/Include/Entities/i↩ _entity.hpp File Reference

Classes

- class IEntity

  The IEntity class represents an entity in the system.

7.41 /home/runner/work/R-Type/R-Type/ECS/Interface/Include/Entities/i↩ _entity_factory.hpp File Reference

```
#include "Components/component_manager.hpp"
#include "entity.hpp"
#include "entity_manager.hpp"
#include "texture_manager.hpp"
```

Classes

- class IEntityFactory

  The interface for an entity factory.

7.42 /home/runner/work/R-Type/R-Type/ECS/Interface/Include/entity_↵ struct.hpp File Reference

```
#include "Components/sprite_data_component.hpp"
#include <cstdint>
```

Classes

- struct vf2d

  Represents a 2D vector with x and y coordinates.
- struct EntityInformation

  Represents information about an entity.

7.43 /home/runner/work/R-Type/R-Type/ECS/Interface/Include/error_↵ handling.hpp File Reference

```
#include <exception>
```

Classes

- class componentNotFound

  Exception class for when a component is not found.
- class entityNotFound

  Exception class for entity not found error.
- class failedToLoadTexture

  Exception class for failed texture loading.

7.44 /home/runner/work/R-Type/R-Type/ECS/Interface/Include/i_↵ scenes.hpp File Reference

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

Classes

- class IScenes

  Interface for managing different scenes in a game.

7.45 /home/runner/work/R-Type/R-Type/ECS/Interface/Include/sprite_↩ path.hpp File Reference

```
#include <cstdint>
#include <string>
```

Enumerations

- enum class SpritePath : uint32_t {
  Ship1 , Ship2 , Ship3 , Ship4 ,
  Enemy1 , Enemy2 , Enemy3 , Enemy4 ,
  Enemy5 , Enemy6 , Monster1 , Monster2 ,
  Monster3 , Monster4 , Monster5 , Missile ,
  Background , Explosion , PowerUp , Boss ,
  BossBullet , NONE }

Functions

- std::string SpriteFactory (SpritePath sprite)

7.45.1 Enumeration Type Documentation

7.45.1.1 SpritePath

```
enum SpritePath : uint32_t [strong]
```

Enumerator

Ship1	
Ship2	
Ship3	
Ship4	
Enemy1	
Enemy2	
Enemy3	
Enemy4	
Enemy5	
Enemy6	
Monster1	
Monster2	
Monster3	
Monster4	
Monster5	
Missile	
Background	
Explosion	
PowerUp	
Boss	
BossBullet	
NONE	

7.45.2 Function Documentation

7.45.2.1 SpriteFactory()

```
std::string SpriteFactory (
            SpritePath sprite )
```

7.46 /home/runner/work/R-Type/R-Type/ECS/Interface/Include/↩ Systems/button_system.hpp File Reference

7.47 /home/runner/work/R-Type/R-Type/ECS/Interface/Include/↩ Systems/i_system.hpp File Reference

```
#include "Components/component_manager.hpp"
#include "Entities/entity_manager.hpp"
#include <SFML/Graphics.hpp>
```

Classes

- class ISystem

7.48 /home/runner/work/R-Type/R-Type/ECS/Interface/Include/↩ Systems/render_system.hpp File Reference

```
#include "Systems/i_system.hpp"
```

Classes

- class RenderSystem

7.49 /home/runner/work/R-Type/R-Type/ECS/Interface/Include/↩ Systems/system_manager.hpp File Reference

```
#include "i_system.hpp"
#include "systems.hpp"
```

Classes

- class SystemManager

7.50 /home/runner/work/R-Type/R-Type/ECS/Interface/Include/↩ Systems/systems.hpp File Reference

```
#include "render_system.hpp"
#include "update_system.hpp"
```

7.51 /home/runner/work/R-Type/R-Type/ECS/Interface/Include/↩ Systems/update_system.hpp File Reference

```
#include "Systems/i_system.hpp"
```

Classes

- class UpdateSystem

7.52 /home/runner/work/R-Type/R-Type/ECS/Interface/Include/texture_↩ manager.hpp File Reference

```
#include "error_handling.hpp"
#include <SFML/Graphics.hpp>
#include <string>
#include <unordered_map>
```

Classes

- class TextureManager

7.53 /home/runner/work/R-Type/R-Type/ECS/Src/a_scenes.cpp File Reference

```
#include <a_scenes.hpp>
```

7.54 /home/runner/work/R-Type/R-Type/ECS/Src/Entities/entity_↩ factory.cpp File Reference

```
#include "hitbox_tmp.hpp"
#include <Components/components.hpp>
#include <Entities/entity_factory.hpp>
#include <SFML/Graphics.hpp>
#include <cstdint>
#include <cstdlib>
```

Functions

- bool CheckPositionEntity (EntityManager &entityManager, ComponentManager &componentManager, u_↩ int32_t entityID)

7.54.1 Function Documentation

7.54.1.1 CheckPositionEntity()

```
bool CheckPositionEntity (
            EntityManager & entityManager,
            ComponentManager & componentManager,
            u_int32_t entityID )
```

7.55 /home/runner/work/R-Type/R-Type/ECS/Src/hitbox_tmp.cpp File Reference

```
#include "hitbox_tmp.hpp"
```

Functions

- static int CheckCollisionLogic (float descLeft, float descRight, float descTop, float descBottom, ComponentManager componentManager, EntityManager entityManager, int entityId)
- int CheckEntityPosition (uint32_t entityId, ComponentManager componentManager, EntityManager entity↩ Manager)
- int CheckEntityMovement (EntityInformation desc, ComponentManager componentManager, EntityManager entityManager)

7.55.1 Function Documentation

7.55.1.1 CheckCollisionLogic()

```
static int CheckCollisionLogic (
             float descLeft,
             float descRight,
             float descTop,
             float descBottom,
             ComponentManager componentManager,
             EntityManager entityManager,
             int entityId )  [static]
```

7.55.1.2 CheckEntityMovement()

```
int CheckEntityMovement (
             EntityInformation desc,
             ComponentManager componentManager,
             EntityManager entityManager )
```

7.55.1.3 CheckEntityPosition()

```
int CheckEntityPosition (
             uint32_t entityId,
             ComponentManager componentManager,
             EntityManager entityManager )
```

7.56 /home/runner/work/R-Type/R-Type/ECS/Src/sprite_path.cpp File Reference

`#include <sprite_path.hpp>`

Functions

- std::string SpriteFactory (SpritePath sprite)

7.56.1 Function Documentation

7.56.1.1 SpriteFactory()

```
std::string SpriteFactory (
             SpritePath sprite )
```

7.57 /home/runner/work/R-Type/R-Type/ECS/Src/Systems/render_↩ system.cpp File Reference

```
#include <Systems/render_system.hpp>
```

7.58 /home/runner/work/R-Type/R-Type/ECS/Src/Systems/update_↩ system.cpp File Reference

```
#include "Systems/update_system.hpp"
```

7.59 /home/runner/work/R-Type/R-Type/Server/Interface/Include/Net/a_↩ server.hpp File Reference

```
#include "hitbox_tmp.hpp"
#include <Components/component_manager.hpp>
#include <Components/components.hpp>
#include <Entities/entity_factory.hpp>
#include <Entities/entity_manager.hpp>
#include <Net/i_server.hpp>
#include <cmath>
#include <entity_struct.hpp>
#include <unordered_map>
```

Classes

- class r_type::net::AServer< T >

  AServer class template for managing server operations.

Namespaces

- r_type
- r_type::net

7.60 /home/runner/work/R-Type/R-Type/Server/Interface/Include/↩ Net/server.hpp File Reference

```
#include "a_server.hpp"
```

Classes

- class r_type::net::Server

Namespaces

- r_type
- r_type::net

7.61 /home/runner/work/R-Type/R-Type/Server/Interface/Include/r_type↩ _server.hpp File Reference

```
#include <iostream>
```

7.62 /home/runner/work/R-Type/R-Type/Server/Src/server.cpp File Reference

```
#include <Net/server.hpp>
#include <creatable_client_object.hpp>
```

Index