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

Classes

- class Level

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.

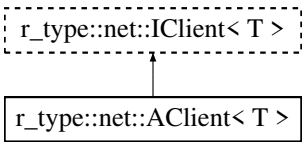
The documentation for this class was generated from the following file:

- /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 ()

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 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 >.

6.2.2.2 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 >.

6.2.2.3 getConnection()

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

6.2.2.4 getPlayerId()

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

6.2.2.5 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 >.

6.2.2.6 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 >.

6.2.2.7 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 >.

6.2.2.8 setPlayerId()

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

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 AnimationComponent Struct Reference

`#include <animation_component.hpp>`

Public Member Functions

- AnimationComponent (vf2d _offset, vf2d _dimension)

Public Attributes

- vf2d offset
- vf2d dimension

6.5.1 Constructor & Destructor Documentation

6.5.1.1 AnimationComponent()

```
AnimationComponent::AnimationComponent (
            vf2d _offset,
            vf2d _dimension )  [inline]
```

6.5.2 Member Data Documentation

6.5.2.1 dimension

```
vf2d AnimationComponent::dimension
```

6.5.2.2 offset

```
vf2d AnimationComponent::offset
```

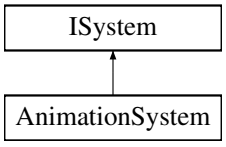
The documentation for this struct was generated from the following file:

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

6.6 AnimationSystem Class Reference

```
#include <animation_system.hpp>
```

Inheritance diagram for AnimationSystem:



Public Member Functions

- AnimationSystem (ComponentManager &componentManager, EntityManager &entityManager)
- void AnimationEntities (ComponentManager &componentManager, EntityManager &entityManager, float deltaTime)

  Animates entities.
- void animatePlayer (std::optional< VelocityComponent * > &velocity, std::optional< AnimationComponent * > &animation)
- void animateBasicMonster (std::optional< AnimationComponent * > &animation)
- void animateWeapon (std::optional< AnimationComponent * > &animation)

Private Attributes

- ComponentManager & _componentManager

  Reference to the ComponentManager instance.
- EntityManager & _entityManager

  Reference to the EntityManager instance.

6.6.1 Constructor & Destructor Documentation

6.6.1.1 AnimationSystem()

```
AnimationSystem::AnimationSystem (
            ComponentManager & componentManager,
            EntityManager & entityManager )  [inline]
```

6.6.2 Member Function Documentation

6.6.2.1 animateBasicMonster()

```
void AnimationSystem::animateBasicMonster (
            std::optional< AnimationComponent * > & animation )
```

6.6.2.2 animatePlayer()

```
void AnimationSystem::animatePlayer (
            std::optional< VelocityComponent * > & velocity,
            std::optional< AnimationComponent * > & animation )
```

6.6.2.3 animateWeapon()

```
void AnimationSystem::animateWeapon (
            std::optional< AnimationComponent * > & animation )
```

6.6.2.4 AnimationEntities()

```
void AnimationSystem::AnimationEntities (
            ComponentManager & componentManager,
            EntityManager & entityManager,
            float deltaTime )
```

Animates entities.

Updates the animation states of entities based on their components.

This function animates entities based on their animation components. It processes each entity in the entity manager and updates their animation based on the delta time provided.

Parameters

componentManager	The component manager used to access entity components.
entityManager	The entity manager used to access entities.
deltaTime	The time elapsed since the last update, used to update animations.

This function iterates through all entities and updates their animation states based on the presence and values of specific components such as AnimationComponent, PlayerComponent, VelocityComponent, and BackgroundComponent.

Parameters

componentManager	Reference to the ComponentManager that handles components.
entityManager	Reference to the EntityManager that handles entities.
deltaTime	The time elapsed since the last update, used for time-based animations.

6.6.3 Member Data Documentation

6.6.3.1 _componentManager

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

Reference to the ComponentManager instance.

This member variable holds a reference to the ComponentManager, which is responsible for managing all the components within the ECS (Entity Component System). It provides functionality to add, remove, and query components associated with entities.

6.6.3.2 _entityManager

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

Reference to the EntityManager instance.

This member variable holds a reference to the EntityManager, which is responsible for managing all entities within the ECS (Entity Component System). It provides functionalities such as entity creation, deletion, and retrieval.

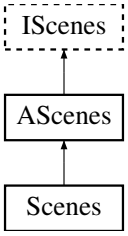
The documentation for this class was generated from the following files:

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

6.7 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.
- enum class SpriteType {
  BACKGROUND , PLAYER , ALLY , ENEMY ,
  FILTER , WEAPON , POWER_UP , UI ,
  OTHER }

  Enumeration representing the type of sprite in the game.

Public Member Functions

- AScenes (std::string ip, int port)
- ~AScenes ()=default
- void setScene (Scene scene)

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

  Get the Previous Scene object.
- DaltonismMode getDaltonism () const

  Get the Daltonism 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
- void setIp (std::string ip)
- void setPort (int port)
- std::string getIp () const
- int getPort () const

Public Attributes

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

  A map that binds game actions to specific keyboard keys.
- std::vector< std::shared_ptr< Entity > > buttons
- std::shared_ptr< Entity > filter

Protected Attributes

- GameMode _currentGameMode = GameMode::MEDIUM
- DaltonismMode _currentDaltonismMode = DaltonismMode::NORMAL
- Scene _currentScene = Scene::MAIN_MENU
- Scene _previousScene = Scene::MAIN_MENU
- bool _displayDaltonismChoice = false
- bool _displayGameModeChoice = false
- bool _displayKeyBindsChoice = false
- std::string _ip

  The IP address of the server.
- int _port

  The port number of the server.

6.7.1 Member Enumeration Documentation

6.7.1.1 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	

6.7.1.2 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.

6.7.1.3 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	

6.7.1.4 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.

6.7.1.5 SpriteType

`enum AScenes::SpriteType [strong]`

Enumeration representing the type of sprite in the game.

This enumeration defines the different sprite types that need to be identified in the game. The types include:

- BACKGROUND: Represents a background sprite.
- PLAYER: Represents a player sprite.
- ALLY: Represents an ally sprite.
- ENEMY: Represents an enemy sprite.
- OTHER: Represents any other type of sprite.

Enumerator

BACKGROUND	
PLAYER	
ALLY	
ENEMY	
FILTER	
WEAPON	
POWER_UP	
UI	
OTHER	

6.7.2 Constructor & Destructor Documentation

6.7.2.1 AScenes()

```
AScenes::AScenes (
            std::string ip,
            int port )
```

6.7.2.2 ~AScenes()

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

6.7.3 Member Function Documentation

6.7.3.1 getDaltonism()

```
DaltonismMode AScenes::getDaltonism ( ) const  [inline]
```

Get the Daltonism object.

Returns

DaltonismMode

6.7.3.2 getDisplayDaltonismChoice()

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

6.7.3.3 getDisplayGameModeChoice()

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

6.7.3.4 getDisplayKeyBindsChoice()

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

6.7.3.5 getIp()

```
std::string AScenes::getIp ( ) const
```

6.7.3.6 getPort()

```
int AScenes::getPort ( ) const
```

6.7.3.7 getPreviousScene()

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

Get the Previous Scene object.

Returns

Scene

6.7.3.8 setDaltonism()

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

Set the Daltonism object.

Parameters

mode	The daltonism mode to set

6.7.3.9 setDisplayDaltonismChoice()

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

6.7.3.10 setDisplayGameModeChoice()

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

6.7.3.11 setDisplayKeyBindsChoice()

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

6.7.3.12 setGameMode()

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

Set the Game Mode object.

Parameters

mode	

6.7.3.13 setIp()

```
void AScenes::setIp (
            std::string ip )
```

6.7.3.14 setPort()

```
void AScenes::setPort (
            int port )
```

6.7.3.15 setScene()

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

Set the Scene object.

Parameters

scene	

6.7.4 Member Data Documentation

6.7.4.1 _currentDaltonismMode

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

6.7.4.2 _currentGameMode

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

6.7.4.3 _currentScene

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

6.7.4.4 _displayDaltonismChoice

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

6.7.4.5 _displayGameModeChoice

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

6.7.4.6 _displayKeyBindsChoice

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

6.7.4.7 _ip

```
std::string AScenes::_ip  [protected]
```

The IP address of the server.

This member variable stores the IP address of the server to which the client will connect. It is a string that contains the IP address in the format "xxx.xxx.xxx.xxx".

6.7.4.8 _port

```
int AScenes::_port  [protected]
```

The port number of the server.

This member variable stores the port number of the server to which the client will connect. It is an integer that represents the port number on which the server is listening for incoming connections.

6.7.4.9 _previousScene

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

6.7.4.10 buttons

```
std::vector<std::shared_ptr<Entity> > AScenes::buttons
```

6.7.4.11 filter

```
std::shared_ptr<Entity> AScenes::filter
```

6.7.4.12 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:

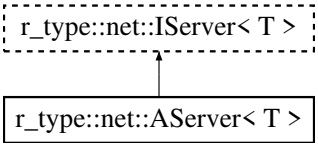
- /home/runner/work/R-Type/R-Type/ECS/Interface/Include/a_scenes.hpp
- /home/runner/work/R-Type/R-Type/ECS/Src/a_scenes.cpp

6.8 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.
- UIEntityInformation UpdateInfoBar (int playerId)
- void Update (size_t nMaxMessages=-1, bool bWait=false)

  Updates the game level based on the provided clock time.
- void UpdatePlayerPosition (PlayerMovement direction, uint32_t entityId) override

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

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

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

  Removes entities associated with a player.
- void RemoveInfoBar (uint32_t infoBarId)
- EntityInformation InitiatePlayer (int clientId)

  Initializes a new player entity and assigns a random position.
- UIEntityInformation InitInfoBar (int clientId)
- EntityInformation FormatEntityInformation (uint32_t entityId)

  Formats the information of a given entity into an EntityInformation structure.
- EntityInformation InitiatePlayerMissile (int entityId)

  Initializes a missile entity associated with a player.
- EntityInformation InitiateEnemyMissile (int enemyId)
- EntityInformation InitiateWeaponForce (int entityId)
- EntityInformation InitiateBackground ()

  Initializes a background entity.
- std::shared_ptr< Connection< T > > getClientById (const std::deque< std::shared_ptr< Connection< T >>> &connections, uint32_t clientId)
- virtual void OnClientValidated (std::shared_ptr< Connection< T >> client)

  Callback function that is called when a client has been successfully validated.
- ComponentManager GetComponentManager () override

  Retrieves the component manager associated with the server.
- EntityManager & GetEntityManager () override

  Retrieves the entity manager associated with the server.
- EntityFactory & GetEntityFactory () override

  Retrieves the entity factory associated with the server.

- std::chrono::system_clock::time_point GetClock () override

  Retrieves the current clock time of the server.
- void SetClock (std::chrono::system_clock::time_point clock)

  Set the Clock object.

Public Attributes

- ThreadSafeQueue< OwnedMessage< T > > _qMessagesIn

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

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

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

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

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

  Represents the endpoint of a client in a UDP connection.
- std::array< uint8_t, 1024 > _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.
- std::unordered_map< uint32_t, uint32_t > _clientInfoBarID
- 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.
- bool _playerConnected = false
- EntityInformation _background

  Holds information about the background entity.
- int _port
- r_type::Level< T > _level

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

6.8.1 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.

6.8.2 Constructor & Destructor Documentation

6.8.2.1 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.

6.8.2.2 ~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.

6.8.3 Member Function Documentation

6.8.3.1 FormatEntityInformation()

```
template<typename T >
EntityInformation r_type::net::AServer< T >::FormatEntityInformation (
            uint32_t entityId )  [inline]
```

Formats the information of a given entity into an EntityInformation structure.

This function retrieves the position and sprite data components of the specified entity and populates an EntityInformation structure with this data. If the entity has both position and sprite data components, their values are copied into the EntityInformation structure. If either component is missing, the EntityInformation structure will be returned with default values.

Parameters

entity	The entity whose information is to be formatted.

Returns

EntityInformation The formatted information of the entity.

6.8.3.2 getClientById()

```
template<typename T >
std::shared_ptr<Connection<T> > r_type::net::AServer< T >::getClientById (
            const std::deque< std::shared_ptr< Connection< T >>> & connections,
            uint32_t clientId )  [inline]
```

6.8.3.3 GetClientInfoBarId()

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

6.8.3.4 GetClientPlayerId()

```
template<typename T >
uint32_t r_type::net::AServer< T >::GetClientPlayerId (
            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.

6.8.3.5 GetClock()

```
template<typename T >
std::chrono::system_clock::time_point r_type::net::AServer< T >::GetClock ( )  [inline], [override]
```

Retrieves the current clock time of the server.

This function returns the current time point of the server's clock, which can be used for time-related calculations, such as updating game state, handling animations, or scheduling events. It provides a consistent reference point for the server's operations.

Returns

std::chrono::system_clock::time_point The current time point of the server's clock.

6.8.3.6 GetComponentManager()

```
template<typename T >
ComponentManager r_type::net::AServer< T >::GetComponentManager ( )  [inline], [override]
```

Retrieves the component manager associated with the server.

This function provides access to the component manager, which is responsible for managing the components associated with entities in the game. It allows for the retrieval and manipulation of entity components, enabling the game logic to interact with them as needed.

Returns

ComponentManager& A reference to the component manager instance.

6.8.3.7 GetEntityFactory()

```
template<typename T >
EntityFactory& r_type::net::AServer< T >::GetEntityFactory ( )  [inline], [override]
```

Retrieves the entity factory associated with the server.

This function provides access to the entity factory, which is responsible for creating new entities in the game. The entity factory provides methods to instantiate various types of entities, such as players, missiles, and background elements, ensuring that they are correctly initialized with the necessary components.

Returns

EntityFactory& A reference to the entity factory instance.

6.8.3.8 GetEntityManager()

```
template<typename T >
EntityManager& r_type::net::AServer< T >::GetEntityManager ( )  [inline], [override]
```

Retrieves the entity manager associated with the server.

This function returns the entity manager responsible for creating, managing, and removing entities in the game. The entity manager handles the lifecycle of entities and ensures that they are correctly processed within the game's systems.

Returns

EntityManager& A reference to the entity manager instance.

6.8.3.9 GetPlayerClientId()

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

6.8.3.10 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.

6.8.3.11 InitiateEnemyMissile()

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

6.8.3.12 InitiatePlayer()

```
template<typename T >
EntityInformation r_type::net::AServer< T >::InitiatePlayer (
            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

clientId	The client ID of the player being initialized.

Returns

EntityInformation The information of the newly created player entity.

6.8.3.13 InitiatePlayerMissile()

```
template<typename T >
EntityInformation r_type::net::AServer< T >::InitiatePlayerMissile (
            int entityId )  [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

clientId	The client ID of the player firing the missile.

Returns

EntityInformation The information of the newly created missile entity.

6.8.3.14 InitiateWeaponForce()

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

6.8.3.15 InitInfoBar()

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

6.8.3.16 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.

6.8.3.17 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.

6.8.3.18 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

6.8.3.19 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	

6.8.3.20 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.

6.8.3.21 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	

6.8.3.22 RemoveEntity()

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

Removes entities associated with a player.

Parameters

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

6.8.3.23 RemoveInfoBar()

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

6.8.3.24 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.

6.8.3.25 SetClock()

```
template<typename T >
void r_type::net::AServer< T >::SetClock (
            std::chrono::system_clock::time_point clock )  [inline]
```

Set the Clock object.

Parameters

clock	

6.8.3.26 Start()

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

Start the server.

Returns

true

false

6.8.3.27 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.

6.8.3.28 Update()

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

Updates the game level based on the provided clock time.

This function performs several tasks to update the game level:

1. Checks if the time difference between the new clock and the stored clock exceeds 100 milliseconds.
2. If so, it updates entity positions, handles collisions, updates animations, and processes auto-firing.
3. Sends appropriate messages to clients about entity updates, creations, and destructions.

Parameters

newClock	The new clock time point to compare with the stored clock.
bUpdateEntities	A boolean reference that will be set to true if entities are updated.

Updates the server state, processes incoming messages, and updates the game level.

This function performs several tasks:

- If no players are connected, it returns immediately.
- If players are connected and the player connection flag is not set, it sets the flag and updates the clock.
- Spawns a thread to update the game level.
- Processes up to nMaxMessages from the incoming message queue.
- Joins the level update thread and updates the clock if entities were updated.

Parameters

nMaxMessages	The maximum number of messages to process from the incoming message queue. Default is -1 (process all messages).
bWait	A flag indicating whether to wait for messages. Default is false.

6.8.3.29 UpdateInfoBar()

```
template<typename T >
UIEntityInformation r_type::net::AServer< T >::UpdateInfoBar (
            int playerId )  [inline]
```

6.8.3.30 UpdatePlayerPosition()

```
template<typename T >
void r_type::net::AServer< T >::UpdatePlayerPosition (
            PlayerMovement direction,
            uint32_t entityId )  [inline], [override]
```

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.
clientId	The ID of the client sending the update.

6.8.3.31 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../

6.8.4 Member Data Documentation

6.8.4.1 _asioContext

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

6.8.4.2 _asioSocket

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

6.8.4.3 _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.

6.8.4.4 _clientEndpoint

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

6.8.4.5 _clientInfoBarID

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

6.8.4.6 _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 of type uint32_t representing the client IDs, and the values are also of type uint32_t representing the player IDs.

6.8.4.7 _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.

6.8.4.8 _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.

6.8.4.9 _deqConnections

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

6.8.4.10 _entityFactory

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

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

6.8.4.11 _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.

6.8.4.12 _level

```
template<typename T >
r_type::Level<T> r_type::net::AServer< T >::_level
```

6.8.4.13 _nbrOfPlayers

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

Number of players currently connected to the server.

6.8.4.14 _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.

6.8.4.15 _playerConnected

```
template<typename T >
bool r_type::net::AServer< T >::_playerConnected = false
```

6.8.4.16 _port

```
template<typename T >
int r_type::net::AServer< T >::_port
```

6.8.4.17 _qMessagesIn

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

6.8.4.18 _tempBuffer

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

6.8.4.19 _threadContext

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

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

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

6.9 AudioManager Class Reference

```
#include <audio_manager.hpp>
```

Public Member Functions

- sf::SoundBuffer & getSoundBuffer (const std::string &filePath)

Private Attributes

- std::unordered_map< std::string, std::shared_ptr< sf::SoundBuffer > > soundBuffers

6.9.1 Member Function Documentation

6.9.1.1 getSoundBuffer()

```
sf::SoundBuffer& AudioManager::getSoundBuffer (
            const std::string & filePath )  [inline]
```

6.9.2 Member Data Documentation

6.9.2.1 soundBuffers

```
std::unordered_map<std::string, std::shared_ptr<sf::SoundBuffer> > AudioManager::soundBuffers
[private]
```

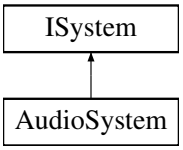
The documentation for this class was generated from the following file:

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

6.10 AudioSystem Class Reference

`#include <audio_system.hpp>`

Inheritance diagram for AudioSystem:



Public Member Functions

- AudioSystem (std::shared_ptr< AudioManager > audioManager)
- void playBackgroundMusic (const std::string &filePath)
- void stopBackgroundMusic ()
- void playSoundEffect (const std::string &filePath)

Private Attributes

- std::shared_ptr< AudioManager > _audioManager
- sf::Music _backgroundMusic
- std::string _currentMusicFilePath
- sf::Sound _soundEffect

6.10.1 Constructor & Destructor Documentation

6.10.1.1 AudioSystem()

```
AudioSystem::AudioSystem (
            std::shared_ptr< AudioManager > audioManager )  [inline]
```

6.10.2 Member Function Documentation

6.10.2.1 playBackgroundMusic()

```
void AudioSystem::playBackgroundMusic (
            const std::string & filePath )
```

6.10.2.2 playSoundEffect()

```
void AudioSystem::playSoundEffect (
            const std::string & filePath )
```

6.10.2.3 stopBackgroundMusic()

```
void AudioSystem::stopBackgroundMusic ( )
```

6.10.3 Member Data Documentation

6.10.3.1 _audioManager

```
std::shared_ptr<AudioManager> AudioSystem::_audioManager  [private]
```

6.10.3.2 _backgroundMusic

```
sf::Music AudioSystem::_backgroundMusic  [private]
```

6.10.3.3 _currentMusicFilePath

```
std::string AudioSystem::_currentMusicFilePath  [private]
```

6.10.3.4 _soundEffect

```
sf::Sound AudioSystem::_soundEffect  [private]
```

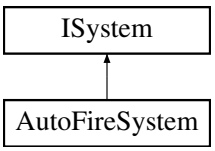
The documentation for this class was generated from the following files:

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

6.11 AutoFireSystem Class Reference

```
#include <auto_fire_system.hpp>
```

Inheritance diagram for AutoFireSystem:



Public Member Functions

- AutoFireSystem (ComponentManager &componentManager, EntityManager &entityManager)
- void handleAutoFire (ComponentManager &componentManager, EntityManager &entityManager)

Private Attributes

- ComponentManager & _componentManager
- EntityManager & _entityManager

6.11.1 Constructor & Destructor Documentation

6.11.1.1 AutoFireSystem()

```
AutoFireSystem::AutoFireSystem (
            ComponentManager & componentManager,
            EntityManager & entityManager )  [inline]
```

6.11.2 Member Function Documentation

6.11.2.1 handleAutoFire()

```
void AutoFireSystem::handleAutoFire (
            ComponentManager & componentManager,
            EntityManager & entityManager )
```

6.11.3 Member Data Documentation

6.11.3.1 _componentManager

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

6.11.3.2 _entityManager

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

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

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

6.12 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

6.13 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

6.14 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

6.14.1 Constructor & Destructor Documentation

6.14.1.1 BindComponent()

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

6.14.2 Member Data Documentation

6.14.2.1 bind

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

6.14.2.2 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

6.15 BossComponent Struct Reference

```
#include <boss_component.hpp>
```

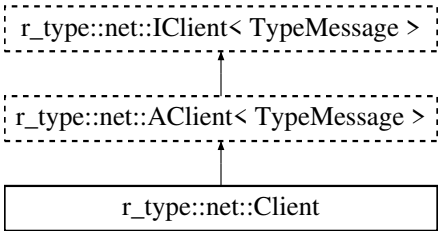
The documentation for this struct was generated from the following file:

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

6.16 r_type::net::Client Class Reference

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

Inheritance diagram for 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 initInfoBar (UIEntityInformation entity, ComponentManager &componentManager, TextureManager &textureManager, FontManager &fontManager, sf::Vector2u windowSize)
- void updateInfoBar (UIEntityInformation entity, ComponentManager &componentManager, TextureManager &textureManager)
- void addEntity (EntityInformation entity, ComponentManager &componentManager, TextureManager &textureManager, sf::Vector2u windowSize)
- void removeEntity (int entityId, ComponentManager &componentManager)
- void moveEntity (uint32_t id, vf2d newPos, ComponentManager &componentManager, sf::Vector2u window↩ Size)
- void animateEntity (int entityId, AnimationComponent rect, ComponentManager &componentManager)

Additional Inherited Members

6.16.1 Member Function Documentation

6.16.1.1 addEntity()

```
void r_type::net::Client::addEntity (
            EntityInformation entity,
            ComponentManager & componentManager,
            TextureManager & textureManager,
            sf::Vector2u windowSize )  [inline]
```

6.16.1.2 animateEntity()

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

6.16.1.3 initInfoBar()

```
void r_type::net::Client::initInfoBar (
            UIEntityInformation entity,
            ComponentManager & componentManager,
            TextureManager & textureManager,
            FontManager & fontManager,
            sf::Vector2u windowSize )  [inline]
```

6.16.1.4 MessageAll()

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

Send a message to the server to all other clients.

6.16.1.5 moveEntity()

```
void r_type::net::Client::moveEntity (
            uint32_t id,
            vf2d newPos,
            ComponentManager & componentManager,
            sf::Vector2u windowSize )  [inline]
```

6.16.1.6 PingServer()

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

Send a message to the server to get the ping.

6.16.1.7 removeEntity()

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

6.16.1.8 updateInfoBar()

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

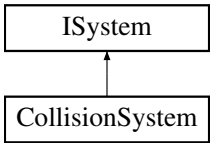
The documentation for this class was generated from the following file:

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

6.17 CollisionSystem Class Reference

```
#include <collision_system.hpp>
```

Inheritance diagram for CollisionSystem:



Public Member Functions

- CollisionSystem (ComponentManager &componentManager, EntityManager &entityManager)
- bool checkCollision (ComponentManager &componentManager, int entityId1, int entityId2)
- bool checkOffScreen (ComponentManager &componentManager, int entityId)

Private Attributes

- ComponentManager & _componentManager
- EntityManager & _entityManager

6.17.1 Constructor & Destructor Documentation

6.17.1.1 CollisionSystem()

```
CollisionSystem::CollisionSystem (
            ComponentManager & componentManager,
            EntityManager & entityManager )  [inline]
```

6.17.2 Member Function Documentation

6.17.2.1 checkCollision()

```
bool CollisionSystem::checkCollision (
            ComponentManager & componentManager,
            int entityId1,
            int entityId2 )
```

6.17.2.2 checkOffScreen()

```
bool CollisionSystem::checkOffScreen (
            ComponentManager & componentManager,
            int entityId )
```

6.17.3 Member Data Documentation

6.17.3.1 _componentManager

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

6.17.3.2 _entityManager

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

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

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

6.18 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)
- void removeEntityFromAllComponents (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.

6.18.1 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.

6.18.2 Member Function Documentation

6.18.2.1 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

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

6.18.2.2 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

entityId	The ID of the entity.

Returns

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

6.18.2.3 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.

6.18.2.4 removeEntityFromAllComponents()

```
void ComponentManager::removeEntityFromAllComponents (
            int entityId )  [inline]
```

6.18.2.5 removeEntityFromComponent()

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

6.18.3 Member Data Documentation

6.18.3.1 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:

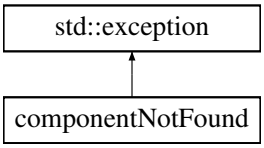
- /home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/component_manager.hpp

6.19 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

6.19.1 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.

6.19.2 Member Function Documentation

6.19.2.1 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

6.20 CreatableClientObject Class Reference

Enum class for the creatable client object.

`#include <creatable_client_object.hpp>`

6.20.1 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

6.21 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

6.22 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

6.23 Entity Class Reference

Represents an entity in the ECS system.

`#include <entity.hpp>`

Public Member Functions

- Entity (int id)

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

  Returns the ID of the entity.

Private Attributes

- int _id

6.23.1 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.

6.23.2 Constructor & Destructor Documentation

6.23.2.1 Entity()

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

Constructs an Entity object with the given ID.

Parameters

id	The ID of the entity.

6.23.3 Member Function Documentation

6.23.3.1 getId()

```
int Entity::getId ( ) const  [inline]
```

Returns the ID of the entity.

Returns

The ID of the entity.

6.23.4 Member Data Documentation

6.23.4.1 _id

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

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

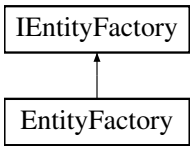
- /home/runner/work/R-Type/R-Type/ECS/Interface/Include/Entities/entity.hpp

6.24 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 createInfoBar (EntityManager &entityManager, ComponentManager &componentManager) override

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

  Creates a player entity.
- Entity createShooterEnemy (EntityManager &entityManager, ComponentManager &componentManager, int posX, int posY) override

  Creates a shooter enemy entity.
- Entity createBasicMonster (EntityManager &entityManager, ComponentManager &componentManager, int posX, int posY) override

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

  Creates a player missile entity.
- Entity createForceWeapon (EntityManager &entityManager, ComponentManager &componentManager, uint32_t entityId) override
- Entity createPowerUpBlueLaserCrystal (EntityManager &entityManager, ComponentManager &component↩ Manager) override
- Entity createButton (EntityManager &entityManager, ComponentManager &componentManager, TextureManager &textureManager, FontManager &fontManager, std::string text, std::function< IScenes *(AScenes *)> *on↩ Click, float x=0, float y=0) override

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

  Creates a small button entity.
- Entity createEnemyMissile (EntityManager &entityManager, ComponentManager &componentManager, uint32_t entityId) override

  Creates an ally missile entity.
- Entity createFilter (EntityManager &entityManager, ComponentManager &componentManager, AScenes::DaltonismMode mode)

  Create a Filter object.

Additional Inherited Members

6.24.1 Detailed Description

A class responsible for creating different types of entities.

6.24.2 Member Function Documentation

6.24.2.1 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.

6.24.2.2 createBasicMonster()

```
Entity EntityFactory::createBasicMonster (
            EntityManager & entityManager,
            ComponentManager & componentManager,
            int posX,
            int posY )  [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.

6.24.2.3 createButton()

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

6.24.2.4 createEnemyMissile()

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

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.

6.24.2.5 createFilter()

```
Entity EntityFactory::createFilter (
            EntityManager & entityManager,
            ComponentManager & componentManager,
            AScenes::DaltonismMode mode )
```

Create a Filter object.

Parameters

entityManager	
componentManager	
mode	

Returns

Entity

6.24.2.6 createForceWeapon()

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

Implements IEntityFactory.

6.24.2.7 createInfoBar()

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

Creates a bar entity.

This function creates a bar with text for displaying player information like health and score.

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 bar entity.

Implements IEntityFactory.

6.24.2.8 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.

6.24.2.9 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.

6.24.2.10 createPowerUpBlueLaserCrystal()

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

Implements IEntityFactory.

6.24.2.11 createShooterEnemy()

```
Entity EntityFactory::createShooterEnemy (
            EntityManager & entityManager,
            ComponentManager & componentManager,
            int posX,
            int posY )  [override], [virtual]
```

Creates a shooter enemy entity.

This function creates a shooter 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.

6.24.2.12 createSmallButton()

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

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.

Implements IEntityFactory.

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

6.25 EntityInformation Struct Reference

Represents information about an entity.

`#include <entity_struct.hpp>`

Public Attributes

- uint32_t uniqueID = 0
- vf2d ratio = {0, 0}
- SpriteDataComponent spriteData
- vf2d vPos = {0, 0}
- AnimationComponent animationComponent = {{0, 0}, {0, 0}}

6.25.1 Detailed Description

Represents information about an entity.

6.25.2 Member Data Documentation

6.25.2.1 animationComponent

```
AnimationComponent EntityInformation::animationComponent = {{0, 0}, {0, 0}}
```

6.25.2.2 ratio

```
vf2d EntityInformation::ratio = {0, 0}
```

6.25.2.3 spriteData

```
SpriteDataComponent EntityInformation::spriteData
```

6.25.2.4 uniqueID

```
uint32_t EntityInformation::uniqueID = 0
```

6.25.2.5 vPos

```
vf2d EntityInformation::vPos = {0, 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

6.26 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.
- std::optional< 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

6.26.1 Detailed Description

Class responsible for managing entities in the ECS system.

6.26.2 Member Function Documentation

6.26.2.1 createEntity()

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

Create a Entity object.

Returns

Entity

6.26.2.2 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.

6.26.2.3 getEntity()

```
std::optional<Entity *> EntityManager::getEntity (
            int entityId )  [inline]
```

Get an entity by its ID.

Parameters

entityId	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.

6.26.2.4 removeEntity()

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

Remove an entity from the entity manager.

Parameters

entityId	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.

6.26.3 Member Data Documentation

6.26.3.1 entities

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

6.26.3.2 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:

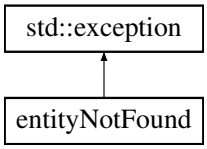
- /home/runner/work/R-Type/R-Type/ECS/Interface/Include/Entities/entity_manager.hpp

6.27 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

6.27.1 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.

6.27.2 Member Function Documentation

6.27.2.1 what()

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

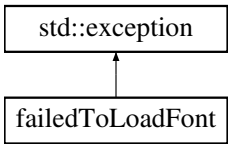
The documentation for this class was generated from the following file:

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

6.28 failedToLoadFont Class Reference

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

Inheritance diagram for failedToLoadFont:



Private Member Functions

- const char * what () const noexcept override

6.28.1 Member Function Documentation

6.28.1.1 what()

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

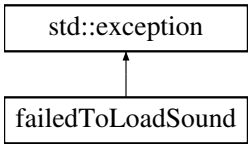
The documentation for this class was generated from the following file:

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

6.29 failedToLoadSound Class Reference

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

Inheritance diagram for failedToLoadSound:



Private Member Functions

- const char ∗ what () const noexcept override

6.29.1 Member Function Documentation

6.29.1.1 what()

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

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

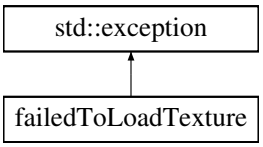
- /home/runner/work/R-Type/R-Type/ECS/Interface/Include/error_handling.hpp

6.30 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

6.30.1 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.

6.30.2 Member Function Documentation

6.30.2.1 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

6.31 FontManager Class Reference

```
#include <font_manager.hpp>
```

Public Member Functions

- sf::Font & getFont (const std::string &filePath)
- void releaseFont (const std::string &filePath)

Private Attributes

- std::unordered_map< std::string, sf::Font > fonts

6.31.1 Member Function Documentation

6.31.1.1 getFont()

```
sf::Font& FontManager::getFont (
            const std::string & filePath )  [inline]
```

6.31.1.2 releaseFont()

```
void FontManager::releaseFont (
            const std::string & filePath )  [inline]
```

6.31.2 Member Data Documentation

6.31.2.1 fonts

```
std::unordered_map<std::string, sf::Font> FontManager::fonts  [private]
```

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

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

6.32 HealthComponent Struct Reference

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

Public Attributes

- int max_health
- int health

6.32.1 Member Data Documentation

6.32.1.1 health

```
int HealthComponent::health
```

6.32.1.2 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

6.33 HitboxComponent Struct Reference

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

Public Attributes

- int w
- int h

6.33.1 Member Data Documentation

6.33.1.1 h

```
int HitboxComponent::h
```

6.33.1.2 w

```
int HitboxComponent::w
```

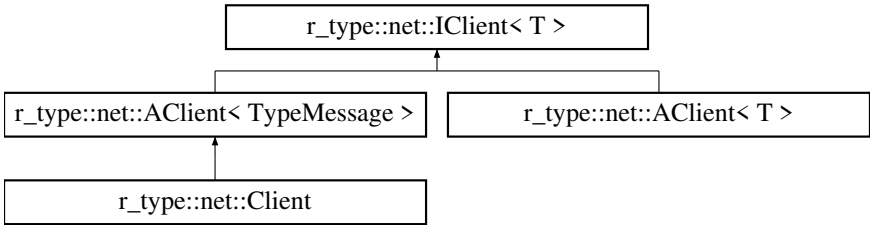
The documentation for this struct was generated from the following file:

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

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

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

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



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

6.34.1 Constructor & Destructor Documentation

6.34.1.1 IClient()

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

6.34.1.2 ~IClient()

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

6.34.2 Member Function Documentation

6.34.2.1 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 >.

6.34.2.2 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 >.

6.34.2.3 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 >.

6.34.2.4 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 >.

6.34.2.5 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:

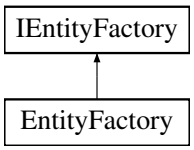
- /home/runner/work/R-Type/R-Type/Client/Interface/Include/Net/i_client.hpp

6.35 IEntityFactory Class Reference

The interface for an entity factory.

`#include <i_entity_factory.hpp>`

Inheritance diagram for IEntityFactory:



Public Types

- enum EnemyType { BasicMonster , ShooterEnemy , Boss }

Public Member Functions

- virtual ~IEntityFactory ()=default

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

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

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

  Creates a player entity.
- virtual Entity createShooterEnemy (EntityManager &entityManager, ComponentManager &componentManager, int posX, int posY)=0

  Creates a shooter enemy entity.
- virtual Entity createBasicMonster (EntityManager &entityManager, ComponentManager &componentManager, int posX, int posY)=0

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

Creates a player missile entity.

- virtual Entity createForceWeapon (EntityManager &entityManager, ComponentManager &componentManager, uint32_t entityId)=0
- virtual Entity createPowerUpBlueLaserCrystal (EntityManager &entityManager, ComponentManager &componentManager)=0
- virtual Entity createEnemyMissile (EntityManager &entityManager, ComponentManager &componentManager, uint32_t entityId)=0

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

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

6.35.1 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.

6.35.2 Member Enumeration Documentation

6.35.2.1 EnemyType

```
enum IEntityFactory::EnemyType
```

Enumerator

BasicMonster	
ShooterEnemy	
Boss	

6.35.3 Constructor & Destructor Documentation

6.35.3.1 ~IEntityFactory()

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

Destroy the IEntityFactory object.

6.35.4 Member Function Documentation

6.35.4.1 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.

6.35.4.2 createBasicMonster()

```
virtual Entity IEntityFactory::createBasicMonster (
            EntityManager & entityManager,
            ComponentManager & componentManager,
            int posX,
            int posY )  [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.

6.35.4.3 createButton()

```
virtual Entity IEntityFactory::createButton (
            EntityManager & entityManager,
            ComponentManager & componentManager,
            TextureManager & textureManager,
            FontManager & fontManager,
            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.

6.35.4.4 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.

6.35.4.5 createForceWeapon()

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

Implemented in EntityFactory.

6.35.4.6 createInfoBar()

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

Creates a bar entity.

This function creates a bar with text for displaying player information like health and score.

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 bar entity.

Implemented in EntityFactory.

6.35.4.7 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.

6.35.4.8 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.

6.35.4.9 createPowerUpBlueLaserCrystal()

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

Implemented in EntityFactory.

6.35.4.10 createShooterEnemy()

```
virtual Entity IEntityFactory::createShooterEnemy (
            EntityManager & entityManager,
            ComponentManager & componentManager,
            int posX,
            int posY )  [pure virtual]
```

Creates a shooter enemy entity.

This function creates a shooter 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 shooter enemy entity.

Implemented in EntityFactory.

6.35.4.11 createSmallButton()

```
virtual Entity IEntityFactory::createSmallButton (
            EntityManager & entityManager,
            ComponentManager & componentManager,
            TextureManager & textureManager,
            FontManager & fontManager,
            std::string text,
            std::function< IScenes *(AScenes *, AScenes::Actions)> * onClick,
            float x = 0,
            float y = 0 )  [pure virtual]
```

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

6.36 InputComponent Struct Reference

`#include <input_component.hpp>`

Public Attributes

- InputType input

6.36.1 Member Data Documentation

6.36.1.1 input

`InputType InputComponent::input`

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

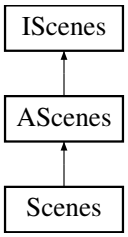
- /home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/input_component.hpp

6.37 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 difficultyChoices ()=0

  Displays the difficulty choices.
- 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.

6.37.1 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.

6.37.2 Constructor & Destructor Documentation

6.37.2.1 ~IScenes()

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

6.37.3 Member Function Documentation

6.37.3.1 difficultyChoices()

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

Displays the difficulty choices.

Implemented in Scenes.

6.37.3.2 gameLoop()

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

Displays the main game loop and creates necessary entities.

Implemented in Scenes.

6.37.3.3 getRenderWindow()

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

Gets the render window.

Returns

Pointer to the sf::RenderWindow.

Implemented in Scenes.

6.37.3.4 inGameMenu()

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

Displays the in-game menu and creates necessary entities.

Implemented in Scenes.

6.37.3.5 mainMenu()

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

Displays the main menu and creates necessary entities.

Implemented in Scenes.

6.37.3.6 render()

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

Displays the current scene and manages its components.

Implemented in Scenes.

6.37.3.7 settingsMenu()

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

Displays the settings menu and creates necessary entities.

Implemented in Scenes.

6.37.3.8 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.

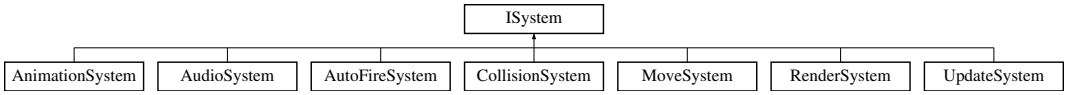
The documentation for this class was generated from the following file:

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

6.38 ISystem Class Reference

`#include <i_system.hpp>`

Inheritance diagram for ISystem:



Public Member Functions

- ISystem ()=default
- virtual ~ISystem ()=default

6.38.1 Constructor & Destructor Documentation

6.38.1.1 ISystem()

`ISystem::ISystem ( )  [default]`

6.38.1.2 ~ISystem()

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

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

6.39 labelComponent Struct Reference

`#include <label_component.hpp>`

Public Attributes

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

6.39.1 Member Data Documentation

6.39.1.1 name

```
std::string labelComponent::name
```

6.39.1.2 x

```
int labelComponent::x
```

6.39.1.3 y

```
int labelComponent::y
```

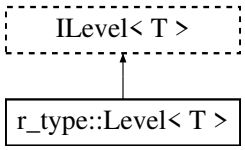
The documentation for this struct was generated from the following file:

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

6.40 r_type::Level< T > Class Template Reference

```
#include <level.hpp>
```

Inheritance diagram for r_type::Level< T >:



Public Member Functions

- Level ()=default
- ~Level ()=default
- void Update (r_type::net::AServer< T > *server, ComponentManager &componentManager, EntityManager &entityManager, std::chrono::system_clock::time_point newClock, bool *bUpdateEntities) override

  Updates the game state by processing entity movements, handling collisions, and sending messages to clients.
- void SetSystem (ComponentManager &componentManager, EntityManager &entityManager) override

  Initializes and sets up various systems for the level.
- void MoveUpdate (r_type::net::AServer< T > *server, ComponentManager &componentManager, EntityManager &entityManager, std::chrono::system_clock::time_point newClock) override

  Updates the positions of entities and notifies clients of any changes.
- void CollisionUpdate (r_type::net::AServer< T > *server, ComponentManager &componentManager, EntityManager &entityManager, std::chrono::system_clock::time_point newClock) override
- void AnimationUpdate (r_type::net::AServer< T > *server, ComponentManager &componentManager, EntityManager &entityManager, std::chrono::system_clock::time_point newClock) override

  Updates the animations of entities and sends messages to clients if animations have changed.
- void FireUpdate (r_type::net::AServer< T > *server, ComponentManager &componentManager, EntityManager &entityManager, std::chrono::system_clock::time_point newClock) override

  Updates the firing mechanism of entities in the game.
- void LevelOne (r_type::net::AServer< T > *server, ComponentManager &componentManager, EntityManager &entityManager, std::chrono::system_clock::time_point newClock) override
- void SpawnEntity (r_type::net::AServer< T > *server, EntityManager &entityManager, ComponentManager &componentManager, int nbrOfEnemy, EntityFactory::EnemyType enemyType)

Protected Attributes

- std::shared_ptr< MoveSystem > _moveSystem
- std::shared_ptr< CollisionSystem > _collisionSystem
- std::shared_ptr< AnimationSystem > _animationSystem
- std::shared_ptr< AutoFireSystem > _autoFireSystem
- r_type::TypeLevel _levelType
- std::chrono::system_clock::time_point _basicMonsterSpawnTime
- std::chrono::system_clock::time_point _shooterEnemySpawnTime
- std::chrono::system_clock::time_point _spawnTimeMonsterThree

6.40.1 Constructor & Destructor Documentation

6.40.1.1 Level()

```
template<typename T >
r_type::Level< T >::Level ( )  [default]
```

6.40.1.2 ~Level()

```
template<typename T >
r_type::Level< T >::~Level ( )  [default]
```

6.40.2 Member Function Documentation

6.40.2.1 AnimationUpdate()

```
template<typename T >
void r_type::Level< T >::AnimationUpdate (
            r_type::net::AServer< T > * server,
            ComponentManager & componentManager,
            EntityManager & entityManager,
            std::chrono::system_clock::time_point newClock )  [inline], [override]
```

Updates the animations of entities and sends messages to clients if animations have changed.

This function performs the following steps:

1. Retrieves the current animation components from the component manager.
2. Saves the current state of animations.
3. Updates the animations using the animation system.
4. Compares the new state of animations with the previous state.
5. Sends messages to all clients if any animations have changed.

Parameters

server	Pointer to the server instance.
componentManager	Reference to the component manager.
entityManager	Reference to the entity manager.
newClock	The current time point.

6.40.2.2 CollisionUpdate()

```
template<typename T >
void r_type::Level< T >::CollisionUpdate (
            r_type::net::AServer< T > * server,
            ComponentManager & componentManager,
            EntityManager & entityManager,
            std::chrono::system_clock::time_point newClock )  [inline], [override]
```

6.40.2.3 FireUpdate()

```
template<typename T >
void r_type::Level< T >::FireUpdate (
```

```
            r_type::net::AServer< T > * server,
            ComponentManager & componentManager,
            EntityManager & entityManager,
            std::chrono::system_clock::time_point newClock )  [inline], [override]
```

Updates the firing mechanism of entities in the game.

This function handles the automatic firing system and processes the firing logic for entities. It retrieves all entities and checks if they can shoot. If an entity can shoot, it sends a message to all clients to create an enemy missile and sets the entity's canShoot flag to false.

Parameters

server	Pointer to the server instance.
componentManager	Reference to the ComponentManager handling components.
entityManager	Reference to the EntityManager handling entities.
newClock	The current time point used for timing events.

6.40.2.4 LevelOne()

```
template<typename T >
void r_type::Level< T >::LevelOne (
            r_type::net::AServer< T > * server,
            ComponentManager & componentManager,
            EntityManager & entityManager,
            std::chrono::system_clock::time_point newClock )  [inline], [override]
```

6.40.2.5 MoveUpdate()

```
template<typename T >
void r_type::Level< T >::MoveUpdate (
            r_type::net::AServer< T > * server,
            ComponentManager & componentManager,
            EntityManager & entityManager,
            std::chrono::system_clock::time_point newClock )  [inline], [override]
```

Updates the positions of entities and notifies clients of any changes.

This function performs the following steps:

1. Retrieves the current positions of entities and stores them.
2. Moves the entities using the move system.
3. Compares the new positions with the previous positions.
4. If an entity's position has changed, sends an update message to all clients.

Parameters

server	Pointer to the server instance.
componentManager	Reference to the ComponentManager.
entityManager	Reference to the EntityManager.
newClock	The current time point.

6.40.2.6 SetSystem()

```
template<typename T >
void r_type::Level< T >::SetSystem (
            ComponentManager & componentManager,
            EntityManager & entityManager )  [inline], [override]
```

Initializes and sets up various systems for the level.

This function overrides a base class method to initialize and set up the MoveSystem, CollisionSystem, AnimationSystem, and AutoFireSystem using the provided ComponentManager and EntityManager.

Parameters

componentManager	Reference to the ComponentManager used to manage components.
entityManager	Reference to the EntityManager used to manage entities.

6.40.2.7 SpawnEntity()

```
template<typename T >
void r_type::Level< T >::SpawnEntity (
            r_type::net::AServer< T > * server,
            EntityManager & entityManager,
            ComponentManager & componentManager,
            int nbrOfEnemy,
            EntityFactory::EnemyType enemyType )  [inline]
```

6.40.2.8 Update()

```
template<typename T >
void r_type::Level< T >::Update (
            r_type::net::AServer< T > * server,
            ComponentManager & componentManager,
            EntityManager & entityManager,
            std::chrono::system_clock::time_point newClock,
            bool * bUpdateEntities )  [inline], [override]
```

Updates the game state by processing entity movements, handling collisions, and sending messages to clients.

This function performs several tasks to update the game state:

- Moves entities based on the elapsed time.
- Handles collisions between entities.
- Sends messages to clients about destroyed entities.
- Updates animations and firing mechanisms.

Parameters

server	Pointer to the server instance.
componentManager	Reference to the ComponentManager handling game components.
entityManager	Reference to the EntityManager handling game entities.
newClock	The current time point used to calculate elapsed time.
bUpdateEntities	Pointer to a boolean flag indicating whether entities should be updated.

6.40.3 Member Data Documentation

6.40.3.1 _animationSystem

```
template<typename T >
std::shared_ptr<AnimationSystem> r_type::Level< T >::_animationSystem  [protected]
```

6.40.3.2 _autoFireSystem

```
template<typename T >
std::shared_ptr<AutoFireSystem> r_type::Level< T >::_autoFireSystem  [protected]
```

6.40.3.3 _basicMonsterSpawnTime

```
template<typename T >
std::chrono::system_clock::time_point r_type::Level< T >::_basicMonsterSpawnTime  [protected]
```

Initial value:

```
=
        std::chrono::system_clock::now()
```

6.40.3.4 _collisionSystem

```
template<typename T >
std::shared_ptr<CollisionSystem> r_type::Level< T >::_collisionSystem  [protected]
```

6.40.3.5 _levelType

```
template<typename T >
r_type::TypeLevel r_type::Level< T >::_levelType  [protected]
```

6.40.3.6 _moveSystem

```
template<typename T >
std::shared_ptr<MoveSystem> r_type::Level< T >::_moveSystem  [protected]
```

6.40.3.7 _shooterEnemySpawnTime

```
template<typename T >
std::chrono::system_clock::time_point r_type::Level< T >::_shooterEnemySpawnTime  [protected]
```

Initial value:

```
=
        std::chrono::system_clock::now()
```

6.40.3.8 _spawnTimeMonsterThree

```
template<typename T >
std::chrono::system_clock::time_point r_type::Level< T >::_spawnTimeMonsterThree  [protected]
```

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

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

6.41 MovementComponent Struct Reference

```
#include <movement_component.hpp>
```

Public Attributes

- MovementType movementType
- uint32_t index

6.41.1 Member Data Documentation

6.41.1.1 index

```
uint32_t MovementComponent::index
```

6.41.1.2 movementType

```
MovementType MovementComponent::movementType
```

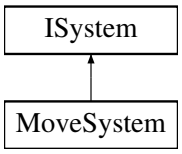
The documentation for this struct was generated from the following file:

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

6.42 MoveSystem Class Reference

```
#include <move_system.hpp>
```

Inheritance diagram for MoveSystem:



Public Member Functions

- MoveSystem (ComponentManager &componentManager, EntityManager &entityManager)
- void moveEntities (ComponentManager &componentManager, EntityManager &entityManager)

Private Attributes

- ComponentManager & _componentManager
- EntityManager & _entityManager

6.42.1 Constructor & Destructor Documentation

6.42.1.1 MoveSystem()

```
MoveSystem::MoveSystem (
            ComponentManager & componentManager,
            EntityManager & entityManager )  [inline]
```

6.42.2 Member Function Documentation

6.42.2.1 moveEntities()

```
void MoveSystem::moveEntities (
            ComponentManager & componentManager,
            EntityManager & entityManager )
```

6.42.3 Member Data Documentation

6.42.3.1 _componentManager

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

6.42.3.2 _entityManager

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

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

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

6.43 OffsetComponent Struct Reference

```
#include <offset_component.hpp>
```

Public Attributes

- float offset

6.43.1 Member Data Documentation

6.43.1.1 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

6.44 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

6.44.1 Constructor & Destructor Documentation

6.44.1.1 OnClickComponent()

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

6.44.2 Member Data Documentation

6.44.2.1 isClicked

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

6.44.2.2 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

6.45 PlayerComponent Struct Reference

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

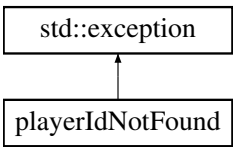
The documentation for this struct was generated from the following file:

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

6.46 playerIdNotFound Class Reference

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

Inheritance diagram for playerIdNotFound:



Private Member Functions

- const char * what () const noexcept override

6.46.1 Member Function Documentation

6.46.1.1 what()

```
const char* playerIdNotFound::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

6.47 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

6.48 PositionComponent Struct Reference

`#include <position_component.hpp>`

Public Member Functions

- PositionComponent (float _x, float _y)

Public Attributes

- float x
- float y

6.48.1 Constructor & Destructor Documentation

6.48.1.1 PositionComponent()

```
PositionComponent::PositionComponent (
            float _x,
            float _y )  [inline]
```

6.48.2 Member Data Documentation

6.48.2.1 x

```
float PositionComponent::x
```

6.48.2.2 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

6.49 PowerUpComponent Struct Reference

```
#include <power_up_component.hpp>
```

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

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

6.50 RectangleShapeComponent Struct Reference

```
#include <rectangleShapeComponent.hpp>
```

Public Member Functions

- RectangleShapeComponent (sf::RectangleShape &rectangleShape)

Public Attributes

- sf::RectangleShape rectangleShape

6.50.1 Constructor & Destructor Documentation

6.50.1.1 RectangleShapeComponent()

```
RectangleShapeComponent::RectangleShapeComponent (
            sf::RectangleShape & rectangleShape )  [inline]
```

6.50.2 Member Data Documentation

6.50.2.1 rectangleShape

```
sf::RectangleShape RectangleShapeComponent::rectangleShape
```

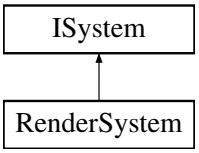
The documentation for this struct was generated from the following file:

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

6.51 RenderSystem Class Reference

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

Inheritance diagram for RenderSystem:



Public Member Functions

- RenderSystem (sf::RenderWindow &window, ComponentManager &componentManager)
- void render (ComponentManager &componentManager)

Private Attributes

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

6.51.1 Constructor & Destructor Documentation

6.51.1.1 RenderSystem()

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

6.51.2 Member Function Documentation

6.51.2.1 render()

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

6.51.3 Member Data Documentation

6.51.3.1 _componentManager

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

6.51.3.2 _font

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

6.51.3.3 _window

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

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

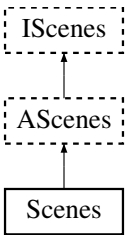
- /home/runner/work/R-Type/R-Type/ECS/Interface/Include/Systems/render_system.hpp
- /home/runner/work/R-Type/R-Type/ECS/Src/Systems/render_system.cpp

6.52 Scenes Class Reference

Represents a class that manages different scenes in a game.

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

Inheritance diagram for Scenes:



Public Member Functions

- Scenes (std::string ip, int port)

  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 HandleMessage (r_type::net::Message< TypeMessage > &msg, ComponentManager &component↩ Manager, TextureManager &textureManager, FontManager &fontManager, const sf::Vector2u &windowSize, std::shared_ptr< AudioSystem > &audioSystem)
- void StopGameLoop (std::shared_ptr< AudioSystem > &audioSystem)
- void settingsMenu ()

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

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

  displays the difficulty choices, 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.
- void run ()

Public Attributes

- sf::RenderWindow _window
- r_type::net::Client _networkClient

Additional Inherited Members

6.52.1 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.

6.52.2 Constructor & Destructor Documentation

6.52.2.1 Scenes()

```
Scenes::Scenes (
            std::string ip,
            int port )
```

Construct a new Scenes object.

Parameters

window	

6.52.2.2 ~Scenes()

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

Destroy the Scenes object.

6.52.3 Member Function Documentation

6.52.3.1 difficultyChoices()

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

displays the difficulty choices, creates all the necessary entities

Implements IScenes.

6.52.3.2 gameLoop()

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

displays the main game loop, creates all the necessary entities

Implements IScenes.

6.52.3.3 getRenderWindow()

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

Get the RenderWindow object.

Returns

sf::RenderWindow*

Implements IScenes.

6.52.3.4 HandleMessage()

```
void Scenes::HandleMessage (
            r_type::net::Message< TypeMessage > & msg,
            ComponentManager & componentManager,
            TextureManager & textureManager,
            FontManager & fontManager,
            const sf::Vector2u & windowSize,
            std::shared_ptr< AudioSystem > & audioSystem )
```

6.52.3.5 inGameMenu()

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

displays the in game menu, 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

Displays the in-game menu.

Implements IScenes.

6.52.3.6 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.

6.52.3.7 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.

6.52.3.8 run()

```
void Scenes::run ( )
```

6.52.3.9 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.

6.52.3.10 shouldQuit()

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

check if game should stop running

Returns

true
false

Implements IScenes.

6.52.3.11 StopGameLoop()

```
void Scenes::StopGameLoop (
            std::shared_ptr< AudioSystem > & audioSystem )
```

6.52.4 Member Data Documentation

6.52.4.1 _networkClient

```
r_type::net::Client Scenes::_networkClient
```

6.52.4.2 _window

```
sf::RenderWindow Scenes::_window
```

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

6.53 ScoreComponent Struct Reference

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

Public Attributes

- int score

6.53.1 Member Data Documentation

6.53.1.1 score

```
int ScoreComponent::score
```

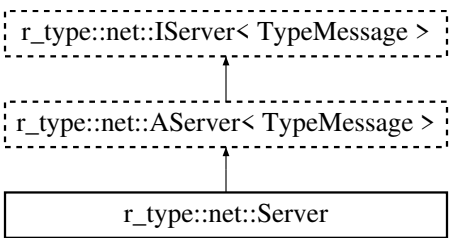
The documentation for this struct was generated from the following file:

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

6.54 r_type::net::Server Class Reference

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

Inheritance diagram for 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.

Additional Inherited Members

6.54.1 Constructor & Destructor Documentation

6.54.1.1 Server()

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

6.54.1.2 ~Server()

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

6.54.2 Member Function Documentation

6.54.2.1 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

6.54.2.2 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	

6.54.2.3 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

6.55 ShaderComponent Struct Reference

```
#include <shader_component.hpp>
```

Public Member Functions

- ShaderComponent (std::string path)

Public Attributes

- std::shared_ptr< sf::Shader > shader

6.55.1 Constructor & Destructor Documentation

6.55.1.1 ShaderComponent()

```
ShaderComponent::ShaderComponent (
            std::string path )  [inline]
```

6.55.2 Member Data Documentation

6.55.2.1 shader

```
std::shared_ptr<sf::Shader> ShaderComponent::shader
```

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

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

6.56 ShootComponent Struct Reference

```
#include <shoot_component.hpp>
```

Public Member Functions

- ShootComponent (std::chrono::milliseconds cooldown)

Public Attributes

- std::chrono::system_clock::time_point nextShootTime
- std::chrono::milliseconds cooldownTime
- bool canShoot

6.56.1 Constructor & Destructor Documentation

6.56.1.1 ShootComponent()

```
ShootComponent::ShootComponent (
            std::chrono::milliseconds cooldown )  [inline]
```

6.56.2 Member Data Documentation

6.56.2.1 canShoot

```
bool ShootComponent::canShoot
```

6.56.2.2 cooldownTime

```
std::chrono::milliseconds ShootComponent::cooldownTime
```

6.56.2.3 nextShootTime

```
std::chrono::system_clock::time_point ShootComponent::nextShootTime
```

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

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

6.57 SpriteComponent Struct Reference

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

Public Member Functions

- SpriteComponent (sf::Texture &texture, const float posX, float posY, const sf::Vector2f &scale, AScenes::SpriteType typeNb, sf::IntRect rect=sf::IntRect(0, 0, 0, 0))

Public Attributes

- sf::Sprite sprite
- AScenes::SpriteType type
- int hitboxX
- int hitboxY

6.57.1 Constructor & Destructor Documentation

6.57.1.1 SpriteComponent()

```
SpriteComponent::SpriteComponent (
            sf::Texture & texture,
            const float posX,
            float posY,
            const sf::Vector2f & scale,
            AScenes::SpriteType typeNb,
            sf::IntRect rect = sf::IntRect(0, 0, 0, 0) )  [inline]
```

6.57.2 Member Data Documentation

6.57.2.1 hitboxX

```
int SpriteComponent::hitboxX
```

6.57.2.2 hitboxY

```
int SpriteComponent::hitboxY
```

6.57.2.3 sprite

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

6.57.2.4 type

```
AScenes::SpriteType SpriteComponent::type
```

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

6.58 SpriteDataComponent Struct Reference

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

Public Attributes

- SpritePath spritePath
- vf2d scale
- AScenes::SpriteType type

6.58.1 Member Data Documentation

6.58.1.1 scale

```
vf2d SpriteDataComponent::scale
```

6.58.1.2 spritePath

```
SpritePath SpriteDataComponent::spritePath
```

6.58.1.3 type

```
AScenes::SpriteType SpriteDataComponent::type
```

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

6.59 TextComponent Struct Reference

`#include <text_component.hpp>`

Public Member Functions

- TextComponent (sf::Font &font, const std::string &string, float posX, float posY)

Public Attributes

- sf::Text text

6.59.1 Constructor & Destructor Documentation

6.59.1.1 TextComponent()

```
TextComponent::TextComponent (
            sf::Font & font,
            const std::string & string,
            float posX,
            float posY )  [inline]
```

6.59.2 Member Data Documentation

6.59.2.1 text

```
sf::Text 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

6.60 TextDataComponent Struct Reference

`#include <text_data_component.hpp>`

Public Attributes

- FontPath fontPath
- uint32_t categoryIds [5] = {0}
- GameText categoryTexts [5]
- uint32_t categorySize = 0

6.60.1 Member Data Documentation

6.60.1.1 categoryIds

```
uint32_t TextDataComponent::categoryIds[5] = {0}
```

6.60.1.2 categorySize

```
uint32_t TextDataComponent::categorySize = 0
```

6.60.1.3 categoryTexts

```
GameText TextDataComponent::categoryTexts[5]
```

6.60.1.4 fontPath

```
FontPath TextDataComponent::fontPath
```

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

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

6.61 TextureManager Class Reference

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

Public Member Functions

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

  Retrieves a texture from the texture manager.
- void releaseTexture (const std::string &filePath)

Private Attributes

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

  A container for storing textures with string keys.

6.61.1 Member Function Documentation

6.61.1.1 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.

6.61.1.2 releaseTexture()

```
void TextureManager::releaseTexture (
            const std::string & filePath )  [inline]
```

6.61.2 Member Data Documentation

6.61.2.1 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

6.62 UIEntityInformation Struct Reference

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

Public Attributes

- uint32_t uniqueID = 0
- uint32_t lives = 0
- uint32_t score = 0
- SpriteDataComponent spriteData
- TextDataComponent textData

6.62.1 Member Data Documentation

6.62.1.1 lives

```
uint32_t UIEntityInformation::lives = 0
```

6.62.1.2 score

```
uint32_t UIEntityInformation::score = 0
```

6.62.1.3 spriteData

```
SpriteDataComponent UIEntityInformation::spriteData
```

6.62.1.4 textData

```
TextDataComponent UIEntityInformation::textData
```

6.62.1.5 uniqueID

```
uint32_t UIEntityInformation::uniqueID = 0
```

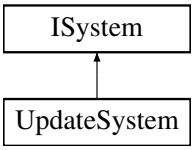
The documentation for this struct was generated from the following file:

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

6.63 UpdateSystem Class Reference

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

Inheritance diagram for UpdateSystem:



Public Member Functions

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

Private Attributes

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

6.63.1 Constructor & Destructor Documentation

6.63.1.1 UpdateSystem()

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

6.63.2 Member Function Documentation

6.63.2.1 updateSpritePositions()

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

6.63.3 Member Data Documentation

6.63.3.1 _componentManager

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

6.63.3.2 _entityManager

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

6.63.3.3 _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

6.64 VelocityComponent Struct Reference

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

Public Attributes

- float x
- float y

6.64.1 Member Data Documentation

6.64.1.1 x

```
float VelocityComponent::x
```

6.64.1.2 y

```
float VelocityComponent::y
```

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

6.65 vf2d Struct Reference

Represents a 2D vector with x and y coordinates.

```
#include <macros.hpp>
```

Public Attributes

- float x = 0
- float y = 0

6.65.1 Detailed Description

Represents a 2D vector with x and y coordinates.

6.65.2 Member Data Documentation

6.65.2.1 x

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

6.65.2.2 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/macros.hpp

6.66 WeaponComponent Struct Reference

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

Public Member Functions

- WeaponComponent (float _damage, float _fire_rate, float _bullet_speed)

Public Attributes

- float damage
- float fire_rate
- float bullet_speed

6.66.1 Constructor & Destructor Documentation

6.66.1.1 WeaponComponent()

```
WeaponComponent::WeaponComponent (
            float _damage,
            float _fire_rate,
            float _bullet_speed )  [inline]
```

6.66.2 Member Data Documentation

6.66.2.1 bullet_speed

```
float WeaponComponent::bullet_speed
```

6.66.2.2 damage

```
float WeaponComponent::damage
```

6.66.2.3 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 <font_manager.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/scenes.hpp File Reference

```
#include <Entities/entity.hpp>
#include <Net/client.hpp>
#include <SFML/Graphics.hpp>
#include <Systems/systems.hpp>
#include <a_scenes.hpp>
#include <memory>
#include <vector>
```

Classes

- class Scenes

  Represents a class that manages different scenes in a game.

Functions

- std::string keyToString (sf::Keyboard::Key key)

7.5.1 Function Documentation

7.5.1.1 keyToString()

```
std::string keyToString (
            sf::Keyboard::Key key )
```

7.6 /home/runner/work/R-Type/R-Type/Client/Src/keyToString.cpp File Reference

```
#include <SFML/Window/Keyboard.hpp>
#include <iostream>
```

Functions

- std::string keyToString (sf::Keyboard::Key key)

7.6.1 Function Documentation

7.6.1.1 keyToString()

```
std::string keyToString (
            sf::Keyboard::Key key )
```

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

```
#include <iostream>
#include <macro.hpp>
#include <scenes.hpp>
#include <sstream>
```

Functions

- static bool isValidIPv4 (const std::string &ip)
- static bool isValidPort (const std::string &portStr)
- int main (int const argc, char const *const *argv)

  The entry point of the program.

7.7.1 Function Documentation

7.7.1.1 isValidIPv4()

```
static bool isValidIPv4 (
            const std::string & ip )  [static]
```

7.7.1.2 isValidPort()

```
static bool isValidPort (
            const std::string & portStr )  [static]
```

7.7.1.3 main()

```
int main (
            int const argc,
            char const *const * argv )
```

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>
#include <errno.h>
#include <signal.h>
#include <stdio.h>
```

Functions

- void signal_handler (int signal)
- static bool isValidPort (const std::string &portStr)
- int main (int const argc, char const *const *const argv)

Variables

- static bool loopRunning = true

7.8.1 Function Documentation

7.8.1.1 isValidPort()

```
static bool isValidPort (
            const std::string & portStr )  [static]
```

7.8.1.2 main()

```
int main (
            int const argc,
            char const *const *const argv )
```

7.8.1.3 signal_handler()

```
void signal_handler (
            int signal )
```

7.8.2 Variable Documentation

7.8.2.1 loopRunning

```
bool loopRunning = true  [static]
```

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

```
#include <Components/components.hpp>
#include <Entities/entity_factory.hpp>
#include <Entities/entity_manager.hpp>
#include <Net/client.hpp>
#include <Systems/systems.hpp>
#include <audio_manager.hpp>
#include <chrono>
#include <creatable_client_object.hpp>
#include <font_manager.hpp>
#include <functional>
#include <iostream>
#include <scenes.hpp>
#include <sound_path.hpp>
#include <texture_manager.hpp>
```

Functions

- void reloadFilter (sf::RectangleShape &rectangle, AScenes::DaltonismMode mode)
- 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, FontManager fontManager, EntityFactory &entityFactory)
- sf::Keyboard::Key waitForKey (sf::RenderWindow *_window)
- void createKeyBindingButtons (std::vector< std::shared_ptr< Entity >> &buttons, ComponentManager &componentManager, EntityManager &entityManager, TextureManager &textureManager, FontManager fontManager, EntityFactory &entityFactory, std::map< Scenes::Actions, sf::Keyboard::Key > &keyBinds)

7.9.1 Function Documentation

7.9.1.1 createDaltonismChoiceButtons()

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

7.9.1.2 createKeyBindingButtons()

```
void createKeyBindingButtons (
            std::vector< std::shared_ptr< Entity >> & buttons,
            ComponentManager & componentManager,
            EntityManager & entityManager,
            TextureManager & textureManager,
            FontManager fontManager,
            EntityFactory & entityFactory,
            std::map< Scenes::Actions, sf::Keyboard::Key > & keyBinds )
```

7.9.1.3 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.

7.9.1.4 reloadFilter()

```
void reloadFilter (
            sf::RectangleShape & rectangle,
            AScenes::DaltonismMode mode )
```

7.9.1.5 waitForKey()

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

7.10 /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

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

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

Classes

- class AudioManager

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/animation_component.hpp File Reference

```
#include <macros.hpp>
```

Classes

- struct AnimationComponent

Functions

- bool operator!= (AnimationComponent animation, AnimationComponent other)

  Inequality operator for AnimationComponent.

7.14.1 Function Documentation

7.14.1.1 operator"!=()

```
bool operator!= (
            AnimationComponent animation,
            AnimationComponent other )
```

Inequality operator for AnimationComponent.

This operator compares two AnimationComponent objects to determine if they are not equal. Two AnimationComponent objects are considered not equal if any of their respective offset or dimension coordinates differ.

Parameters

animation	The first AnimationComponent to compare.
other	The second AnimationComponent to compare.

Returns

true if the AnimationComponent objects are not equal, false otherwise.

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

Classes

- struct BackgroundComponent

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

Classes

- struct BasicMonsterComponent

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

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

Classes

- struct BindComponent

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

Classes

- struct BossComponent

7.19 /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.

7.20 /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 "animation_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 "movement_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 "power_up_component.hpp"
#include "rectangleShapeComponent.hpp"
#include "score_component.hpp"
#include "shoot_component.hpp"
#include "sprite_component.hpp"
#include "sprite_data_component.hpp"
#include "text_component.hpp"
#include "text_data_component.hpp"
#include "velocity_component.hpp"
#include "weapon_component.hpp"
```

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

Classes

- struct EnemyComponent

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

Classes

- struct EnemyMissileComponent

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

Classes

- struct HealthComponent

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

Classes

- struct HitboxComponent

7.25 /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 }

7.25.1 Enumeration Type Documentation

7.25.1.1 InputType

```
enum InputType  [strong]
```

Enumerator

UP	
DOWN	
LEFT	
RIGHT	
SHOOT	
QUIT	
NONE	

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

```
#include <iostream>
```

Classes

- struct labelComponent

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

```
#include <cstdint>
```

Classes

- struct MovementComponent

Enumerations

- enum class MovementType { WIGGLE , DIAGONAL , CIRCLE }

7.27.1 Enumeration Type Documentation

7.27.1.1 MovementType

```
enum MovementType  [strong]
```

Enumerator

WIGGLE	
DIAGONAL	
CIRCLE	

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

Classes

- struct OffsetComponent

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

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

Classes

- struct OnClickComponent

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

Classes

- struct PlayerComponent

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

Classes

- struct PlayerMissileComponent

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

Classes

- struct PositionComponent

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

Classes

- struct PowerUpComponent

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

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

Classes

- struct RectangleShapeComponent

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

Classes

- struct ScoreComponent

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

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

Classes

- struct ShaderComponent

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

```
#include <chrono>
```

Classes

- struct ShootComponent

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

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

Classes

- struct SpriteComponent

7.39 /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 "animation_component.hpp"
#include "position_component.hpp"
#include <SFML/Graphics.hpp>
#include <a_scenes.hpp>
#include <cstdint>
#include <macros.hpp>
#include <string>
```

Classes

- struct SpriteDataComponent

Functions

- std::ostream & operator<< (std::ostream &os, const SpriteDataComponent &spriteData)

7.39.1 Function Documentation

7.39.1.1 operator<<()

```
std::ostream& operator<< (
            std::ostream & os,
            const SpriteDataComponent & spriteData )
```

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

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

Classes

- struct TextComponent

7.41 /home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/text_data_component.hpp File Reference

```
#include "../font_path.hpp"
#include "../game_text.hpp"
```

Classes

- struct TextDataComponent

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

Classes

- struct VelocityComponent

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

Classes

- struct WeaponComponent

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

```
#include <cstdint>
```

Enumerations

- enum class CreatableClientObject : uint32_t { PLAYERMISSILE , NONE }

7.44.1 Enumeration Type Documentation

7.44.1.1 CreatableClientObject

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

Enumerator

PLAYERMISSILE	
NONE	

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

Classes

- class Entity

  Represents an entity in the ECS system.

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

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

Classes

- class EntityFactory

  A class responsible for creating different types of entities.

7.47 /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 <memory>
#include <optional>
#include <vector>
```

Classes

- class EntityManager

  Class responsible for managing entities in the ECS system.

7.48 /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 "font_manager.hpp"
#include "texture_manager.hpp"
```

Classes

- class IEntityFactory

  The interface for an entity factory.

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

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

Classes

- struct EntityInformation

  Represents information about an entity.
- struct UIEntityInformation

7.50 /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.
- class failedToLoadSound
- class failedToLoadFont
- class playerIdNotFound

7.51 /home/runner/work/R-Type/R-Type/ECS/Interface/Include/font_manager.hpp File Reference

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

Classes

- class FontManager

7.52 /home/runner/work/R-Type/R-Type/ECS/Interface/Include/font_path.hpp File Reference

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

Enumerations

- enum class FontPath : uint32_t { MAIN , NONE }

Functions

- std::string FontFactory (FontPath font)
- std::ostream & operator<< (std::ostream &os, const FontPath &fontPath)

7.52.1 Enumeration Type Documentation

7.52.1.1 FontPath

```
enum FontPath :  uint32_t  [strong]
```

Enumerator

MAIN	
NONE	

7.52.2 Function Documentation

7.52.2.1 FontFactory()

```
std::string FontFactory (
            FontPath font )
```

7.52.2.2 operator<<()

```
std::ostream& operator<< (
            std::ostream & os,
            const FontPath & fontPath )
```

7.53 /home/runner/work/R-Type/R-Type/ECS/Interface/Include/game_text.hpp File Reference

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

Enumerations

- enum class GameText : uint32_t { Lives , Score , NONE }

Functions

- std::string GameTextFactory (GameText text)
- std::ostream & operator<< (std::ostream &os, const GameText &text)

7.53.1 Enumeration Type Documentation

7.53.1.1 GameText

```
enum GameText :  uint32_t  [strong]
```

Enumerator

Lives	
Score	
NONE	

7.53.2 Function Documentation

7.53.2.1 GameTextFactory()

```
std::string GameTextFactory (
            GameText text )
```

7.53.2.2 operator<<()

```
std::ostream& operator<< (
            std::ostream & os,
            const GameText & text )
```

7.54 /home/runner/work/R-Type/R-Type/ECS/Interface/Include/hitbox_↩ tmp.hpp File Reference

```
#include <Components/component_manager.hpp>
#include <Entities/entity.hpp>
#include <Entities/entity_manager.hpp>
#include <entity_struct.hpp>
```

Functions

- int CheckEntityPosition (uint32_t entityId, ComponentManager componentManager, EntityManager entity↩ Manager)
- int CheckEntityMovement (EntityInformation desc, ComponentManager componentManager, EntityManager entityManager)

7.54.1 Function Documentation

7.54.1.1 CheckEntityMovement()

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

7.54.1.2 CheckEntityPosition()

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

7.55 /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.

7.56 /home/runner/work/R-Type/R-Type/ECS/Interface/Include/macros.hpp File Reference

Classes

- struct vf2d

  Represents a 2D vector with x and y coordinates.

Macros

- #define SCREEN_WIDTH 1920
- #define SCREEN_HEIGHT 1080

7.56.1 Macro Definition Documentation

7.56.1.1 SCREEN_HEIGHT

```
#define SCREEN_HEIGHT 1080
```

7.56.1.2 SCREEN_WIDTH

```
#define SCREEN_WIDTH 1920
```

7.57 /home/runner/work/R-Type/R-Type/ECS/Interface/Include/sound_path.hpp File Reference

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

Enumerations

- enum class ActionType : uint32_t {
  Win , Shot , Boss , PowerUp ,
  GameOver , BossDeath , Explosion , Background ,
  NONE }

Functions

- std::string SoundFactory (ActionType action)

7.57.1 Enumeration Type Documentation

7.57.1.1 ActionType

```
enum ActionType :  uint32_t  [strong]
```

Enumerator

Win	
Shot	
Boss	
PowerUp	
GameOver	
BossDeath	
Explosion	

7.57.2 Function Documentation

7.57.2.1 SoundFactory()

```
std::string SoundFactory (
            ActionType action )
```

7.58 /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 , Missile , Weapon ,
  BlueLaserCrystal , Background1 , Background2 , Background3 ,
  Explosion , PowerUp , Boss , BossBullet ,
  Bar , NONE }

Functions

- std::string SpriteFactory (SpritePath sprite)
- std::ostream & operator<< (std::ostream &os, const SpritePath &spritePath)

7.58.1 Enumeration Type Documentation

Enumerator

7.58.1.1 SpritePath

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

Enumerator

Ship1	
Ship2	
Ship3	
Ship4	
Enemy1	
Enemy2	
Enemy3	
Enemy4	
Enemy5	
Enemy6	
Missile	
Weapon	
BlueLaserCrystal	
Background1	
Background2	
Background3	
Explosion	
PowerUp	
Boss	
BossBullet	
Bar	
NONE	

7.58.2 Function Documentation

7.58.2.1 operator<<()

```
std::ostream& operator<< (
            std::ostream & os,
            const SpritePath & spritePath )
```

7.58.2.2 SpriteFactory()

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

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

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

Classes

- class AnimationSystem

Enumerations

- enum class AnimationShip : uint32_t {
  SHIP_DOWN , SHIP_FLIP_DOWN , SHIP_STRAIT , SHIP_FLIP_UP ,
  SHIP_UP }
- enum class AnimationBasicMonster : uint32_t {
  BASIC_MONSTER_DEFAULT , BASIC_MONSTER_1 , BASIC_MONSTER_2 , BASIC_MONSTER_3 ,
  BASIC_MONSTER_4 , BASIC_MONSTER_5 , BASIC_MONSTER_6 , BASIC_MONSTER_7 }
- enum class AnimationWeapon1 : uint32_t {
  WEAPON_1_DEFAULT , WEAPON_1_1 , WEAPON_1_2 , WEAPON_1_3 ,
  WEAPON_1_4 , WEAPON_1_5 }

Functions

- bool operator!= (AnimationComponent animation, AnimationComponent other)

  get if two animations are different.
- vf2d animationShipFactory (AnimationShip animation)

  Factory function to create a ship animation.

7.59.1 Enumeration Type Documentation

7.59.1.1 AnimationBasicMonster

```
enum AnimationBasicMonster :  uint32_t  [strong]
```

Enumerator

BASIC_MONSTER_DEFAULT	
BASIC_MONSTER_1	
BASIC_MONSTER_2	
BASIC_MONSTER_3	
BASIC_MONSTER_4	
BASIC_MONSTER_5	
BASIC_MONSTER_6	
BASIC_MONSTER_7	

7.59.1.2 AnimationShip

```
enum AnimationShip : uint32_t [strong]
```

Enumerator

SHIP_DOWN	Ship animation when going down.
SHIP_FLIP_DOWN	Ship animation when flipping down.
SHIP_STRAIT	Ship animation when going strait.
SHIP_FLIP_UP	Ship animation when flipping up.
SHIP_UP	Ship animation when going up.

7.59.1.3 AnimationWeapon1

```
enum AnimationWeapon1 : uint32_t [strong]
```

Enumerator

WEAPON_1_DEFAULT	
WEAPON_1_1	
WEAPON_1_2	
WEAPON_1_3	
WEAPON_1_4	
WEAPON_1_5	

7.59.2 Function Documentation

7.59.2.1 animationShipFactory()

```
vf2d animationShipFactory (
            AnimationShip animation )
```

Factory function to create a ship animation.

This function takes an AnimationShip object and generates a corresponding vf2d object that represents the animation of the ship.

Parameters

animation	The AnimationShip object containing the animation details.

Returns

vf2d The generated animation for the ship.

Factory function to create a ship animation.

This function takes an AnimationShip enumeration value and returns a vf2d vector that corresponds to the animation state of the ship.

Parameters

animation	The animation state of the ship, represented by the AnimationShip enumeration.

Returns

vf2d A vector representing the animation state of the ship. The x-coordinate of the vector corresponds to the frame position, and the y-coordinate is always -1 for valid states. If the animation state is not recognized, the function returns {0, 0}.

7.59.2.2 operator"!=()

```
bool operator!= (
            AnimationComponent animation,
            AnimationComponent other )
```

get if two animations are different.

Parameters

animation	The first animation.
other	The second animation.

Returns

bool true if the animations are different, false otherwise.

get if two animations are different.

This operator compares two AnimationComponent objects to determine if they are not equal. Two AnimationComponent objects are considered not equal if any of their respective offset or dimension coordinates differ.

Parameters

animation	The first AnimationComponent to compare.
other	The second AnimationComponent to compare.

Returns

true if the AnimationComponent objects are not equal, false otherwise.

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

```
#include <SFML/Audio.hpp>
#include <Systems/i_system.hpp>
#include <audio_manager.hpp>
#include <error_handling.hpp>
#include <memory>
#include <string>
```

Classes

- class AudioSystem

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

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

Classes

- class AutoFireSystem

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

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

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

Classes

- class CollisionSystem

7.64 /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

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

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

Classes

- class MoveSystem

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

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

Classes

- class RenderSystem

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

```
#include <Systems/animation_system.hpp>
#include <Systems/audio_system.hpp>
#include <Systems/auto_fire_system.hpp>
#include <Systems/collision_system.hpp>
#include <Systems/move_system.hpp>
#include <Systems/render_system.hpp>
#include <Systems/update_system.hpp>
```

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

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

Classes

- class UpdateSystem

7.69 /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

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

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

7.71 /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>
#include <macros.hpp>
```

Functions

- std::ostream & operator<< (std::ostream &os, const SpritePath &spritePath)
- std::ostream & operator<< (std::ostream &os, const AScenes::SpriteType &spriteType)
- std::ostream & operator<< (std::ostream &os, const SpriteDataComponent &spriteData)

7.71.1 Function Documentation

7.71.1.1 operator<<() [1/3]

```
std::ostream& operator<< (
            std::ostream & os,
            const AScenes::SpriteType & spriteType )
```

7.71.1.2 operator<<() [2/3]

```
std::ostream& operator<< (
            std::ostream & os,
            const SpriteDataComponent & spriteData )
```

7.71.1.3 operator<<() [3/3]

```
std::ostream& operator<< (
            std::ostream & os,
            const SpritePath & spritePath )
```

7.72 /home/runner/work/R-Type/R-Type/ECS/Src/font_path.cpp File Reference

```
#include <font_path.hpp>
```

Functions

- std::string FontFactory (FontPath font)

7.72.1 Function Documentation

7.72.1.1 FontFactory()

```
std::string FontFactory (
            FontPath font )
```

7.73 /home/runner/work/R-Type/R-Type/ECS/Src/game_text.cpp File Reference

```
#include <game_text.hpp>
```

Functions

- std::string GameTextFactory (GameText text)

7.73.1 Function Documentation

7.73.1.1 GameTextFactory()

```
std::string GameTextFactory (
            GameText text )
```

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

```
#include "hitbox_tmp.hpp"
#include <macros.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)

7.74.1 Function Documentation

7.74.1.1 CheckCollisionLogic()

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

7.74.1.2 CheckEntityMovement()

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

7.74.1.3 CheckEntityPosition()

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

7.75 /home/runner/work/R-Type/R-Type/ECS/Src/sound_path.cpp File Reference

```
#include <sound_path.hpp>
```

Functions

- std::string SoundFactory (ActionType action)

7.75.1 Function Documentation

7.75.1.1 SoundFactory()

```
std::string SoundFactory (
            ActionType action )
```

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

```
#include <sprite_path.hpp>
```

Functions

- std::string SpriteFactory (SpritePath sprite)

7.76.1 Function Documentation

7.76.1.1 SpriteFactory()

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

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

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

Functions

- vf2d animationShipFactory (AnimationShip animation)

  Generates a vector representing the animation state of a ship.
- vf2d animationBasicMonsterFactory (AnimationBasicMonster animation)
- vf2d animationWeapon1Factory (AnimationWeapon1 animation)
- bool operator!= (AnimationComponent animation, AnimationComponent other)

  Inequality operator for AnimationComponent.

7.77.1 Function Documentation

7.77.1.1 animationBasicMonsterFactory()

```
vf2d animationBasicMonsterFactory (
            AnimationBasicMonster animation )
```

7.77.1.2 animationShipFactory()

```
vf2d animationShipFactory (
            AnimationShip animation )
```

Generates a vector representing the animation state of a ship.

Factory function to create a ship animation.

This function takes an AnimationShip enumeration value and returns a vf2d vector that corresponds to the animation state of the ship.

Parameters

animation	The animation state of the ship, represented by the AnimationShip enumeration.

Returns

vf2d A vector representing the animation state of the ship. The x-coordinate of the vector corresponds to the frame position, and the y-coordinate is always -1 for valid states. If the animation state is not recognized, the function returns {0, 0}.

7.77.1.3 animationWeapon1Factory()

```
vf2d animationWeapon1Factory (
            AnimationWeapon1 animation )
```

7.77.1.4 operator"!=()

```
bool operator!= (
            AnimationComponent animation,
            AnimationComponent other )
```

Inequality operator for AnimationComponent.

get if two animations are different.

This operator compares two AnimationComponent objects to determine if they are not equal. Two AnimationComponent objects are considered not equal if any of their respective offset or dimension coordinates differ.

Parameters

animation	The first AnimationComponent to compare.
other	The second AnimationComponent to compare.

Returns

true if the AnimationComponent objects are not equal, false otherwise.

7.78 /home/runner/work/R-Type/R-Type/ECS/Src/Systems/audio_system.cpp File Reference

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

7.79 /home/runner/work/R-Type/R-Type/ECS/Src/Systems/auto_fire_system.cpp File Reference

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

7.80 /home/runner/work/R-Type/R-Type/ECS/Src/Systems/collision_system.cpp File Reference

```
#include <Systems/collision_system.hpp>
#include <macros.hpp>
#include <vector>
```

7.81 /home/runner/work/R-Type/R-Type/ECS/Src/Systems/move_system.cpp File Reference

```
#include <Systems/move_system.hpp>
#include <cmath>
```

7.82 /home/runner/work/R-Type/R-Type/ECS/Src/Systems/render_system.cpp File Reference

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

7.83 /home/runner/work/R-Type/R-Type/ECS/Src/Systems/update_system.cpp File Reference

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

7.84 /home/runner/work/R-Type/R-Type/Server/Interface/Include/level.hpp File Reference

```
#include <Components/component_manager.hpp>
#include <Components/components.hpp>
#include <cmath>
#include <i_level.hpp>
```

Classes

- class r_type::Level< T >

Namespaces

- r_type
- r_type::net

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

```
#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 <Systems/systems.hpp>
#include <cmath>
#include <entity_struct.hpp>
#include <error_handling.hpp>
#include <level.hpp>
#include <macros.hpp>
#include <unordered_map>
```

Classes

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

  AServer class template for managing server operations.

Namespaces

- r_type
- r_type::net

7.86 /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

7.87 /home/runner/work/R-Type/R-Type/Server/Interface/Include/r_type-server.hpp File Reference

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

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

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

Index