

R-Type

Generated by Doxygen 1.9.1



<b>1 Namespace Index</b>	<b>1</b>
1.1 Namespace List	1
<b>2 Hierarchical Index</b>	<b>3</b>
2.1 Class Hierarchy	3
<b>3 Class Index</b>	<b>5</b>
3.1 Class List	5
<b>4 File Index</b>	<b>7</b>
4.1 File List	7
<b>5 Namespace Documentation</b>	<b>9</b>
5.1 r_type Namespace Reference	9
5.2 r_type::net Namespace Reference	9
<b>6 Class Documentation</b>	<b>11</b>
6.1 r_type::net::AClient< T > Class Template Reference	11
6.1.1 Constructor & Destructor Documentation	12
6.1.1.1 AClient()	12
6.1.1.2 ~AClient()	12
6.1.2 Member Function Documentation	12
6.1.2.1 addEntity()	12
6.1.2.2 Connect()	12
6.1.2.3 Disconnect()	13
6.1.2.4 getConnection()	13
6.1.2.5 getPlayerId()	13
6.1.2.6 Incoming()	13
6.1.2.7 isConnected()	14
6.1.2.8 removeEntity()	14
6.1.2.9 Send()	14
6.1.2.10 setPlayerId()	14
6.1.2.11 updateEntity()	15
6.1.3 Member Data Documentation	15
6.1.3.1 m_connection	15
6.1.3.2 m_context	15
6.1.3.3 m_qMessagesIn	15
6.1.3.4 playerId	15
6.1.3.5 thrContext	15
6.2 AllyComponent Struct Reference	16
6.3 AllyMissileComponent Struct Reference	16
6.4 r_type::net::AServer< T > Class Template Reference	16
6.4.1 Detailed Description	18
6.4.2 Constructor & Destructor Documentation	18

6.4.2.1 AServer()	18
6.4.2.2 ~AServer()	18
6.4.3 Member Function Documentation	18
6.4.3.1 CheckPlayerPosition()	18
6.4.3.2 GetClientEntityId()	19
6.4.3.3 InitiateBackground()	19
6.4.3.4 InitiateMissile()	19
6.4.3.5 InitiatePlayers()	20
6.4.3.6 InitListEntities()	20
6.4.3.7 MessageAllClients()	21
6.4.3.8 MessageClient()	21
6.4.3.9 OnClientConnect()	21
6.4.3.10 OnClientDisconnect()	22
6.4.3.11 OnClientValidated()	22
6.4.3.12 OnMessage()	22
6.4.3.13 RemoveEntities()	22
6.4.3.14 RemovePlayer()	23
6.4.3.15 Start()	23
6.4.3.16 Stop()	23
6.4.3.17 Update()	24
6.4.3.18 UpdateEntityPosition()	24
6.4.3.19 WaitForClientMessage()	24
6.4.4 Member Data Documentation	25
6.4.4.1 _clock	25
6.4.4.2 background	25
6.4.4.3 clientPlayerID	25
6.4.4.4 componentManager	25
6.4.4.5 entityFactory	25
6.4.4.6 entityManager	26
6.4.4.7 m_asioContext	26
6.4.4.8 m_asioSocket	26
6.4.4.9 m_clientEndpoint	26
6.4.4.10 m_deqConnections	26
6.4.4.11 m_qMessagesIn	26
6.4.4.12 m_tempBuffer	26
6.4.4.13 m_threadContext	27
6.4.4.14 nbrOfPlayers	27
6.4.4.15 nIDCounter	27
6.5 BackgroundComponent Struct Reference	27
6.6 BasicMonsterComponent Struct Reference	27
6.7 BindComponent Struct Reference	27
6.7.1 Constructor & Destructor Documentation	28

6.7.1.1 BindComponent()	28
6.7.2 Member Data Documentation	28
6.7.2.1 bind	28
6.7.2.2 isHovered	28
6.8 r_type::net::Client Class Reference	28
6.8.1 Member Function Documentation	29
6.8.1.1 addEntity()	29
6.8.1.2 MessageAll()	29
6.8.1.3 PingServer()	29
6.8.1.4 removeEntity()	29
6.8.1.5 updateEntity()	30
6.9 ComponentManager Class Reference	30
6.9.1 Detailed Description	30
6.9.2 Member Function Documentation	30
6.9.2.1 addComponent()	30
6.9.2.2 GetComponent()	31
6.9.2.3 GetComponentMap()	31
6.9.2.4 removeEntityFromComponent()	32
6.9.3 Member Data Documentation	32
6.9.3.1 components	32
6.10 componentNotFound Class Reference	32
6.10.1 Detailed Description	33
6.10.2 Member Function Documentation	33
6.10.2.1 what()	33
6.11 CreatableClientObject Class Reference	33
6.11.1 Detailed Description	33
6.12 EnemyComponent Struct Reference	33
6.13 EnemyMissileComponent Struct Reference	34
6.14 Entity Class Reference	34
6.14.1 Detailed Description	34
6.14.2 Constructor & Destructor Documentation	34
6.14.2.1 Entity()	34
6.14.3 Member Function Documentation	35
6.14.3.1 getId()	35
6.14.4 Member Data Documentation	35
6.14.4.1 _id	35
6.15 EntityFactory Class Reference	35
6.15.1 Detailed Description	36
6.15.2 Member Function Documentation	36
6.15.2.1 createAlly()	36
6.15.2.2 createAllyMissile()	37
6.15.2.3 createBackground()	37

6.15.2.4 createBasicEnemy()	38
6.15.2.5 createBasicMonster()	38
6.15.2.6 createButton()	39
6.15.2.7 createEnemyMissile()	39
6.15.2.8 createPlayer()	40
6.15.2.9 createPlayerMissile()	40
6.15.2.10 createSmallButton()	41
6.16 EntityInformation Struct Reference	41
6.16.1 Detailed Description	42
6.16.2 Member Data Documentation	42
6.16.2.1 spriteData	42
6.16.2.2 uniqueID	42
6.16.2.3 vPos	42
6.17 EntityManager Class Reference	42
6.17.1 Detailed Description	43
6.17.2 Member Function Documentation	43
6.17.2.1 createEntity()	43
6.17.2.2 getAllEntities()	43
6.17.2.3 getEntity()	43
6.17.2.4 removeEntity()	44
6.17.3 Member Data Documentation	44
6.17.3.1 entities	44
6.17.3.2 entityNb	44
6.18 entityNotFound Class Reference	45
6.18.1 Detailed Description	45
6.18.2 Member Function Documentation	45
6.18.2.1 what()	45
6.19 failedToLoadTexture Class Reference	45
6.19.1 Detailed Description	46
6.19.2 Member Function Documentation	46
6.19.2.1 what()	46
6.20 HealthComponent Struct Reference	46
6.20.1 Member Data Documentation	46
6.20.1.1 health	46
6.20.1.2 max_health	47
6.21 HitboxComponent Struct Reference	47
6.21.1 Member Data Documentation	47
6.21.1.1 h	47
6.21.1.2 w	47
6.22 r_type::net::IClient< T > Class Template Reference	47
6.22.1 Constructor & Destructor Documentation	48
6.22.1.1 IClient()	48

6.22.1.2 ~IClient()	48
6.22.2 Member Function Documentation	48
6.22.2.1 Connect()	48
6.22.2.2 Disconnect()	49
6.22.2.3 Incoming()	49
6.22.2.4 IsConnected()	49
6.22.2.5 Send()	49
6.23 IEntity Class Reference	50
6.23.1 Detailed Description	50
6.23.2 Constructor & Destructor Documentation	50
6.23.2.1 ~IEntity()	51
6.23.3 Member Function Documentation	51
6.23.3.1 getId()	51
6.24 IEntityFactory Class Reference	51
6.24.1 Detailed Description	52
6.24.2 Constructor & Destructor Documentation	52
6.24.2.1 ~IEntityFactory()	52
6.24.3 Member Function Documentation	53
6.24.3.1 createAlly()	53
6.24.3.2 createAllyMissile()	53
6.24.3.3 createBackground()	54
6.24.3.4 createBasicEnemy()	54
6.24.3.5 createBasicMonster()	54
6.24.3.6 createButton()	55
6.24.3.7 createEnemyMissile()	55
6.24.3.8 createPlayer()	56
6.24.3.9 createPlayerMissile()	56
6.25 InputComponent Struct Reference	57
6.25.1 Member Data Documentation	57
6.25.1.1 input	57
6.26 ISystem Class Reference	57
6.26.1 Constructor & Destructor Documentation	58
6.26.1.1 ISystem()	58
6.27 labelComponent Struct Reference	58
6.27.1 Member Data Documentation	58
6.27.1.1 name	58
6.27.1.2 x	58
6.27.1.3 y	59
6.28 OffsetComponent Struct Reference	59
6.28.1 Member Data Documentation	59
6.28.1.1 offset	59
6.29 OnClickComponent Struct Reference	59

6.29.1 Constructor & Destructor Documentation	60
6.29.1.1 OnClickComponent()	60
6.29.2 Member Data Documentation	60
6.29.2.1 isClicked	60
6.29.2.2 onClick	60
6.30 PlayerComponent Struct Reference	60
6.31 PlayerMissileComponent Struct Reference	60
6.32 PositionComponent Struct Reference	61
6.32.1 Constructor & Destructor Documentation	61
6.32.1.1 PositionComponent()	61
6.32.2 Member Data Documentation	61
6.32.2.1 x	61
6.32.2.2 y	61
6.33 RenderSystem Class Reference	62
6.33.1 Detailed Description	62
6.33.2 Constructor & Destructor Documentation	62
6.33.2.1 RenderSystem()	62
6.33.3 Member Function Documentation	63
6.33.3.1 render()	63
6.33.4 Member Data Documentation	63
6.33.4.1 _window	63
6.34 Rtype Class Reference	63
6.34.1 Constructor & Destructor Documentation	64
6.34.1.1 Rtype()	64
6.34.2 Member Function Documentation	64
6.34.2.1 gameLoop()	64
6.34.2.2 handleEvents()	64
6.34.2.3 mainMenu()	65
6.34.2.4 processServerMessages()	65
6.34.2.5 renderGame()	65
6.34.2.6 run()	65
6.34.2.7 updateGame()	65
6.34.3 Member Data Documentation	65
6.34.3.1 _scenes	65
6.34.3.2 _window	66
6.35 Scenes Class Reference	66
6.35.1 Detailed Description	67
6.35.2 Member Enumeration Documentation	67
6.35.2.1 Actions	67
6.35.2.2 DaltonismMode	68
6.35.2.3 GameMode	68
6.35.2.4 Scene	68



6.35.3 Constructor & Destructor Documentation	69
6.35.3.1 Scenes()	69
6.35.3.2 ~Scenes()	69
6.35.4 Member Function Documentation	69
6.35.4.1 gameLoop()	69
6.35.4.2 getPreviousScene()	70
6.35.4.3 getRenderWindow()	70
6.35.4.4 inGameMenu()	71
6.35.4.5 mainMenu()	71
6.35.4.6 render()	71
6.35.4.7 setDaltonism()	71
6.35.4.8 setGameMode()	72
6.35.4.9 setScene()	72
6.35.4.10 settingsMenu()	72
6.35.4.11 shouldQuit()	72
6.35.5 Member Data Documentation	73
6.35.5.1 _window	73
6.35.5.2 binding	73
6.35.5.3 buttons	73
6.35.5.4 currentDaltonismMode	73
6.35.5.5 currentGameMode	73
6.35.5.6 currentScene	73
6.35.5.7 displayDaltonismChoice	74
6.35.5.8 displayGameModeChoice	74
6.35.5.9 displayKeyBinds	74
6.35.5.10 keyBinds	74
6.35.5.11 previousScene	74
6.36 ScoreComponent Struct Reference	74
6.36.1 Member Data Documentation	75
6.36.1.1 score	75
6.37 r_type::net::Server Class Reference	75
6.37.1 Constructor & Destructor Documentation	75
6.37.1.1 Server()	76
6.37.1.2 ~Server()	76
6.37.2 Member Function Documentation	76
6.37.2.1 OnClientConnect()	76
6.37.2.2 OnClientDisconnect()	76
6.37.2.3 OnMessage()	77
6.38 SpriteComponent Struct Reference	77
6.38.1 Constructor & Destructor Documentation	77
6.38.1.1 SpriteComponent()	77
6.38.2 Member Data Documentation	77

6.38.2.1 sprite . . . . .	78
6.39 SpriteDataComponent Struct Reference . . . . .	78
6.39.1 Member Data Documentation . . . . .	78
6.39.1.1 dimension . . . . .	78
6.39.1.2 offSet . . . . .	78
6.39.1.3 scale . . . . .	78
6.39.1.4 spritePath . . . . .	79
6.40 TextComponent Struct Reference . . . . .	79
6.40.1 Constructor & Destructor Documentation . . . . .	79
6.40.1.1 TextComponent() . . . . .	79
6.40.2 Member Data Documentation . . . . .	79
6.40.2.1 _text . . . . .	79
6.41 TextureManager Class Reference . . . . .	80
6.41.1 Member Function Documentation . . . . .	80
6.41.1.1 getTexture() . . . . .	80
6.41.2 Member Data Documentation . . . . .	80
6.41.2.1 textures . . . . .	81
6.42 UpdateSystem Class Reference . . . . .	81
6.42.1 Detailed Description . . . . .	81
6.42.2 Constructor & Destructor Documentation . . . . .	81
6.42.2.1 UpdateSystem() . . . . .	82
6.42.3 Member Function Documentation . . . . .	82
6.42.3.1 getGameBgOffset() . . . . .	82
6.42.3.2 setGameBgOffset() . . . . .	82
6.42.3.3 update() . . . . .	82
6.42.3.4 updateBackground() . . . . .	82
6.42.3.5 updateSpritePosition() . . . . .	82
6.42.4 Member Data Documentation . . . . .	83
6.42.4.1 _window . . . . .	83
6.42.4.2 gameBgOffset . . . . .	83
6.43 Vector< T > Struct Template Reference . . . . .	83
6.43.1 Member Data Documentation . . . . .	83
6.43.1.1 x . . . . .	83
6.43.1.2 y . . . . .	83
6.44 VelocityComponent Struct Reference . . . . .	84
6.44.1 Member Data Documentation . . . . .	84
6.44.1.1 speed . . . . .	84
6.45 vf2d Struct Reference . . . . .	84
6.45.1 Detailed Description . . . . .	84
6.45.2 Member Data Documentation . . . . .	84
6.45.2.1 x . . . . .	84
6.45.2.2 y . . . . .	85

6.46 WeaponComponent Struct Reference . . . . .	85
6.46.1 Member Data Documentation . . . . .	85
6.46.1.1 bullet_lifetime . . . . .	85
6.46.1.2 bullet_speed . . . . .	85
6.46.1.3 damage . . . . .	85
6.46.1.4 fire_rate . . . . .	85
<b>7 File Documentation . . . . .</b>	<b>87</b>
7.1 /home/runner/work/R-Type/R-Type/Client/Interface/Include/mainmenu.hpp File Reference . . . . .	87
7.1.1 Function Documentation . . . . .	87
7.1.1.1 MainMenu() . . . . .	87
7.2 /home/runner/work/R-Type/R-Type/Client/Interface/Include/Net/a_client.hpp File Reference . . . . .	87
7.3 /home/runner/work/R-Type/R-Type/Client/Interface/Include/Net/client.hpp File Reference . . . . .	88
7.4 /home/runner/work/R-Type/R-Type/Client/Interface/Include/Net/i_client.hpp File Reference . . . . .	88
7.5 /home/runner/work/R-Type/R-Type/Client/Interface/Include/r_type_client.hpp File Reference . . . . .	89
7.6 /home/runner/work/R-Type/R-Type/Client/Src/main.cpp File Reference . . . . .	89
7.6.1 Function Documentation . . . . .	89
7.6.1.1 main() . . . . .	89
7.7 /home/runner/work/R-Type/R-Type/Server/Src/main.cpp File Reference . . . . .	89
7.7.1 Function Documentation . . . . .	90
7.7.1.1 main() . . . . .	90
7.8 /home/runner/work/R-Type/R-Type/Client/Src/r_type_client.cpp File Reference . . . . .	90
7.9 /home/runner/work/R-Type/R-Type/Client/Src/scenes.cpp File Reference . . . . .	90
7.9.1 Function Documentation . . . . .	91
7.9.1.1 createDaltonismChoiceButtons() . . . . .	91
7.9.1.2 createGameModeChoiceButtons() . . . . .	91
7.9.1.3 createKeyBindingButtons() . . . . .	91
7.9.1.4 handleEvents() . . . . .	91
7.9.1.5 waitForKey() . . . . .	92
7.10 /home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/ally_component.hpp File Reference . . . . .	92
7.11 /home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/ally_missile_component.hpp File Reference . . . . .	92
7.12 /home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/background_component.hpp File Reference . . . . .	92
7.13 /home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/basic_monster_component.hpp File Reference . . . . .	92
7.14 /home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/bind_component.hpp File Reference . . . . .	92
7.15 /home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/component_manager.hpp File Reference . . . . .	93
7.16 /home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/components.hpp File Reference . . . . .	93
7.17 /home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/enemy_component.hpp File Reference . . . . .	94

7.18	/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/enemy_missile_component.hpp File Reference	94
7.19	/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/health_component.hpp File Reference	94
7.20	/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/hitbox_component.hpp File Reference	94
7.21	/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/input_component.hpp File Reference	94
7.21.1	Enumeration Type Documentation	94
7.21.1.1	InputType	94
7.22	/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/label_component.hpp File Reference	95
7.23	/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/offset_component.hpp File Reference	95
7.24	/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/on_click_component.hpp File Reference	95
7.25	/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/player_component.hpp File Reference	95
7.26	/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/player_missile_component.hpp File Reference	96
7.27	/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/position_component.hpp File Reference	96
7.28	/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/score_component.hpp File Reference	96
7.29	/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/sprite_component.hpp File Reference	96
7.30	/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/sprite_data_component.hpp File Reference	96
7.31	/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/text_component.hpp File Reference	97
7.32	/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/velocity_component.hpp File Reference	97
7.33	/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/weapon_component.hpp File Reference	97
7.34	/home/runner/work/R-Type/R-Type/ECS/Interface/Include/creatable_client_object.hpp File Reference	97
7.34.1	Enumeration Type Documentation	97
7.34.1.1	CreatableClientObject	97
7.35	/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Entities/entity.hpp File Reference	98
7.36	/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Entities/entity_factory.hpp File Reference	98
7.37	/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Entities/entity_manager.hpp File Reference	98
7.38	/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Entities/i_entity.hpp File Reference	99
7.39	/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Entities/i_entity_factory.hpp File Reference	99
7.40	/home/runner/work/R-Type/R-Type/ECS/Interface/Include/entity_struct.hpp File Reference	99
7.41	/home/runner/work/R-Type/R-Type/ECS/Interface/Include/error_handling.hpp File Reference	99
7.42	/home/runner/work/R-Type/R-Type/ECS/Interface/Include/scenes.hpp File Reference	100
7.43	/home/runner/work/R-Type/R-Type/ECS/Interface/Include/sprite_path.hpp File Reference	100

7.43.1 Enumeration Type Documentation . . . . .	100
7.43.1.1 SpritePath . . . . .	100
7.43.2 Function Documentation . . . . .	101
7.43.2.1 SpriteFactory() . . . . .	101
7.44 /home/runner/work/R-Type/R-Type/ECS/Interface/Include/Systems/button_system.hpp File Reference	101
7.45 /home/runner/work/R-Type/R-Type/ECS/Interface/Include/Systems/i_system.hpp File Reference . .	101
7.46 /home/runner/work/R-Type/R-Type/ECS/Interface/Include/Systems/render_system.hpp File Reference	102
7.47 /home/runner/work/R-Type/R-Type/ECS/Interface/Include/Systems/systems.hpp File Reference . . .	102
7.48 /home/runner/work/R-Type/R-Type/ECS/Interface/Include/Systems/update_system.hpp File Reference	102
7.49 /home/runner/work/R-Type/R-Type/ECS/Interface/Include/texture_manager.hpp File Reference . . .	102
7.50 /home/runner/work/R-Type/R-Type/ECS/Src/Entities/entity_factory.cpp File Reference . . . . .	103
7.50.1 Function Documentation . . . . .	103
7.50.1.1 CheckPositionEntity() . . . . .	103
7.51 /home/runner/work/R-Type/R-Type/ECS/Src/sprite_path.cpp File Reference . . . . .	103
7.51.1 Function Documentation . . . . .	103
7.51.1.1 SpriteFactory() . . . . .	104
7.52 /home/runner/work/R-Type/R-Type/ECS/Src/Systems/render_system.cpp File Reference . . . . .	104
7.53 /home/runner/work/R-Type/R-Type/ECS/Src/Systems/update_system.cpp File Reference . . . . .	104
7.54 /home/runner/work/R-Type/R-Type/Server/Interface/Include/Net/a_server.hpp File Reference . . . .	104
7.55 /home/runner/work/R-Type/R-Type/Server/Interface/Include/Net/server.hpp File Reference . . . .	104
7.56 /home/runner/work/R-Type/R-Type/Server/Interface/Include/r_type_server.hpp File Reference . . .	105
7.57 /home/runner/work/R-Type/R-Type/Server/Src/server.cpp File Reference . . . . .	105
<b>Index</b>	<b>107</b>



# Chapter 1

## Namespace Index

### 1.1 Namespace List

Here is a list of all namespaces with brief descriptions:

<a href="#">r_type</a>	.....	9
<a href="#">r_type::net</a>	.....	9





## Chapter 2

# Hierarchical Index

### 2.1 Class Hierarchy

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

AllyComponent . . . . .	16
AllyMissileComponent . . . . .	16
BackgroundComponent . . . . .	27
BasicMonsterComponent . . . . .	27
BindComponent . . . . .	27
ComponentManager . . . . .	30
CreatableClientObject . . . . .	33
EnemyComponent . . . . .	33
EnemyMissileComponent . . . . .	34
EntityInformation . . . . .	41
EntityManager . . . . .	42
std::exception	
componentNotFound . . . . .	32
entityNotFound . . . . .	45
failedToLoadTexture . . . . .	45
HealthComponent . . . . .	46
HitboxComponent . . . . .	47
r_type::net::IClient< T > . . . . .	47
r_type::net::AClient< TypeMessage > . . . . .	11
r_type::net::Client . . . . .	28
r_type::net::AClient< T > . . . . .	11
IEntity . . . . .	50
Entity . . . . .	34
IEntityFactory . . . . .	51
EntityFactory . . . . .	35
InputComponent . . . . .	57
r_type::net::IServer	
r_type::net::AServer< TypeMessage > . . . . .	16
r_type::net::Server . . . . .	75
r_type::net::AServer< T > . . . . .	16
ISystem . . . . .	57
RenderSystem . . . . .	62
UpdateSystem . . . . .	81
labelComponent . . . . .	58

OffsetComponent . . . . .	59
OnClickComponent . . . . .	59
PlayerComponent . . . . .	60
PlayerMissileComponent . . . . .	60
PositionComponent . . . . .	61
Rtype . . . . .	63
Scenes . . . . .	66
ScoreComponent . . . . .	74
SpriteComponent . . . . .	77
SpriteDataComponent . . . . .	78
TextComponent . . . . .	79
TextureManager . . . . .	80
Vector< T > . . . . .	83
Vector< float > . . . . .	83
Vector< uint32_t > . . . . .	83
VelocityComponent . . . . .	84
vf2d . . . . .	84
WeaponComponent . . . . .	85

## Chapter 3

# Class Index

### 3.1 Class List

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

<a href="#">r_type::net::AClient&lt; T &gt;</a>	11
<a href="#">AllyComponent</a>	16
<a href="#">AllyMissileComponent</a>	16
<a href="#">r_type::net::AServer&lt; T &gt;</a>	
AServer class	16
<a href="#">BackgroundComponent</a>	27
<a href="#">BasicMonsterComponent</a>	27
<a href="#">BindComponent</a>	27
<a href="#">r_type::net::Client</a>	28
<a href="#">ComponentManager</a>	
Manages the components of entities in an ECS system	30
<a href="#">componentNotFound</a>	
Exception class for when a component is not found	32
<a href="#">CreatableClientObject</a>	
Enum class for the creatable client object	33
<a href="#">EnemyComponent</a>	33
<a href="#">EnemyMissileComponent</a>	34
<a href="#">Entity</a>	
Represents an entity in the ECS system	34
<a href="#">EntityFactory</a>	
A class responsible for creating different types of entities	35
<a href="#">EntityInformation</a>	
Represents information about an entity	41
<a href="#">EntityManager</a>	
Class responsible for managing entities in the ECS system	42
<a href="#">entityNotFound</a>	
Exception class for entity not found error	45
<a href="#">failedToLoadTexture</a>	
Exception class for failed texture loading	45
<a href="#">HealthComponent</a>	46
<a href="#">HitboxComponent</a>	47
<a href="#">r_type::net::IClient&lt; T &gt;</a>	47
<a href="#">IEntity</a>	
Entity in the system	50
<a href="#">IEntityFactory</a>	
The interface for an entity factory	51

<a href="#">InputComponent</a>	57
<a href="#">ISystem</a>	57
<a href="#">labelComponent</a>	58
<a href="#">OffsetComponent</a>	59
<a href="#">OnClickComponent</a>	59
<a href="#">PlayerComponent</a>	60
<a href="#">PlayerMissileComponent</a>	60
<a href="#">PositionComponent</a>	61
<a href="#">RenderSystem</a>	
A system responsible for rendering components	62
<a href="#">Rtype</a>	63
<a href="#">Scenes</a>	
Represents a class that manages different scenes in a game	66
<a href="#">ScoreComponent</a>	74
<a href="#">r_type::net::Server</a>	75
<a href="#">SpriteComponent</a>	77
<a href="#">SpriteDataComponent</a>	78
<a href="#">TextComponent</a>	79
<a href="#">TextureManager</a>	80
<a href="#">UpdateSystem</a>	
A system responsible for updating entities in the game	81
<a href="#">Vector&lt; T &gt;</a>	83
<a href="#">VelocityComponent</a>	84
<a href="#">vf2d</a>	
Represents a 2D vector with x and y coordinates	84
<a href="#">WeaponComponent</a>	85

## Chapter 4

# File Index

### 4.1 File List

Here is a list of all files with brief descriptions:

/home/runner/work/R-Type/R-Type/Client/Interface/Include/mainmenu.hpp	87
/home/runner/work/R-Type/R-Type/Client/Interface/Include/r_type_client.hpp	89
/home/runner/work/R-Type/R-Type/Client/Interface/Include/Net/a_client.hpp	87
/home/runner/work/R-Type/R-Type/Client/Interface/Include/Net/client.hpp	88
/home/runner/work/R-Type/R-Type/Client/Interface/Include/Net/i_client.hpp	88
/home/runner/work/R-Type/R-Type/Client/Src/main.cpp	89
/home/runner/work/R-Type/R-Type/Client/Src/r_type_client.cpp	90
/home/runner/work/R-Type/R-Type/Client/Src/scenes.cpp	90
/home/runner/work/R-Type/R-Type/ECS/Interface/Include/creatable_client_object.hpp	97
/home/runner/work/R-Type/R-Type/ECS/Interface/Include/entity_struct.hpp	99
/home/runner/work/R-Type/R-Type/ECS/Interface/Include/error_handling.hpp	99
/home/runner/work/R-Type/R-Type/ECS/Interface/Include/scenes.hpp	100
/home/runner/work/R-Type/R-Type/ECS/Interface/Include/sprite_path.hpp	100
/home/runner/work/R-Type/R-Type/ECS/Interface/Include/texture_manager.hpp	102
/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/ally_component.hpp	92
/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/ally_missile_component.hpp	92
/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/background_component.hpp	92
/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/basic_monster_component.hpp	92
/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/bind_component.hpp	92
/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/component_manager.hpp	93
/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/components.hpp	93
/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/enemy_component.hpp	94
/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/enemy_missile_component.hpp	94
/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/health_component.hpp	94
/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/hitbox_component.hpp	94
/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/input_component.hpp	94
/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/label_component.hpp	95
/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/offset_component.hpp	95
/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/on_click_component.hpp	95
/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/player_component.hpp	95
/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/player_missile_component.hpp	96
/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/position_component.hpp	96
/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/score_component.hpp	96
/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/sprite_component.hpp	96
/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/sprite_data_component.hpp	96

/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/text_component.hpp . . . . .	97
/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/velocity_component.hpp . . . . .	97
/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/weapon_component.hpp . . . . .	97
/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Entities/entity.hpp . . . . .	98
/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Entities/entity_factory.hpp . . . . .	98
/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Entities/entity_manager.hpp . . . . .	98
/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Entities/i_entity.hpp . . . . .	99
/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Entities/i_entity_factory.hpp . . . . .	99
/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Systems/button_system.hpp . . . . .	101
/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Systems/i_system.hpp . . . . .	101
/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Systems/render_system.hpp . . . . .	102
/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Systems/systems.hpp . . . . .	102
/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Systems/update_system.hpp . . . . .	102
/home/runner/work/R-Type/R-Type/ECS/Src/sprite_path.cpp . . . . .	103
/home/runner/work/R-Type/R-Type/ECS/Src/Entities/entity_factory.cpp . . . . .	103
/home/runner/work/R-Type/R-Type/ECS/Src/Systems/render_system.cpp . . . . .	104
/home/runner/work/R-Type/R-Type/ECS/Src/Systems/update_system.cpp . . . . .	104
/home/runner/work/R-Type/R-Type/Server/Interface/Include/r_type_server.hpp . . . . .	105
/home/runner/work/R-Type/R-Type/Server/Interface/Include/Net/a_server.hpp . . . . .	104
/home/runner/work/R-Type/R-Type/Server/Interface/Include/Net/server.hpp . . . . .	104
/home/runner/work/R-Type/R-Type/Server/Src/main.cpp . . . . .	89
/home/runner/work/R-Type/R-Type/Server/Src/server.cpp . . . . .	105

## Chapter 5

# Namespace Documentation

### 5.1 `r_type` Namespace Reference

#### Namespaces

- [net](#)

### 5.2 `r_type::net` Namespace Reference

#### Classes

- class [AClient](#)
- class [Client](#)
- class [IClient](#)
- class [AServer](#)
  - [AServer](#) *class.*
- class [Server](#)





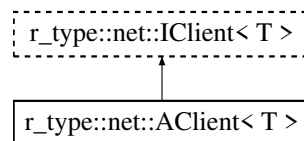
## Chapter 6

# Class Documentation

### 6.1 r\_type::net::AClient< T > Class Template Reference

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

Inheritance diagram for r\_type::net::AClient< T >:



#### Public Member Functions

- [AClient](#) ()
- virtual [~AClient](#) ()
- bool [Connect](#) (const std::string &host, const uint16\_t port)  
*Connects to a remote host using UDP protocol.*
- void [Disconnect](#) ()  
*Disconnects the client from the server.*
- bool [IsConnected](#) ()  
*Checks if the client is connected to the server.*
- void [Send](#) (const Message< T > &msg)  
*Send message to server.*
- ThreadSafeQueue< OwnedMessage< T > > & [Incoming](#) ()  
*get incoming messages*
- const std::unique\_ptr< Connection< T > > & [getConnection](#) ()
- void [setPlayerId](#) (int id)
- uint32\_t [getPlayerId](#) ()
- void [addEntity](#) (EntityInformation entity, ComponentManager &componentManager, TextureManager &textureManager)
- void [removeEntity](#) (int entityId, ComponentManager &componentManager)
- void [updateEntity](#) (EntityInformation entity, ComponentManager &componentManager)

## Protected Attributes

- asio::io\_context [m\\_context](#)
- std::thread [thrContext](#)
- std::unique\_ptr< Connection< T > > [m\\_connection](#)

## Private Attributes

- ThreadSafeQueue< OwnedMessage< T > > [m\\_qMessagesIn](#)
- uint32\_t [playerId](#) = 0

## 6.1.1 Constructor & Destructor Documentation

### 6.1.1.1 AClient()

```
template<typename T >
r\_type::net::AClient< T >::AClient ( ) [inline]
```

### 6.1.1.2 ~AClient()

```
template<typename T >
virtual r\_type::net::AClient< T >::~~AClient ( ) [inline], [virtual]
```

## 6.1.2 Member Function Documentation

### 6.1.2.1 addEntity()

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

### 6.1.2.2 Connect()

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

Connects to a remote host using UDP protocol.

## Parameters

<i>host</i>	The IP address or hostname of the remote host.
<i>port</i>	The port number of the remote host.

## Returns

true if the connection is successful, false otherwise.

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

## 6.1.2.3 Disconnect()

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

Disconnects the client from the server.

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

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

## 6.1.2.4 getConnection()

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

## 6.1.2.5 getPlayerId()

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

## 6.1.2.6 Incoming()

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

get incoming messages

## Returns

`ThreadSafeQueue<OwnedMessage<T>>&`

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

### 6.1.2.7 isConnected()

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

Checks if the client is connected to the server.

#### Returns

true  
false

Implements [r\\_type::net::IClient< T >](#).

### 6.1.2.8 removeEntity()

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

### 6.1.2.9 Send()

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

Send message to server.

#### Parameters

<i>msg</i>	
------------	--

Implements [r\\_type::net::IClient< T >](#).

### 6.1.2.10 setPlayerId()

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

#### 6.1.2.11 `updateEntity()`

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

### 6.1.3 Member Data Documentation

#### 6.1.3.1 `m_connection`

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

#### 6.1.3.2 `m_context`

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

#### 6.1.3.3 `m_qMessagesIn`

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

#### 6.1.3.4 `playerId`

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

#### 6.1.3.5 `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`

## 6.2 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](#)

## 6.3 AllyMissileComponent Struct Reference

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

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

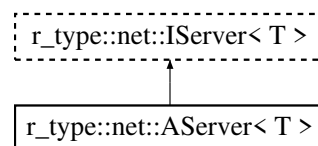
- [/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/ally\\_missile\\_component.hpp](#)

## 6.4 r\_type::net::AServer< T > Class Template Reference

[AServer](#) class.

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

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



### Public Member Functions

- [AServer](#) (uint16\_t port)  
*Construct a new [Server](#) Interface object.*
- [~AServer](#) ()  
*Destroy the [Server](#) Interface object.*
- 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)  
*send message message to client*
- void [MessageAllClients](#) (const Message< T > &msg, std::shared\_ptr< Connection< T >> plgnore← Client=nullptr)  
*message all clients*

- void [Update](#) (size\_t nMaxMessages=-1, bool bWait=false)  
*update server*
- void [UpdateEntityPosition](#) (r\_type::net::Message< T > &msg, uint32\_t clientId)  
*Updates the position of an entity based on the message received and the client ID.*
- uint32\_t [GetClientEntityId](#) (uint32\_t id)  
*Retrieves the entity ID associated with a client ID.*
- void [RemovePlayer](#) (uint32\_t id)  
*Removes a player from the game based on the client ID.*
- void [RemoveEntities](#) (uint32\_t id)  
*Removes entities associated with a player.*
- [EntityInformation InitiatePlayers](#) (int clientId)  
*Initializes a new player entity and assigns a random position.*
- [EntityInformation InitiateMissile](#) (int clientId)  
*Initializes a missile entity associated with a player.*
- [EntityInformation InitiateBackground](#) ()  
*Initializes a background entity.*
- void [InitListEntities](#) (std::shared\_ptr< r\_type::net::Connection< T >> client, u\_int32\_t entityId)  
*Sends a list of existing entities to a newly connected client for initialization.*
- int [CheckPlayerPosition](#) ([EntityInformation](#) desc)  
*check player position to avoid collision*
- virtual void [OnClientValidated](#) (std::shared\_ptr< Connection< T >> client)

## Public Attributes

- ThreadSafeQueue< OwnedMessage< T > > [m\\_qMessagesIn](#)
- std::deque< std::shared\_ptr< Connection< T > > > [m\\_deqConnections](#)
- asio::io\_context [m\\_asioContext](#)
- std::thread [m\\_threadContext](#)
- asio::ip::udp::socket [m\\_asioSocket](#)
- asio::ip::udp::endpoint [m\\_clientEndpoint](#)
- std::array< uint8\_t, 1024 > [m\\_tempBuffer](#)
- uint32\_t [nIDCounter](#) = 10000
- [ComponentManager](#) [componentManager](#)
- [EntityManager](#) [entityManager](#)
- [EntityFactory](#) [entityFactory](#)
- std::unordered\_map< uint32\_t, uint32\_t > [clientPlayerID](#)  
*A container that maps client IDs to player IDs.*
- int [nbrOfPlayers](#) = 0
- std::chrono::system\_clock::time\_point [\\_clock](#) = std::chrono::system\_clock::now()
- [EntityInformation](#) [background](#)

## 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*

### 6.4.1 Detailed Description

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

[AServer](#) class.

#### Parameters

<i>T</i>	
----------	--

### 6.4.2 Constructor & Destructor Documentation

#### 6.4.2.1 AServer()

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

Construct a new [Server](#) Interface object.

#### Parameters

<i>port</i>	
-------------	--

#### 6.4.2.2 ~AServer()

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

Destroy the [Server](#) Interface object.

### 6.4.3 Member Function Documentation

#### 6.4.3.1 CheckPlayerPosition()

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

check player position to avoid collision



## Parameters

<i>desc</i>	
-------------	--

## Returns

true

false

**6.4.3.2 GetClientEntityId()**

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

Retrieves the entity ID associated with a client ID.

## Parameters

<i>id</i>	The client ID.
-----------	----------------

## Returns

uint32\_t The entity ID associated with the client.

**6.4.3.3 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.

**6.4.3.4 InitiateMissile()**

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

Initializes a missile entity associated with a player.

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

**Parameters**

<i>clientId</i>	The client ID of the player firing the missile.
-----------------	---

**Returns**

[EntityInformation](#) The information of the newly created missile entity.

**6.4.3.5 InitiatePlayers()**

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

Initializes a new player entity and assigns a random position.

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

**Parameters**

<i>clientId</i>	The client ID of the player being initialized.
-----------------	--

**Returns**

[EntityInformation](#) The information of the newly created player entity.

**6.4.3.6 InitListEntities()**

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

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

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

**Parameters**

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

### 6.4.3.7 `MessageAllClients()`

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

message all clients

#### Parameters

<i>msg</i>	
<i>pIgnoreClient</i>	

### 6.4.3.8 `MessageClient()`

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

send message message to client

#### Parameters

<i>client</i>	
<i>msg</i>	

### 6.4.3.9 `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

<i>client</i>	
---------------	--

**Returns**

true  
false

**6.4.3.10 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**

<i>client</i>	
---------------	--

**6.4.3.11 OnClientValidated()**

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

**6.4.3.12 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**

<i>client</i>	
<i>msg</i>	

**6.4.3.13 RemoveEntities()**

```
template<typename T >
```

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

Removes entities associated with a player.

#### Parameters

<i>id</i>	The ID of the player whose entities are to be removed.
-----------	--

#### 6.4.3.14 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

<i>id</i>	The client ID of the player to be removed.
-----------	--

#### 6.4.3.15 Start()

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

Start the server.

#### Returns

true

false

#### 6.4.3.16 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.

#### 6.4.3.17 Update()

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

update server

##### Parameters

<i>nMaxMessages</i>	
<i>bWait</i>	

#### 6.4.3.18 UpdateEntityPosition()

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

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

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

##### Parameters

<i>msg</i>	The message containing the new position of the entity.
<i>clientId</i>	The ID of the client sending the update.

#### 6.4.3.19 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..

## 6.4.4 Member Data Documentation

### 6.4.4.1 `_clock`

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

### 6.4.4.2 `background`

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

### 6.4.4.3 `clientPlayerID`

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

A container that maps client IDs to player IDs.

left: client ID right: player ID

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

### 6.4.4.4 `componentManager`

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

### 6.4.4.5 `entityFactory`

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

#### 6.4.4.6 entityManager

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

#### 6.4.4.7 m\_asioContext

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

#### 6.4.4.8 m\_asioSocket

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

#### 6.4.4.9 m\_clientEndpoint

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

#### 6.4.4.10 m\_deqConnections

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

#### 6.4.4.11 m\_qMessagesIn

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

#### 6.4.4.12 m\_tempBuffer

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



#### 6.4.4.13 m\_threadContext

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

#### 6.4.4.14 nbrOfPlayers

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

#### 6.4.4.15 nIDCounter

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

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

- [/home/runner/work/R-Type/R-Type/Server/Interface/Include/Net/a\\_server.hpp](#)

## 6.5 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](#)

## 6.6 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](#)

## 6.7 BindComponent Struct Reference

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

## Public Member Functions

- [BindComponent](#) (std::function< [Scenes](#) \*([Scenes](#) \*, [Scenes::Actions](#))> bindFunction)

## Public Attributes

- bool [isHovered](#) = false
- std::function< [Scenes](#) \*([Scenes](#) \*, [Scenes::Actions](#))> [bind](#)

## 6.7.1 Constructor & Destructor Documentation

### 6.7.1.1 BindComponent()

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

## 6.7.2 Member Data Documentation

### 6.7.2.1 bind

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

### 6.7.2.2 isHovered

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

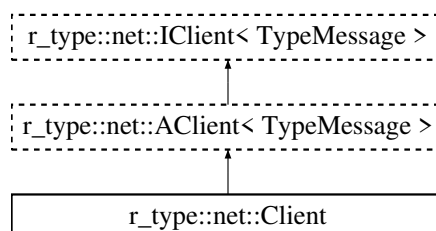
The documentation for this struct was generated from the following file:

- [/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/bind\\_component.hpp](#)

## 6.8 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 [addEntity](#) ([EntityInformation](#) entity, [ComponentManager](#) &componentManager, [TextureManager](#) &textureManager)
- void [removeEntity](#) (int entityId, [ComponentManager](#) &componentManager)
- void [updateEntity](#) ([EntityInformation](#) entity, [ComponentManager](#) &componentManager)

## Additional Inherited Members

### 6.8.1 Member Function Documentation

#### 6.8.1.1 addEntity()

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

#### 6.8.1.2 MessageAll()

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

Send a message to the server to all other clients.

#### 6.8.1.3 PingServer()

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

Send a message to the server to get the ping.

#### 6.8.1.4 removeEntity()

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

### 6.8.1.5 updateEntity()

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

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

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

## 6.9 ComponentManager Class Reference

Manages the components of entities in an ECS system.

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

### Public Member Functions

- template<typename ComponentType , typename... Args>  
void [addComponent](#) (int entityId, Args &&...args)  
*Adds a component to an entity.*
- template<typename ComponentType >  
std::optional< ComponentType \* > [getComponent](#) (int entityId)  
*Retrieves the component of the specified type associated with the given entity ID.*
- template<typename ComponentType >  
std::optional< std::unordered\_map< int, std::any > \* > [getComponentMap](#) ()  
*Retrieves the component map for the specified component type.*
- template<typename ComponentType >  
void [removeEntityFromComponent](#) (int entityId)

### Private Attributes

- std::unordered\_map< std::type\_index, std::unordered\_map< int, std::any > > [components](#)  
*A component manager that stores components in an unordered map.*

### 6.9.1 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.

### 6.9.2 Member Function Documentation

#### 6.9.2.1 addComponent()

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

Adds a component to an entity.

## Template Parameters

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

## Parameters

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

## 6.9.2.2 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

<i>ComponentType</i>	The type of the component to retrieve.
----------------------	--

## Parameters

<i>entityId</i>	The ID of the entity.
-----------------	-----------------------

## Returns

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

## 6.9.2.3 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

<i>ComponentType</i>	The type of the component.
----------------------	----------------------------

**Returns**

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

**6.9.2.4 removeEntityFromComponent()**

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

**6.9.3 Member Data Documentation****6.9.3.1 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:

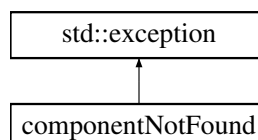
- [/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/component\\_manager.hpp](/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/component_manager.hpp)

**6.10 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

### 6.10.1 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.

### 6.10.2 Member Function Documentation

#### 6.10.2.1 `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`

## 6.11 CreatableClientObject Class Reference

Enum class for the creatable client object.

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

### 6.11.1 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`

## 6.12 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`

## 6.13 EnemyMissileComponent Struct Reference

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

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

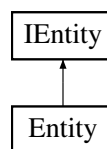
- [/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/enemy\\_missile\\_component.hpp](#)

## 6.14 Entity Class Reference

Represents an entity in the ECS system.

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

Inheritance diagram for Entity:



### Public Member Functions

- [Entity](#) (int id)  
*Constructs an [Entity](#) object with the given ID.*
- int [getId](#) () const override  
*Returns the ID of the entity.*

### Private Attributes

- int [\\_id](#)

### 6.14.1 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.

### 6.14.2 Constructor & Destructor Documentation

#### 6.14.2.1 Entity()

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

Constructs an [Entity](#) object with the given ID.



## Parameters

<i>id</i>	The ID of the entity.
-----------	-----------------------

### 6.14.3 Member Function Documentation

#### 6.14.3.1 getId()

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

Returns the ID of the entity.

## Returns

The ID of the entity.

Implements [IEntity](#).

### 6.14.4 Member Data Documentation

#### 6.14.4.1 \_id

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

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

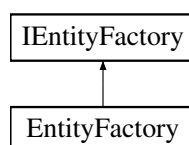
- [/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Entities/entity.hpp](#)

## 6.15 EntityFactory Class Reference

A class responsible for creating different types of entities.

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

Inheritance diagram for EntityFactory:



## Public Member Functions

- [Entity createBackground](#) ([EntityManager](#) &entityManager, [ComponentManager](#) &componentManager) override  
*Creates a background entity.*
- [Entity createPlayer](#) ([EntityManager](#) &entityManager, [ComponentManager](#) &componentManager) override  
*Creates a player entity.*
- [Entity createAlly](#) ([EntityManager](#) &entityManager, [ComponentManager](#) &componentManager) override  
*Creates a player entity.*
- [Entity createBasicEnemy](#) ([EntityManager](#) &entityManager, [ComponentManager](#) &componentManager) override  
*Creates a basic enemy entity.*
- [Entity createBasicMonster](#) ([EntityManager](#) &entityManager, [ComponentManager](#) &componentManager) override  
*Creates a basic monster entity.*
- [Entity createPlayerMissile](#) ([EntityManager](#) &entityManager, [ComponentManager](#) &componentManager) override  
*Creates a player missile entity.*
- [Entity createButton](#) ([EntityManager](#) &entityManager, [ComponentManager](#) &componentManager, [TextureManager](#) &textureManager, std::string text, std::function< [Scenes](#) \*([Scenes](#) \*)> \*onClick)  
*Creates a button entity.*
- [Entity createSmallButton](#) ([EntityManager](#) &entityManager, [ComponentManager](#) &componentManager, [TextureManager](#) &textureManager, std::string text, std::function< [Scenes](#) \*([Scenes](#) \*, [Scenes::Actions](#))> \*onClick)  
*Creates a small button entity.*
- [Entity createAllyMissile](#) ([EntityManager](#) &entityManager, [ComponentManager](#) &componentManager) override  
*Creates an ally missile entity.*
- [Entity createEnemyMissile](#) ([EntityManager](#) &entityManager, [ComponentManager](#) &componentManager) override  
*Creates an enemy missile entity.*

### 6.15.1 Detailed Description

A class responsible for creating different types of entities.

### 6.15.2 Member Function Documentation

#### 6.15.2.1 createAlly()

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

Creates a player entity.

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

## Parameters

<i>entityManager</i>	The entity manager to use for creating the entity.
<i>componentManager</i>	The component manager to use for adding components to the entity.

## Returns

The created player entity.

Implements [IEntityFactory](#).

**6.15.2.2 createAllyMissile()**

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

Creates an ally missile entity.

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

## Parameters

<i>entityManager</i>	The entity manager used to create the entity.
<i>componentManager</i>	The component manager used to manage the components of the entity.

## Returns

The created ally missile entity.

Implements [IEntityFactory](#).

**6.15.2.3 createBackground()**

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

Creates a background entity.

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

## Parameters

<i>entityManager</i>	The entity manager to use for creating the entity.
<i>componentManager</i>	The component manager to use for adding components to the entity.

**Returns**

The created background entity.

Implements [IEntityFactory](#).

**6.15.2.4 createBasicEnemy()**

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

Creates a basic enemy entity.

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

**Parameters**

<i>entityManager</i>	The entity manager used to create the entity.
<i>componentManager</i>	The component manager used to add components to the entity.

**Returns**

The created basic enemy entity.

Implements [IEntityFactory](#).

**6.15.2.5 createBasicMonster()**

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

Creates a basic monster entity.

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

**Parameters**

<i>entityManager</i>	The entity manager used to create the entity.
<i>componentManager</i>	The component manager used to add components to the entity.

**Returns**

The created basic monster entity.

Implements [IEntityFactory](#).

#### 6.15.2.6 createButton()

```
Entity EntityFactory::createButton (
    EntityManager & entityManager,
    ComponentManager & componentManager,
    TextureManager & textureManager,
    std::string text,
    std::function< Scenes *(Scenes *)> * onClick ) [virtual]
```

Creates a button entity.

This function creates a button entity with the specified parameters.

##### Parameters

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

##### Returns

The created button entity.

Implements [IEntityFactory](#).

#### 6.15.2.7 createEnemyMissile()

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

Creates an enemy missile entity.

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

##### Parameters

<i>entityManager</i>	The entity manager used to create the entity.
<i>componentManager</i>	The component manager used to add components to the entity.

**Returns**

The created enemy missile entity.

Implements [IEntityFactory](#).

**6.15.2.8 createPlayer()**

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

Creates a player entity.

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

**Parameters**

<i>entityManager</i>	The entity manager to use for creating the entity.
<i>componentManager</i>	The component manager to use for adding components to the entity.

**Returns**

The created player entity.

Implements [IEntityFactory](#).

**6.15.2.9 createPlayerMissile()**

```
Entity EntityFactory::createPlayerMissile (
    EntityManager & entityManager,
    ComponentManager & componentManager ) [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**

<i>playerId</i>	The ID of the player.
<i>entityManager</i>	The entity manager to add the player missile entity to.
<i>componentManager</i>	The component manager to initialize the components for the player missile entity.

**Returns**

The created player missile entity.

Implements [IEntityFactory](#).

**6.15.2.10 createSmallButton()**

```
Entity IEntityFactory::createSmallButton (
    EntityManager & entityManager,
    ComponentManager & componentManager,
    TextureManager & textureManager,
    std::string text,
    std::function< Scenes *(Scenes *, Scenes::Actions)> * onClick )
```

Creates a small button entity.

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

**Parameters**

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

**Returns**

The created small button entity.

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

- /home/runner/work/R-Type/R-Type/ECS/Interface/Include/Entities/[entity\\_factory.hpp](#)
- /home/runner/work/R-Type/R-Type/ECS/Src/Entities/[entity\\_factory.cpp](#)

## 6.16 EntityInformation Struct Reference

Represents information about an entity.

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

**Public Attributes**

- uint32\_t [uniqueID](#) = 0
- [SpriteDataComponent](#) [spriteData](#)
- [vf2d](#) [vPos](#)

### 6.16.1 Detailed Description

Represents information about an entity.

### 6.16.2 Member Data Documentation

#### 6.16.2.1 spriteData

```
SpriteDataComponent EntityInformation::spriteData
```

#### 6.16.2.2 uniqueID

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

#### 6.16.2.3 vPos

```
vf2d EntityInformation::vPos
```

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

- [/home/runner/work/R-Type/R-Type/ECS/Interface/Include/entity\\_struct.hpp](#)

## 6.17 EntityManager Class Reference

Class responsible for managing entities in the ECS system.

```
#include <entity_manager.hpp>
```

### Public Member Functions

- [Entity createEntity \(\)](#)  
*Create a [Entity](#) object.*
- void [removeEntity](#) (int entityId)  
*Remove an entity from the entity manager.*
- [Entity & getEntity](#) (int entityId)  
*Get an entity by its ID.*
- const std::vector< [Entity](#) > & [getAllEntities](#) () const  
*Get all entities in the entity manager.*



## Private Attributes

- int `entityNb` = 0  
*The number of entities in the entity manager.*
- std::vector< `Entity` > `entities`

### 6.17.1 Detailed Description

Class responsible for managing entities in the ECS system.

### 6.17.2 Member Function Documentation

#### 6.17.2.1 createEntity()

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

Create a `Entity` object.

Returns

`Entity`

#### 6.17.2.2 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.

#### 6.17.2.3 getEntity()

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

Get an entity by its ID.

## Parameters

<i>entity</i> ↔ <i>Id</i>	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.

**6.17.2.4 removeEntity()**

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

Remove an entity from the entity manager.

## Parameters

<i>entity</i> ↔ <i>Id</i>	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.

**6.17.3 Member Data Documentation****6.17.3.1 entities**

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

**6.17.3.2 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:

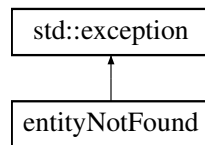
- [/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Entities/entity\\_manager.hpp](#)

## 6.18 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

#### 6.18.1 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.

#### 6.18.2 Member Function Documentation

##### 6.18.2.1 what()

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

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

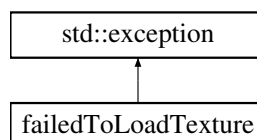
- `/home/runner/work/R-Type/R-Type/ECS/Interface/Include/error\_handling.hpp`

## 6.19 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

### 6.19.1 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.

### 6.19.2 Member Function Documentation

#### 6.19.2.1 `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`

## 6.20 HealthComponent Struct Reference

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

### Public Attributes

- `int max\_health`
- `int health`

### 6.20.1 Member Data Documentation

#### 6.20.1.1 `health`

```
int HealthComponent::health
```

### 6.20.1.2 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](/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/health_component.hpp)

## 6.21 HitboxComponent Struct Reference

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

### Public Attributes

- int [w](#)
- int [h](#)

### 6.21.1 Member Data Documentation

#### 6.21.1.1 h

```
int HitboxComponent::h
```

#### 6.21.1.2 w

```
int HitboxComponent::w
```

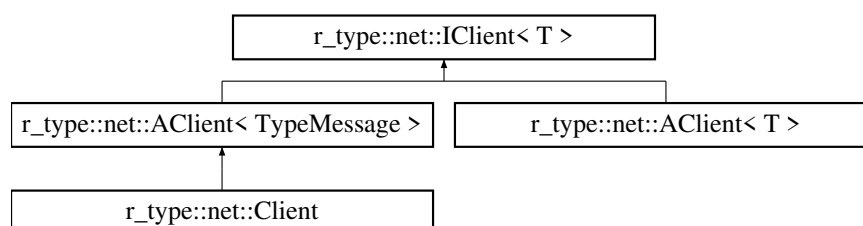
The documentation for this struct was generated from the following file:

- [/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/hitbox\\_component.hpp](/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/hitbox_component.hpp)

## 6.22 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*

## 6.22.1 Constructor & Destructor Documentation

### 6.22.1.1 IClient()

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

### 6.22.1.2 ~IClient()

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

## 6.22.2 Member Function Documentation

### 6.22.2.1 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

<i>host</i>	The IP address or hostname of the remote host.
<i>port</i>	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 >`.

**6.22.2.2 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 >`.

**6.22.2.3 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 >`.

**6.22.2.4 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 >`.

**6.22.2.5 Send()**

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

Send message to server.

#### Parameters

<code>msg</code>	
------------------	--

Implemented in [r\\_type::net::AClient< T >](#).

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

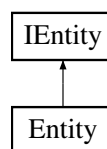
- [/home/runner/work/R-Type/R-Type/Client/Interface/Include/Net/i\\_client.hpp](#)

## 6.23 IEntity Class Reference

The [IEntity](#) class represents an entity in the system.

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

Inheritance diagram for IEntity:



### Public Member Functions

- virtual [~IEntity](#) ()=default  
*Destructor for the [IEntity](#) class.*
- virtual int [getId](#) () const =0  
*Gets the ID of the entity.*

#### 6.23.1 Detailed Description

The [IEntity](#) class represents an entity in the system.

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

#### Note

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

#### 6.23.2 Constructor & Destructor Documentation



### 6.23.2.1 ~IEntity()

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

Destructor for the [IEntity](#) class.

## 6.23.3 Member Function Documentation

### 6.23.3.1 getId()

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

Gets the ID of the entity.

#### Returns

The ID of the entity.

Implemented in [Entity](#).

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

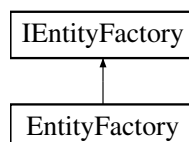
- [/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Entities/i\\_entity.hpp](#)

## 6.24 IEntityFactory Class Reference

The interface for an entity factory.

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

Inheritance diagram for IEntityFactory:



## Public Member Functions

- virtual [~IEntityFactory](#) ()=default  
*Destroy the [IEntityFactory](#) object.*
- virtual [Entity](#) [createBackground](#) ([EntityManager](#) &entityManager, [ComponentManager](#) &componentManager)=0  
*Creates a background entity.*
- virtual [Entity](#) [createPlayer](#) ([EntityManager](#) &entityManager, [ComponentManager](#) &componentManager)=0  
*Creates a player entity.*
- virtual [Entity](#) [createAlly](#) ([EntityManager](#) &entityManager, [ComponentManager](#) &componentManager)=0  
*Creates an ally entity.*
- virtual [Entity](#) [createBasicEnemy](#) ([EntityManager](#) &entityManager, [ComponentManager](#) &componentManager)=0  
*Creates a basic enemy entity.*
- virtual [Entity](#) [createBasicMonster](#) ([EntityManager](#) &entityManager, [ComponentManager](#) &componentManager)=0  
*Creates a basic monster entity.*
- virtual [Entity](#) [createPlayerMissile](#) ([EntityManager](#) &entityManager, [ComponentManager](#) &componentManager)=0  
*Creates a player missile entity.*
- virtual [Entity](#) [createAllyMissile](#) ([EntityManager](#) &entityManager, [ComponentManager](#) &componentManager)=0  
*Creates an ally missile entity.*
- virtual [Entity](#) [createEnemyMissile](#) ([EntityManager](#) &entityManager, [ComponentManager](#) &componentManager)=0  
*Creates an enemy missile entity.*
- virtual [Entity](#) [createButton](#) ([EntityManager](#) &entityManager, [ComponentManager](#) &componentManager, [TextureManager](#) &textureManager, std::string text, std::function< [Scenes](#) \*([Scenes](#) \*)> \*onClick)=0  
*Creates a button entity.*

### 6.24.1 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.

### 6.24.2 Constructor & Destructor Documentation

#### 6.24.2.1 ~IEntityFactory()

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

Destroy the [IEntityFactory](#) object.

### 6.24.3 Member Function Documentation

#### 6.24.3.1 createAlly()

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

Creates an ally entity.

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

##### Parameters

<i>entityManager</i>	The entity manager used to create the entity.
<i>componentManager</i>	The component manager used to manage the components of the entity.

##### Returns

The created ally entity.

Implemented in [EntityFactory](#).

#### 6.24.3.2 createAllyMissile()

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

Creates an ally missile entity.

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

##### Parameters

<i>entityManager</i>	The entity manager used to create the entity.
<i>componentManager</i>	The component manager used to manage the components of the entity.

##### Returns

The created ally missile entity.

Implemented in [EntityFactory](#).

### 6.24.3.3 createBackground()

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

Creates a background entity.

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

#### Parameters

<i>entityManager</i>	The entity manager to use for creating the entity.
<i>componentManager</i>	The component manager to use for adding components to the entity.

#### Returns

The created background entity.

Implemented in [EntityFactory](#).

### 6.24.3.4 createBasicEnemy()

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

Creates a basic enemy entity.

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

#### Parameters

<i>entityManager</i>	The entity manager used to create the entity.
<i>componentManager</i>	The component manager used to add components to the entity.

#### Returns

The created basic enemy entity.

Implemented in [EntityFactory](#).

### 6.24.3.5 createBasicMonster()

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

Creates a basic monster entity.

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

#### Parameters

<i>entityManager</i>	The entity manager used to create the entity.
<i>componentManager</i>	The component manager used to add components to the entity.

#### Returns

The created basic monster entity.

Implemented in [EntityFactory](#).

#### 6.24.3.6 createButton()

```
virtual Entity IEntityFactory::createButton (
    EntityManager & entityManager,
    ComponentManager & componentManager,
    TextureManager & textureManager,
    std::string text,
    std::function< Scenes *(Scenes *)> * onClick ) [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

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

#### Returns

The created button entity.

Implemented in [EntityFactory](#).

#### 6.24.3.7 createEnemyMissile()

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

Creates an enemy missile entity.

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

#### Parameters

<i>entityManager</i>	The entity manager used to create the entity.
<i>componentManager</i>	The component manager used to add components to the entity.

#### Returns

The created enemy missile entity.

Implemented in [EntityFactory](#).

### 6.24.3.8 createPlayer()

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

Creates a player entity.

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

#### Parameters

<i>entityManager</i>	The entity manager used to create the entity.
<i>componentManager</i>	The component manager used to add components to the entity.

#### Returns

The created player entity.

Implemented in [EntityFactory](#).

### 6.24.3.9 createPlayerMissile()

```
virtual Entity IEntityFactory::createPlayerMissile (
    EntityManager & entityManager,
    ComponentManager & componentManager ) [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

<i>playerId</i>	The ID of the player.
<i>entityManager</i>	The entity manager to add the player missile entity to.
<i>componentManager</i>	The component manager to initialize the components for the player missile entity.

## Returns

The created player missile entity.

Implemented in [EntityFactory](#).

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

- [/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Entities/i\\_entity\\_factory.hpp](#)

## 6.25 InputComponent Struct Reference

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

## Public Attributes

- [InputType input](#)

### 6.25.1 Member Data Documentation

#### 6.25.1.1 input

[InputType](#) InputComponent::input

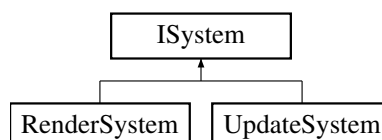
The documentation for this struct was generated from the following file:

- [/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/input\\_component.hpp](#)

## 6.26 ISystem Class Reference

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

Inheritance diagram for ISystem:



## Public Member Functions

- [ISystem](#) ()=default

### 6.26.1 Constructor & Destructor Documentation

#### 6.26.1.1 ISystem()

```
ISystem::ISystem ( ) [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](#)

## 6.27 labelComponent Struct Reference

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

### Public Attributes

- `std::string` [name](#)
- `int` [x](#)
- `int` [y](#)

### 6.27.1 Member Data Documentation

#### 6.27.1.1 name

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

#### 6.27.1.2 x

```
int labelComponent::x
```



### 6.27.1.3 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](/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/label_component.hpp)

## 6.28 OffsetComponent Struct Reference

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

### Public Attributes

- float [offset](#)

### 6.28.1 Member Data Documentation

#### 6.28.1.1 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](/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/offset_component.hpp)

## 6.29 OnClickComponent Struct Reference

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

### Public Member Functions

- [OnClickComponent](#) (std::function< [Scenes](#) \*([Scenes](#) \*)> &onClickfunction)

### Public Attributes

- bool [isClicked](#) = false
- std::function< [Scenes](#) \*([Scenes](#) \*)> & [onClick](#)

## 6.29.1 Constructor & Destructor Documentation

### 6.29.1.1 OnClickComponent()

```
OnClickComponent::OnClickComponent (
    std::function< Scenes *(Scenes *)> & onClickfunction ) [inline]
```

## 6.29.2 Member Data Documentation

### 6.29.2.1 isClicked

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

### 6.29.2.2 onClick

```
std::function<Scenes *(Scenes *)>& 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](#)

## 6.30 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](#)

## 6.31 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](#)

## 6.32 PositionComponent Struct Reference

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

### Public Member Functions

- [PositionComponent](#) (float [x](#), float [y](#))

### Public Attributes

- float [x](#)
- float [y](#)

### 6.32.1 Constructor & Destructor Documentation

#### 6.32.1.1 PositionComponent()

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

### 6.32.2 Member Data Documentation

#### 6.32.2.1 [x](#)

```
float PositionComponent::x
```

#### 6.32.2.2 [y](#)

```
float PositionComponent::y
```

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

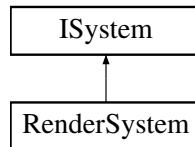
- [/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/position\\_component.hpp](#)

## 6.33 RenderSystem Class Reference

A system responsible for rendering components.

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

Inheritance diagram for RenderSystem:



### Public Member Functions

- [RenderSystem](#) (sf::RenderWindow &window)  
*Construct a new Render System object.*
- void [render](#) ([ComponentManager](#) &componentManager)  
*Renders the components managed by the given [ComponentManager](#).*

### Private Attributes

- sf::RenderWindow & [\\_window](#)

#### 6.33.1 Detailed Description

A system responsible for rendering components.

#### 6.33.2 Constructor & Destructor Documentation

##### 6.33.2.1 RenderSystem()

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

Construct a new Render System object.

A system responsible for rendering entities on a given window.

##### Parameters

<i>window</i>	This system takes a reference to an sf::RenderWindow object and is responsible for rendering entities on that window. It provides functionality to render sprites, shapes, and other graphical components associated with entities in the game.
---------------	---

### 6.33.3 Member Function Documentation

#### 6.33.3.1 render()

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

Renders the components managed by the given [ComponentManager](#).

##### Parameters

<i>componentManager</i>	The <a href="#">ComponentManager</a> that manages the components to be rendered.
-------------------------	--

This function is responsible for rendering the components managed by the provided [ComponentManager](#). It takes the [ComponentManager](#) as a parameter and performs the necessary rendering operations.

### 6.33.4 Member Data Documentation

#### 6.33.4.1 \_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](#)

## 6.34 Rtype Class Reference

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

### Public Member Functions

- [Rtype](#) ()  
*Construct a new [Rtype](#) object This will init the player.*
- void [run](#) ()  
*If `_mainMenu` variable is true, call `mainMenu`.*
- void [mainMenu](#) ()  
*Open window.*
- void [gameLoop](#) ()  
*Open window.*

- void `handleEvents` ()  
*This is where I will handle the events for the window & player (key input, etc.).*
- void `processServerMessages` ()  
*This is where I will process the info from the server.*
- void `updateGame` ()  
*This is where I will update the time, position of sprites, etc.*
- void `renderGame` ()  
*This is where I will render the game.*

## Private Attributes

- `Scenes * _scenes`  
*Set the Game Mode object.*
- `sf::RenderWindow _window`

## 6.34.1 Constructor & Destructor Documentation

### 6.34.1.1 Rtype()

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

Construct a new `Rtype` object This will init the player.

Construct a new `Rtype:: Rtype` object.

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

## 6.34.2 Member Function Documentation

### 6.34.2.1 gameLoop()

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

Open window.

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

### 6.34.2.2 handleEvents()

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

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

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

### 6.34.2.3 mainMenu()

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

Open window.

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

### 6.34.2.4 processServerMessages()

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

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

### 6.34.2.5 renderGame()

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

This is where I will render the game.

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

### 6.34.2.6 run()

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

If `_mainMenu` variable is true, call `mainMenu`.

While `_mainMenu` is false, call `gameLoop`.

### 6.34.2.7 updateGame()

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

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

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

## 6.34.3 Member Data Documentation

### 6.34.3.1 \_scenes

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

Set the Game Mode object.

## Parameters

<code>mode</code>	
-------------------	--

6.34.3.2 `_window`

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

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

- [/home/runner/work/R-Type/R-Type/Client/Interface/Include/r\\_type\\_client.hpp](#)
- [/home/runner/work/R-Type/R-Type/Client/Src/r-type\\_client.cpp](#)

## 6.35 Scenes Class Reference

Represents a class that manages different scenes in a game.

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

## Public Types

- enum class [Scene](#) {  
    [MAIN\\_MENU](#) , [GAME\\_LOOP](#) , [SETTINGS\\_MENU](#) , [IN\\_GAME\\_MENU](#) ,  
    [EXIT](#) }
- enum class [GameMode](#) { [EASY](#) , [MEDIUM](#) , [HARD](#) }  
    *Enumeration representing different game modes.*
- enum class [DaltonismMode](#) { [NORMAL](#) , [TRITANOPIA](#) , [DEUTERANOPIA](#) , [PROTANOPIA](#) }  
    *Enum class representing different modes of Daltonism.*
- enum class [Actions](#) {  
    [UP](#) , [DOWN](#) , [LEFT](#) , [RIGHT](#) ,  
    [FIRE](#) , [PAUSE](#) , [QUIT](#) }

## Public Member Functions

- [Scenes](#) (sf::RenderWindow \*window)  
    *Construct a new [Scenes](#) object.*
- [~Scenes](#) ()=default  
    *Destroy the [Scenes](#) object.*
- void [mainMenu](#) ()  
    *displays the main menu, creates all the necessary entities*
- void [gameLoop](#) ()  
    *displays the main game loop, creates all the necessary entities*
- void [settingsMenu](#) ()  
    *displays the settings menu, creates all the necessary entities*
- void [inGameMenu](#) ()



- displays the in game menu, creates all the necessary entities*
- void [render](#) ()
  - display what must be displayed (main menu, game loop, settings menu, in game menu), creates all the components needed and manages them*
- void [setDaltonism](#) ([DaltonismMode](#) mode)
  - Set the Daltonism object.*
- void [setGameMode](#) ([GameMode](#) mode)
  - Set the Game Mode object.*
- void [setScene](#) ([Scene](#) scene)
  - Set the Scene object.*
- [Scene](#) [getPreviousScene](#) ()
  - Get the Previous Scene object.*
- bool [shouldQuit](#) ()
  - check if game should stop running*
- sf::RenderWindow \* [getRenderWindow](#) ()
  - Get the RenderWindow object.*

## Public Attributes

- std::map< [Actions](#), sf::Keyboard::Key > [keyBinds](#)

## Private Attributes

- [GameMode](#) [currentGameMode](#) = [GameMode::MEDIUM](#)
  - the current game mode*
- [DaltonismMode](#) [currentDaltonismMode](#) = [DaltonismMode::NORMAL](#)
  - the current daltonism mode*
- [Scene](#) [currentScene](#) = [Scene::MAIN\\_MENU](#)
  - the current scene*
- [Scene](#) [previousScene](#) = [Scene::MAIN\\_MENU](#)
- bool [displayDaltonismChoice](#) = false
- bool [displayGameModeChoice](#) = false
- bool [displayKeyBinds](#) = false
- sf::RenderWindow \* [\\_window](#)
  - the window*
- std::vector< [Entity](#) \* > [buttons](#)
- sf::Keyboard::Key [binding](#)

### 6.35.1 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.

### 6.35.2 Member Enumeration Documentation

#### 6.35.2.1 Actions

```
enum Scenes::Actions [strong]
```

**Enumerator**

UP	
DOWN	
LEFT	
RIGHT	
FIRE	
PAUSE	
QUIT	

**6.35.2.2 DaltonismMode**

```
enum Scenes::DaltonismMode [strong]
```

Enum class representing different modes of Daltonism.

**Enumerator**

NORMAL	
TRITANOPIA	
DEUTERANOPIA	
PROTANOPIA	

**6.35.2.3 GameMode**

```
enum Scenes::GameMode [strong]
```

Enumeration representing different game modes.

**Enumerator**

EASY	
MEDIUM	
HARD	

**6.35.2.4 Scene**

```
enum Scenes::Scene [strong]
```

**Enumerator**

MAIN_MENU	
-----------	--

## Enumerator

GAME_LOOP	
SETTINGS_MENU	
IN_GAME_MENU	
EXIT	

## 6.35.3 Constructor & Destructor Documentation

### 6.35.3.1 Scenes()

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

Construct a new [Scenes](#) object.

## Parameters

<i>window</i>	
---------------	--

### 6.35.3.2 ~Scenes()

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

Destroy the [Scenes](#) object.

## 6.35.4 Member Function Documentation

### 6.35.4.1 gameLoop()

```
void Scenes::gameLoop ( )
```

displays the main game loop, creates all the necessary entities

This function handles the main game loop for the [Scenes](#) class.

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

The game loop performs the following steps:

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

#### Note

This code assumes the presence of the [r\\_type::net::Client](#), [ComponentManager](#), [TextureManager](#), [EntityManager](#), [UpdateSystem](#), and [RenderSystem](#) classes.

#### See also

[r\\_type::net::Client](#)  
[ComponentManager](#)  
[TextureManager](#)  
[EntityManager](#)  
[UpdateSystem](#)  
[RenderSystem](#)

#### 6.35.4.2 `getPreviousScene()`

```
Scene Scenes::getPreviousScene ( ) [inline]
```

Get the Previous Scene object.

#### Returns

Scene

#### 6.35.4.3 `getRenderWindow()`

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

Get the RenderWindow object.

#### Returns

sf::RenderWindow\*

#### 6.35.4.4 inGameMenu()

```
void Scenes::inGameMenu ( )
```

displays the in game menu, creates all the necessary entities

Displays the in-game menu.

#### 6.35.4.5 mainMenu()

```
void Scenes::mainMenu ( )
```

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

#### 6.35.4.6 render()

```
void Scenes::render ( )
```

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.

#### 6.35.4.7 setDaltonism()

```
void Scenes::setDaltonism (
    DaltonismMode mode ) [inline]
```

Set the Daltonism object.

## Parameters

<i>mode</i>	The daltonism mode to set
-------------	---------------------------

**6.35.4.8 setGameMode()**

```
void Scenes::setGameMode (
    GameMode mode ) [inline]
```

Set the Game Mode object.

## Parameters

<i>mode</i>	
-------------	--

**6.35.4.9 setScene()**

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

Set the Scene object.

## Parameters

<i>scene</i>	
--------------	--

**6.35.4.10 settingsMenu()**

```
void Scenes::settingsMenu ( )
```

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.

**6.35.4.11 shouldQuit()**

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

check if game should stop running

## Returns

true

false

## 6.35.5 Member Data Documentation

### 6.35.5.1 `_window`

```
sf::RenderWindow* Scenes::_window [private]
```

the window

### 6.35.5.2 `binding`

```
sf::Keyboard::Key Scenes::binding [private]
```

### 6.35.5.3 `buttons`

```
std::vector<Entity*> Scenes::buttons [private]
```

### 6.35.5.4 `currentDaltonismMode`

```
DaltonismMode Scenes::currentDaltonismMode = DaltonismMode::NORMAL [private]
```

the current daltonism mode

### 6.35.5.5 `currentGameMode`

```
GameMode Scenes::currentGameMode = GameMode::MEDIUM [private]
```

the current game mode

### 6.35.5.6 `currentScene`

```
Scene Scenes::currentScene = Scene::MAIN_MENU [private]
```

the current scene

#### 6.35.5.7 displayDaltonismChoice

```
bool Scenes::displayDaltonismChoice = false [private]
```

#### 6.35.5.8 displayGameModeChoice

```
bool Scenes::displayGameModeChoice = false [private]
```

#### 6.35.5.9 displayKeyBinds

```
bool Scenes::displayKeyBinds = false [private]
```

#### 6.35.5.10 keyBinds

```
std::map<Actions, sf::Keyboard::Key> Scenes::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}}
```

#### 6.35.5.11 previousScene

```
Scene Scenes::previousScene = Scene::MAIN_MENU [private]
```

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

- /home/runner/work/R-Type/R-Type/ECS/Interface/Include/[scenes.hpp](#)
- /home/runner/work/R-Type/R-Type/Client/Src/[scenes.cpp](#)

## 6.36 ScoreComponent Struct Reference

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

### Public Attributes

- int [score](#)



### 6.36.1 Member Data Documentation

#### 6.36.1.1 score

```
int ScoreComponent::score
```

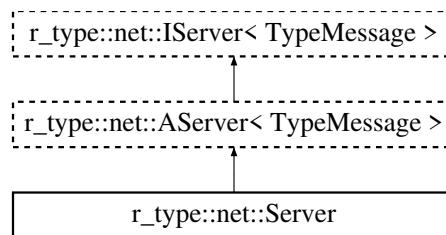
The documentation for this struct was generated from the following file:

- [/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/score\\_component.hpp](#)

## 6.37 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

#### 6.37.1 Constructor & Destructor Documentation

### 6.37.1.1 Server()

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

### 6.37.1.2 ~Server()

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

## 6.37.2 Member Function Documentation

### 6.37.2.1 OnClientConnect()

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

Called when a client is validated.

#### Parameters

<i>client</i>	
---------------	--

#### Returns

true

false

### 6.37.2.2 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

<i>client</i>	
---------------	--

### 6.37.2.3 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

<i>client</i>	
<i>msg</i>	

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](#)

## 6.38 SpriteComponent Struct Reference

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

### Public Member Functions

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

### Public Attributes

- sf::Sprite [sprite](#)

### 6.38.1 Constructor & Destructor Documentation

#### 6.38.1.1 SpriteComponent()

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

### 6.38.2 Member Data Documentation

### 6.38.2.1 sprite

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

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

- [/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/sprite\\_component.hpp](#)

## 6.39 SpriteDataComponent Struct Reference

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

### Public Attributes

- [SpritePath](#) [spritePath](#)
- [Vector< uint32\\_t >](#) [offSet](#)
- [Vector< uint32\\_t >](#) [dimension](#)
- [Vector< float >](#) [scale](#)

### 6.39.1 Member Data Documentation

#### 6.39.1.1 dimension

```
Vector<uint32\_t> SpriteDataComponent::dimension
```

#### 6.39.1.2 offSet

```
Vector<uint32\_t> SpriteDataComponent::offSet
```

#### 6.39.1.3 scale

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

#### 6.39.1.4 spritePath

```
SpritePath SpriteDataComponent::spritePath
```

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

- [/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/sprite\\_data\\_component.hpp](#)

## 6.40 TextComponent Struct Reference

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

### Public Member Functions

- [TextComponent](#) (std::string text)

### Public Attributes

- std::string [\\_text](#)

### 6.40.1 Constructor & Destructor Documentation

#### 6.40.1.1 TextComponent()

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

### 6.40.2 Member Data Documentation

#### 6.40.2.1 \_text

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

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

- [/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/text\\_component.hpp](#)

## 6.41 TextureManager Class Reference

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

### Public Member Functions

- `sf::Texture & getTexture (const std::string &filePath)`  
*Retrieves a texture from the texture manager.*

### Private Attributes

- `std::unordered_map< std::string, sf::Texture > textures`  
*A container for storing textures with string keys.*

## 6.41.1 Member Function Documentation

### 6.41.1.1 `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

<a href="#">failedToLoadTexture</a>	If the texture fails to load from the file path.
-------------------------------------	--

#### Parameters

<i>filePath</i>	The file path of the texture to retrieve.
-----------------	---

#### Returns

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

## 6.41.2 Member Data Documentation

### 6.41.2.1 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:

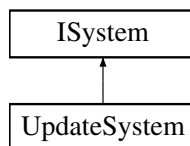
- `/home/runner/work/R-Type/R-Type/ECS/Interface/Include/texture\_manager.hpp`

## 6.42 UpdateSystem Class Reference

A system responsible for updating entities in the game.

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

Inheritance diagram for UpdateSystem:



### Public Member Functions

- [UpdateSystem](#) (`sf::RenderWindow &window`)
- void [update](#) ([EntityManager](#) &entityManager, [ComponentManager](#) &componentManager, float deltaTime)
- void [updateSpritePosition](#) ([EntityManager](#) &entityManager, [ComponentManager](#) &componentManager)
- void [updateBackground](#) ([ComponentManager](#) &componentManager, float deltaTime)
- void [setGameBgOffset](#) (int offset)
- int [getGameBgOffset](#) ()

### Private Attributes

- `sf::RenderWindow & _window`
- int [gameBgOffset](#) = 0

### 6.42.1 Detailed Description

A system responsible for updating entities in the game.

### 6.42.2 Constructor & Destructor Documentation

### 6.42.2.1 UpdateSystem()

```
UpdateSystem::UpdateSystem (
    sf::RenderWindow & window ) [inline]
```

## 6.42.3 Member Function Documentation

### 6.42.3.1 getGameBgOffset()

```
int UpdateSystem::getGameBgOffset ( ) [inline]
```

### 6.42.3.2 setGameBgOffset()

```
void UpdateSystem::setGameBgOffset (
    int offset ) [inline]
```

### 6.42.3.3 update()

```
void UpdateSystem::update (
    EntityManager & entityManager,
    ComponentManager & componentManager,
    float deltaTime )
```

### 6.42.3.4 updateBackground()

```
void UpdateSystem::updateBackground (
    ComponentManager & componentManager,
    float deltaTime )
```

### 6.42.3.5 updateSpritePosition()

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



## 6.42.4 Member Data Documentation

### 6.42.4.1 \_window

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

### 6.42.4.2 gameBgOffset

```
int UpdateSystem::gameBgOffset = 0 [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](#)

## 6.43 Vector< T > Struct Template Reference

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

### Public Attributes

- [T x](#)
- [T y](#)

### 6.43.1 Member Data Documentation

#### 6.43.1.1 x

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

#### 6.43.1.2 y

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

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

- [/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/sprite\\_data\\_component.hpp](#)

## 6.44 VelocityComponent Struct Reference

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

### Public Attributes

- float [speed](#)

### 6.44.1 Member Data Documentation

#### 6.44.1.1 [speed](#)

```
float VelocityComponent::speed
```

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

- [/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/velocity\\_component.hpp](#)

## 6.45 vf2d Struct Reference

Represents a 2D vector with x and y coordinates.

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

### Public Attributes

- float [x](#) = 0
- float [y](#) = 0

### 6.45.1 Detailed Description

Represents a 2D vector with x and y coordinates.

### 6.45.2 Member Data Documentation

#### 6.45.2.1 [x](#)

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

### 6.45.2.2 y

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

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

- [/home/runner/work/R-Type/R-Type/ECS/Interface/Include/entity\\_struct.hpp](#)

## 6.46 WeaponComponent Struct Reference

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

### Public Attributes

- float [damage](#)
- float [fire\\_rate](#)
- float [bullet\\_speed](#)
- float [bullet\\_lifetime](#)

### 6.46.1 Member Data Documentation

#### 6.46.1.1 bullet\_lifetime

```
float WeaponComponent::bullet_lifetime
```

#### 6.46.1.2 bullet\_speed

```
float WeaponComponent::bullet_speed
```

#### 6.46.1.3 damage

```
float WeaponComponent::damage
```

#### 6.46.1.4 fire\_rate

```
float WeaponComponent::fire_rate
```

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

- [/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/weapon\\_component.hpp](#)



## Chapter 7

# File Documentation

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

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

#### Functions

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

#### 7.1.1 Function Documentation

##### 7.1.1.1 MainMenu()

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

### 7.2 /home/runner/work/R-Type/R-Type/Client/Interface/Include/Net/a\_↵ client.hpp File Reference

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

## Classes

- class [r\\_type::net::AClient< T >](#)

## Namespaces

- [r\\_type](#)
- [r\\_type::net](#)

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

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

## Classes

- class [r\\_type::net::Client](#)

## Namespaces

- [r\\_type](#)
- [r\\_type::net](#)

### 7.4 /home/runner/work/R-Type/R-Type/Client/Interface/Include/Net/i\_↵ client.hpp File Reference

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

## Classes

- class [r\\_type::net::IClient< T >](#)

## Namespaces

- [r\\_type](#)
- [r\\_type::net](#)

## 7.5 /home/runner/work/R-Type/R-Type/Client/Interface/Include/r\_type\_client.hpp File Reference

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

### Classes

- class [Rtype](#)

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

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

### Functions

- int [main](#) ()  
*The entry point of the program.*

### 7.6.1 Function Documentation

#### 7.6.1.1 main()

```
int main ( )
```

The entry point of the program.

This function initializes the [Rtype](#) object and runs the game.

#### Returns

0 indicating successful program execution.  
int

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

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

## Functions

- int [main](#) ()

### 7.7.1 Function Documentation

#### 7.7.1.1 main()

```
int main ( )
```

## 7.8 /home/runner/work/R-Type/R-Type/Client/Src/r-type\_client.cpp File Reference

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

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

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

## Functions

- void [handleEvents](#) (sf::Event event, [ComponentManager](#) &componentManager, sf::RenderWindow \*\_↵ window, std::vector< [Entity](#) \* > \*buttons, [Scenes](#) \*scenes)
- void [createDaltonismChoiceButtons](#) (std::vector< [Entity](#) \* > \*buttons, [ComponentManager](#) &component↵ Manager, [EntityManager](#) &entityManager, [TextureManager](#) &textureManager, [EntityFactory](#) &entityFactory)
- void [createGameModeChoiceButtons](#) (std::vector< [Entity](#) \* > \*buttons, [ComponentManager](#) &component↵ Manager, [EntityManager](#) &entityManager, [TextureManager](#) &textureManager, [EntityFactory](#) &entityFactory)
- sf::Keyboard::Key [waitForKey](#) (sf::RenderWindow \*\_window)
- void [createKeyBindingButtons](#) (std::vector< [Entity](#) \* > \*buttons, [ComponentManager](#) &componentManager, [EntityManager](#) &entityManager, [TextureManager](#) &textureManager, [EntityFactory](#) &entityFactory)



## 7.9.1 Function Documentation

### 7.9.1.1 createDaltonismChoiceButtons()

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

### 7.9.1.2 createGameModeChoiceButtons()

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

### 7.9.1.3 createKeyBindingButtons()

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

### 7.9.1.4 handleEvents()

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

#### 7.9.1.5 waitForKey()

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

### 7.10 /home/runner/work/R-Type/R-Type/ECS/Interface/Include/↵ Components/ally\_component.hpp File Reference

#### Classes

- struct [AllyComponent](#)

### 7.11 /home/runner/work/R-Type/R-Type/ECS/Interface/Include/↵ Components/ally\_missile\_component.hpp File Reference

#### Classes

- struct [AllyMissileComponent](#)

### 7.12 /home/runner/work/R-Type/R-Type/ECS/Interface/Include/↵ Components/background\_component.hpp File Reference

#### Classes

- struct [BackgroundComponent](#)

### 7.13 /home/runner/work/R-Type/R-Type/ECS/Interface/Include/↵ Components/basic\_monster\_component.hpp File Reference

#### Classes

- struct [BasicMonsterComponent](#)

### 7.14 /home/runner/work/R-Type/R-Type/ECS/Interface/Include/↵ Components/bind\_component.hpp File Reference

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

## Classes

- struct [BindComponent](#)

## 7.15 /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.*

## 7.16 /home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/components.hpp File Reference ↵

```
#include "ally_component.hpp"
#include "ally_missile_component.hpp"
#include "background_component.hpp"
#include "basic_monster_component.hpp"
#include "enemy_component.hpp"
#include "enemy_missile_component.hpp"
#include "health_component.hpp"
#include "hitbox_component.hpp"
#include "input_component.hpp"
#include "offset_component.hpp"
#include "on_click_component.hpp"
#include "bind_component.hpp"
#include "player_component.hpp"
#include "player_missile_component.hpp"
#include "position_component.hpp"
#include "score_component.hpp"
#include "sprite_component.hpp"
#include "sprite_data_component.hpp"
#include "text_component.hpp"
#include "velocity_component.hpp"
#include "weapon_component.hpp"
```

## 7.17 [/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/enemy\\_component.hpp](#) File Reference

### Classes

- struct [EnemyComponent](#)

## 7.18 [/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/enemy\\_missile\\_component.hpp](#) File Reference

### Classes

- struct [EnemyMissileComponent](#)

## 7.19 [/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/health\\_component.hpp](#) File Reference

### Classes

- struct [HealthComponent](#)

## 7.20 [/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/hitbox\\_component.hpp](#) File Reference

### Classes

- struct [HitboxComponent](#)

## 7.21 [/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](#) }

#### 7.21.1 Enumeration Type Documentation

##### 7.21.1.1 InputType

```
enum InputType [strong]
```

## Enumerator

UP	
DOWN	
LEFT	
RIGHT	
SHOOT	
QUIT	
NONE	

## 7.22 /home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/label\_component.hpp File Reference

```
#include <iostream>
```

## Classes

- struct [labelComponent](#)

## 7.23 /home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/offset\_component.hpp File Reference

## Classes

- struct [OffsetComponent](#)

## 7.24 /home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/on\_click\_component.hpp File Reference

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

## Classes

- struct [OnClickComponent](#)

## 7.25 /home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/player\_component.hpp File Reference

## Classes

- struct [PlayerComponent](#)

## 7.26 /home/runner/work/R-Type/R-Type/ECS/Interface/Include/↵ Components/player\_missile\_component.hpp File Reference

### Classes

- struct [PlayerMissileComponent](#)

## 7.27 /home/runner/work/R-Type/R-Type/ECS/Interface/Include/↵ Components/position\_component.hpp File Reference

### Classes

- struct [PositionComponent](#)

## 7.28 /home/runner/work/R-Type/R-Type/ECS/Interface/Include/↵ Components/score\_component.hpp File Reference

### Classes

- struct [ScoreComponent](#)

## 7.29 /home/runner/work/R-Type/R-Type/ECS/Interface/Include/↵ Components/sprite\_component.hpp File Reference

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

### Classes

- struct [SpriteComponent](#)

## 7.30 /home/runner/work/R-Type/R-Type/ECS/Interface/Include/↵ Components/sprite\_data\_component.hpp File Reference

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

## Classes

- struct [Vector< T >](#)
- struct [SpriteDataComponent](#)

## 7.31 /home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/text\_component.hpp File Reference

```
#include <iostream>
```

## Classes

- struct [TextComponent](#)

## 7.32 /home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/velocity\_component.hpp File Reference

## Classes

- struct [VelocityComponent](#)

## 7.33 /home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/weapon\_component.hpp File Reference

## Classes

- struct [WeaponComponent](#)

## 7.34 /home/runner/work/R-Type/R-Type/ECS/Interface/Include/creatable\_client\_object.hpp File Reference

```
#include <stdint>
```

## Enumerations

- enum class [CreatableClientObject](#) : uint32\_t { [MISSILE](#) , [NONE](#) }

### 7.34.1 Enumeration Type Documentation

#### 7.34.1.1 CreatableClientObject

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

## Enumerator

MISSILE	
NONE	

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

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

## Classes

- class [Entity](#)  
*Represents an entity in the ECS system.*

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

```
#include "i_entity_factory.hpp"  
#include "scenes.hpp"  
#include <functional>
```

## Classes

- class [EntityFactory](#)  
*A class responsible for creating different types of entities.*

### 7.37 `/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Entities/entity_manager.hpp` File Reference

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

## Classes

- class [EntityManager](#)  
*Class responsible for managing entities in the ECS system.*



## 7.38 /home/runner/work/R-Type/R-Type/ECS/Interface/Include/Entities/i\_entity.hpp File Reference

### Classes

- class [IEntity](#)

*The [IEntity](#) class represents an entity in the system.*

## 7.39 /home/runner/work/R-Type/R-Type/ECS/Interface/Include/Entities/i\_entity\_factory.hpp File Reference

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

### Classes

- class [IEntityFactory](#)

*The interface for an entity factory.*

## 7.40 /home/runner/work/R-Type/R-Type/ECS/Interface/Include/entity\_struct.hpp File Reference

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

### Classes

- struct [vf2d](#)

*Represents a 2D vector with x and y coordinates.*

- struct [EntityInformation](#)

*Represents information about an entity.*

## 7.41 /home/runner/work/R-Type/R-Type/ECS/Interface/Include/error\_handling.hpp File Reference

```
#include <exception>
```

## Classes

- class [componentNotFound](#)  
*Exception class for when a component is not found.*
- class [entityNotFound](#)  
*Exception class for entity not found error.*
- class [failedToLoadTexture](#)  
*Exception class for failed texture loading.*

## 7.42 /home/runner/work/R-Type/R-Type/ECS/Interface/↔ Include/scenes.hpp File Reference

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

## Classes

- class [Scenes](#)  
*Represents a class that manages different scenes in a game.*

## 7.43 /home/runner/work/R-Type/R-Type/ECS/Interface/Include/sprite\_↔ path.hpp File Reference

```
#include <stdint>
#include <string>
```

## Enumerations

- enum class [SpritePath](#) : uint32\_t {  
    [Ship1](#) , [Ship2](#) , [Ship3](#) , [Ship4](#) ,  
    [Enemy1](#) , [Enemy2](#) , [Enemy3](#) , [Enemy4](#) ,  
    [Enemy5](#) , [Enemy6](#) , [Monster1](#) , [Monster2](#) ,  
    [Monster3](#) , [Monster4](#) , [Monster5](#) , [Missile](#) ,  
    [Background](#) , [Explosion](#) , [PowerUp](#) , [Boss](#) ,  
    [BossBullet](#) , [NONE](#) }

## Functions

- std::string [SpriteFactory](#) ([SpritePath](#) sprite)

### 7.43.1 Enumeration Type Documentation

#### 7.43.1.1 SpritePath

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

## Enumerator

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

## 7.43.2 Function Documentation

### 7.43.2.1 SpriteFactory()

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

## 7.44 /home/runner/work/R-Type/R-Type/ECS/Interface/Include/↵ Systems/button\_system.hpp File Reference

## 7.45 /home/runner/work/R-Type/R-Type/ECS/Interface/Include/↵ Systems/i\_system.hpp File Reference

```
#include "Components/component_manager.hpp"
#include "Components/components.hpp"
#include "Entities/entity_manager.hpp"
```

## Classes

- class [ISystem](#)

### 7.46 /home/runner/work/R-Type/R-Type/ECS/Interface/Include/↵ Systems/render\_system.hpp File Reference

```
#include "Systems/i_system.hpp"  
#include <SFML/Graphics.hpp>
```

## Classes

- class [RenderSystem](#)  
*A system responsible for rendering components.*

### 7.47 /home/runner/work/R-Type/R-Type/ECS/Interface/Include/↵ Systems/systems.hpp File Reference

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

### 7.48 /home/runner/work/R-Type/R-Type/ECS/Interface/Include/↵ Systems/update\_system.hpp File Reference

```
#include "Components/component_manager.hpp"  
#include "Components/components.hpp"  
#include "Entities/entity_manager.hpp"  
#include "Systems/i_system.hpp"
```

## Classes

- class [UpdateSystem](#)  
*A system responsible for updating entities in the game.*

### 7.49 /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](#)

## 7.50 /home/runner/work/R-Type/R-Type/ECS/Src/Entities/entity\_↵ factory.cpp File Reference

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

## Functions

- bool [CheckPositionEntity](#) ([EntityManager](#) &entityManager, [ComponentManager](#) &componentManager, u\_↵  
int32\_t entityID)

### 7.50.1 Function Documentation

#### 7.50.1.1 CheckPositionEntity()

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

## 7.51 /home/runner/work/R-Type/R-Type/ECS/Src/sprite\_path.cpp File Reference

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

## Functions

- std::string [SpriteFactory](#) ([SpritePath](#) sprite)

### 7.51.1 Function Documentation

### 7.51.1.1 SpriteFactory()

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

## 7.52 /home/runner/work/R-Type/R-Type/ECS/Src/Systems/render\_↵ system.cpp File Reference

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

## 7.53 /home/runner/work/R-Type/R-Type/ECS/Src/Systems/update\_↵ system.cpp File Reference

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

## 7.54 /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 <cmath>
#include <entity_struct.hpp>
#include <unordered_map>
```

### Classes

- class [r\\_type::net::AServer< T >](#)  
*AServer class.*

### Namespaces

- [r\\_type](#)
- [r\\_type::net](#)

## 7.55 /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](#)

## 7.56 /home/runner/work/R-Type/R-Type/Server/Interface/Include/r\_type\_↵ \_server.hpp File Reference

```
#include <iostream>
```

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

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





# Index

/home/runner/work/R-Type/R-Type/Client/Interface/Include/Net/a\_client.hpp, 87  
/home/runner/work/R-Type/R-Type/Client/Interface/Include/Net/client.hpp, 88  
/home/runner/work/R-Type/R-Type/Client/Interface/Include/Net/i\_client.hpp, 88  
/home/runner/work/R-Type/R-Type/Client/Interface/Include/mainmenu.hpp, 87  
/home/runner/work/R-Type/R-Type/Client/Interface/Include/r\_type\_client.hpp, 89  
/home/runner/work/R-Type/R-Type/Client/Src/main.cpp, 89  
/home/runner/work/R-Type/R-Type/Client/Src/r\_type\_client.cpp, 90  
/home/runner/work/R-Type/R-Type/Client/Src/scenes.cpp, 90  
/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/all\_component.hpp, 92  
/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/all\_missile\_component.hpp, 92  
/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/background\_component.hpp, 92  
/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/basic\_monster\_component.hpp, 92  
/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/bind\_component.hpp, 92  
/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/component\_manager.hpp, 93  
/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/components.hpp, 93  
/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/enemy\_component.hpp, 94  
/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/enemy\_missile\_component.hpp, 94  
/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/health\_component.hpp, 94  
/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/hitbox\_component.hpp, 94  
/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/input\_component.hpp, 94  
/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/label\_component.hpp, 95  
/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/offset\_component.hpp, 95  
/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/on\_click\_component.hpp, 95  
/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/player\_component.hpp, 95  
/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/player\_missile\_component.hpp, 96

/home/runner/work/R-Type/R-Type/ECS/Src/Systems/render\_system.cpp, 104  
 /home/runner/work/R-Type/R-Type/ECS/Src/Systems/update\_system.cpp, 104  
 /home/runner/work/R-Type/R-Type/ECS/Src/sprite\_path.cpp, 103  
 /home/runner/work/R-Type/R-Type/Server/Interface/Include/Net/BindComponent.hpp, 104  
 /home/runner/work/R-Type/R-Type/Server/Interface/Include/Net/Bind.hpp, 104  
 /home/runner/work/R-Type/R-Type/Server/Interface/Include/r\_type\_server.hpp, 105  
 /home/runner/work/R-Type/R-Type/Server/Src/main.cpp, 89  
 /home/runner/work/R-Type/R-Type/Server/Src/server.cpp, 105  
 \_clock  
     r\_type::net::AServer< T >, 25  
 \_id  
     Entity, 35  
 \_scenes  
     Rtype, 65  
 \_text  
     TextComponent, 79  
 \_window  
     RenderSystem, 63  
     Rtype, 66  
     Scenes, 73  
     UpdateSystem, 83  
 ~AClient  
     r\_type::net::AClient< T >, 12  
 ~AServer  
     r\_type::net::AServer< T >, 18  
 ~IClient  
     r\_type::net::IClient< T >, 48  
 ~IEntity  
     IEntity, 50  
 ~IEntityFactory  
     IEntityFactory, 52  
 ~Scenes  
     Scenes, 69  
 ~Server  
     r\_type::net::Server, 76  
 AClient  
     r\_type::net::AClient< T >, 12  
 Actions  
     Scenes, 67  
 addComponent  
     ComponentManager, 30  
 addEntity  
     r\_type::net::AClient< T >, 12  
     r\_type::net::Client, 29  
 AllyComponent, 16  
 AllyMissileComponent, 16  
 AServer  
     r\_type::net::AServer< T >, 18  
 Background  
     sprite\_path.hpp, 101  
     background  
         r\_type::AServer< T >, 25  
         BackgroundComponent, 27  
         BasicMonsterComponent, 27  
         bind  
         BindComponent, 28  
         Bind.hpp, 28  
         BindComponent, 28  
         is\_server.hpp, 28  
         binding  
         Scenes, 73  
         Boss  
         sprite\_path.hpp, 101  
     BossBullet  
         sprite\_path.hpp, 101  
     bullet\_lifetime  
         WeaponComponent, 85  
     bullet\_speed  
         WeaponComponent, 85  
     buttons  
         Scenes, 73  
     CheckPlayerPosition  
         r\_type::net::AServer< T >, 18  
     CheckPositionEntity  
         entity\_factory.cpp, 103  
     clientPlayerID  
         r\_type::net::AServer< T >, 25  
     ComponentManager, 30  
         addComponent, 30  
         components, 32  
         getComponent, 31  
         getComponentMap, 31  
         removeEntityFromComponent, 32  
     componentManager  
         r\_type::net::AServer< T >, 25  
     componentNotFound, 32  
         what, 33  
     components  
         ComponentManager, 32  
     Connect  
         r\_type::net::AClient< T >, 12  
         r\_type::net::IClient< T >, 48  
     creatable\_client\_object.hpp  
         CreatableClientObject, 97  
         MISSILE, 98  
         NONE, 98  
     CreatableClientObject, 33  
         creatable\_client\_object.hpp, 97  
     createAlly  
         EntityFactory, 36  
         IEntityFactory, 53  
     createAllyMissile  
         EntityFactory, 37  
         IEntityFactory, 53  
     createBackground  
         EntityFactory, 37

- IEntityFactory, 53
- createBasicEnemy
  - EntityFactory, 38
  - IEntityFactory, 54
- createBasicMonster
  - EntityFactory, 38
  - IEntityFactory, 54
- createButton
  - EntityFactory, 39
  - IEntityFactory, 55
- createDaltonismChoiceButtons
  - scenes.cpp, 91
- createEnemyMissile
  - EntityFactory, 39
  - IEntityFactory, 55
- createEntity
  - EntityManager, 43
- createGameModeChoiceButtons
  - scenes.cpp, 91
- createKeyBindingButtons
  - scenes.cpp, 91
- createPlayer
  - EntityFactory, 40
  - IEntityFactory, 56
- createPlayerMissile
  - EntityFactory, 40
  - IEntityFactory, 56
- createSmallButton
  - EntityFactory, 41
- currentDaltonismMode
  - Scenes, 73
- currentGameMode
  - Scenes, 73
- currentScene
  - Scenes, 73
- DaltonismMode
  - Scenes, 68
- damage
  - WeaponComponent, 85
- DEUTERANOPIA
  - Scenes, 68
- dimension
  - SpriteDataComponent, 78
- Disconnect
  - r\_type::net::AClient< T >, 13
  - r\_type::net::IClient< T >, 49
- displayDaltonismChoice
  - Scenes, 73
- displayGameModeChoice
  - Scenes, 74
- displayKeyBinds
  - Scenes, 74
- DOWN
  - input\_component.hpp, 95
  - Scenes, 68
- EASY
  - Scenes, 68
- Enemy1
  - sprite\_path.hpp, 101
- Enemy2
  - sprite\_path.hpp, 101
- Enemy3
  - sprite\_path.hpp, 101
- Enemy4
  - sprite\_path.hpp, 101
- Enemy5
  - sprite\_path.hpp, 101
- Enemy6
  - sprite\_path.hpp, 101
- EnemyComponent, 33
- EnemyMissileComponent, 34
- entities
  - EntityManager, 44
- Entity, 34
  - \_id, 35
  - Entity, 34
  - getId, 35
- entity\_factory.cpp
  - CheckPositionEntity, 103
- EntityFactory, 35
  - createAlly, 36
  - createAllyMissile, 37
  - createBackground, 37
  - createBasicEnemy, 38
  - createBasicMonster, 38
  - createButton, 39
  - createEnemyMissile, 39
  - createPlayer, 40
  - createPlayerMissile, 40
  - createSmallButton, 41
- entityFactory
  - r\_type::net::AServer< T >, 25
- EntityInformation, 41
  - spriteData, 42
  - uniqueID, 42
  - vPos, 42
- EntityManager, 42
  - createEntity, 43
  - entities, 44
  - entityNb, 44
  - getAllEntities, 43
  - getEntity, 43
  - removeEntity, 44
- entityManager
  - r\_type::net::AServer< T >, 25
- entityNb
  - EntityManager, 44
- entityNotFound, 45
  - what, 45
- EXIT
  - Scenes, 69
- Explosion
  - sprite\_path.hpp, 101
- failedToLoadTexture, 45
  - what, 46

- FIRE
  - Scenes, 68
- fire\_rate
  - WeaponComponent, 85
- GAME\_LOOP
  - Scenes, 69
- gameBgOffset
  - UpdateSystem, 83
- gameLoop
  - Rtype, 64
  - Scenes, 69
- GameMode
  - Scenes, 68
- getAllEntities
  - EntityManager, 43
- GetClientEntityId
  - r\_type::net::AServer< T >, 19
- getComponent
  - ComponentManager, 31
- getComponentMap
  - ComponentManager, 31
- getConnection
  - r\_type::net::AClient< T >, 13
- getEntity
  - EntityManager, 43
- getGameBgOffset
  - UpdateSystem, 82
- getId
  - Entity, 35
  - IEntity, 51
- getPlayerId
  - r\_type::net::AClient< T >, 13
- getPreviousScene
  - Scenes, 70
- getRenderWindow
  - Scenes, 70
- getTexture
  - TextureManager, 80
- h
  - HitboxComponent, 47
- handleEvents
  - Rtype, 64
  - scenes.cpp, 91
- HARD
  - Scenes, 68
- health
  - HealthComponent, 46
- HealthComponent, 46
  - health, 46
  - max\_health, 46
- HitboxComponent, 47
  - h, 47
  - w, 47
- IClient
  - r\_type::net::IClient< T >, 48
- IEntity, 50
  - ~IEntity, 50
  - getId, 51
- IEntityFactory, 51
  - ~IEntityFactory, 52
  - createAlly, 53
  - createAllyMissile, 53
  - createBackground, 53
  - createBasicEnemy, 54
  - createBasicMonster, 54
  - createButton, 55
  - createEnemyMissile, 55
  - createPlayer, 56
  - createPlayerMissile, 56
- IN\_GAME\_MENU
  - Scenes, 69
- Incoming
  - r\_type::net::AClient< T >, 13
  - r\_type::net::IClient< T >, 49
- inGameMenu
  - Scenes, 70
- InitiateBackground
  - r\_type::net::AServer< T >, 19
- InitiateMissile
  - r\_type::net::AServer< T >, 19
- InitiatePlayers
  - r\_type::net::AServer< T >, 20
- InitListEntities
  - r\_type::net::AServer< T >, 20
- input
  - InputComponent, 57
- input\_component.hpp
  - DOWN, 95
  - InputType, 94
  - LEFT, 95
  - NONE, 95
  - QUIT, 95
  - RIGHT, 95
  - SHOOT, 95
  - UP, 95
- InputComponent, 57
  - input, 57
- InputType
  - input\_component.hpp, 94
- isClicked
  - OnClickComponent, 60
- IsConnected
  - r\_type::net::AClient< T >, 13
  - r\_type::net::IClient< T >, 49
- isHovered
  - BindComponent, 28
- ISystem, 57
  - ISystem, 58
- keyBinds
  - Scenes, 74
- labelComponent, 58
  - name, 58
  - x, 58

- y, 58
- LEFT
  - input\_component.hpp, 95
  - Scenes, 68
- m\_asioContext
  - r\_type::net::AServer< T >, 26
- m\_asioSocket
  - r\_type::net::AServer< T >, 26
- m\_clientEndpoint
  - r\_type::net::AServer< T >, 26
- m\_connection
  - r\_type::net::AClient< T >, 15
- m\_context
  - r\_type::net::AClient< T >, 15
- m\_deqConnections
  - r\_type::net::AServer< T >, 26
- m\_qMessagesIn
  - r\_type::net::AClient< T >, 15
  - r\_type::net::AServer< T >, 26
- m\_tempBuffer
  - r\_type::net::AServer< T >, 26
- m\_threadContext
  - r\_type::net::AServer< T >, 26
- main
  - main.cpp, 89, 90
- main.cpp
  - main, 89, 90
- MAIN\_MENU
  - Scenes, 68
- MainMenu
  - mainmenu.hpp, 87
- mainMenu
  - Rtype, 64
  - Scenes, 71
- mainmenu.hpp
  - MainMenu, 87
- max\_health
  - HealthComponent, 46
- MEDIUM
  - Scenes, 68
- MessageAll
  - r\_type::net::Client, 29
- MessageAllClients
  - r\_type::net::AServer< T >, 21
- MessageClient
  - r\_type::net::AServer< T >, 21
- MISSILE
  - creatable\_client\_object.hpp, 98
- Missile
  - sprite\_path.hpp, 101
- Monster1
  - sprite\_path.hpp, 101
- Monster2
  - sprite\_path.hpp, 101
- Monster3
  - sprite\_path.hpp, 101
- Monster4
  - sprite\_path.hpp, 101
- Monster5
  - sprite\_path.hpp, 101
- name
  - labelComponent, 58
- nbrOfPlayers
  - r\_type::net::AServer< T >, 27
- nIDCounter
  - r\_type::net::AServer< T >, 27
- NONE
  - creatable\_client\_object.hpp, 98
  - input\_component.hpp, 95
  - sprite\_path.hpp, 101
- NORMAL
  - Scenes, 68
- offSet
  - SpriteDataComponent, 78
- offset
  - OffsetComponent, 59
- OffsetComponent, 59
  - offset, 59
- onClick
  - OnClickComponent, 60
- OnClickComponent, 59
  - isClicked, 60
  - onClick, 60
  - OnClickComponent, 60
- OnClientConnect
  - r\_type::net::AServer< T >, 21
  - r\_type::net::Server, 76
- OnClientDisconnect
  - r\_type::net::AServer< T >, 22
  - r\_type::net::Server, 76
- OnClientValidated
  - r\_type::net::AServer< T >, 22
- OnMessage
  - r\_type::net::AServer< T >, 22
  - r\_type::net::Server, 76
- PAUSE
  - Scenes, 68
- PingServer
  - r\_type::net::Client, 29
- PlayerComponent, 60
- playerId
  - r\_type::net::AClient< T >, 15
- PlayerMissileComponent, 60
- PositionComponent, 61
  - PositionComponent, 61
  - x, 61
  - y, 61
- PowerUp
  - sprite\_path.hpp, 101
- previousScene
  - Scenes, 74
- processServerMessages
  - Rtype, 65
- PROTANOPIA

- Scenes, 68
- QUIT
  - input\_component.hpp, 95
  - Scenes, 68
- r\_type, 9
- r\_type::net, 9
- r\_type::net::AClient< T >, 11
  - ~AClient, 12
  - AClient, 12
  - addEntity, 12
  - Connect, 12
  - Disconnect, 13
  - getConnection, 13
  - getPlayerId, 13
  - Incoming, 13
  - IsConnected, 13
  - m\_connection, 15
  - m\_context, 15
  - m\_qMessagesIn, 15
  - playerId, 15
  - removeEntity, 14
  - Send, 14
  - setPlayerId, 14
  - thrContext, 15
  - updateEntity, 14
- r\_type::net::AServer< T >, 16
  - \_clock, 25
  - ~AServer, 18
  - AServer, 18
  - background, 25
  - CheckPlayerPosition, 18
  - clientPlayerID, 25
  - componentManager, 25
  - entityFactory, 25
  - entityManager, 25
  - GetClientEntityId, 19
  - InitiateBackground, 19
  - InitiateMissile, 19
  - InitiatePlayers, 20
  - InitListEntities, 20
  - m\_asioContext, 26
  - m\_asioSocket, 26
  - m\_clientEndpoint, 26
  - m\_deqConnections, 26
  - m\_qMessagesIn, 26
  - m\_tempBuffer, 26
  - m\_threadContext, 26
  - MessageAllClients, 21
  - MessageClient, 21
  - nbrOfPlayers, 27
  - nIDCounter, 27
  - OnClientConnect, 21
  - OnClientDisconnect, 22
  - OnClientValidated, 22
  - OnMessage, 22
  - RemoveEntities, 22
  - RemovePlayer, 23
- Start, 23
- Stop, 23
- Update, 23
- UpdateEntityPosition, 24
- WaitForClientMessage, 24
- r\_type::net::Client, 28
  - addEntity, 29
  - MessageAll, 29
  - PingServer, 29
  - removeEntity, 29
  - updateEntity, 29
- r\_type::net::IClient< T >, 47
  - ~IClient, 48
  - Connect, 48
  - Disconnect, 49
  - IClient, 48
  - Incoming, 49
  - IsConnected, 49
  - Send, 49
- r\_type::net::Server, 75
  - ~Server, 76
  - OnClientConnect, 76
  - OnClientDisconnect, 76
  - OnMessage, 76
  - Server, 75
- RemoveEntities
  - r\_type::net::AServer< T >, 22
- removeEntity
  - EntityManager, 44
  - r\_type::net::AClient< T >, 14
  - r\_type::net::Client, 29
- removeEntityFromComponent
  - ComponentManager, 32
- RemovePlayer
  - r\_type::net::AServer< T >, 23
- render
  - RenderSystem, 63
  - Scenes, 71
- renderGame
  - Rtype, 65
- RenderSystem, 62
  - \_window, 63
  - render, 63
  - RenderSystem, 62
- RIGHT
  - input\_component.hpp, 95
  - Scenes, 68
- Rtype, 63
  - \_scenes, 65
  - \_window, 66
  - gameLoop, 64
  - handleEvents, 64
  - mainMenu, 64
  - processServerMessages, 65
  - renderGame, 65
  - Rtype, 64
  - run, 65
  - updateGame, 65

- run
  - Rtype, 65
- scale
  - SpriteDataComponent, 78
- Scene
  - Scenes, 68
- Scenes, 66
  - \_window, 73
  - ~Scenes, 69
  - Actions, 67
  - binding, 73
  - buttons, 73
  - currentDaltonismMode, 73
  - currentGameMode, 73
  - currentScene, 73
  - DaltonismMode, 68
  - DEUTERANOPIA, 68
  - displayDaltonismChoice, 73
  - displayGameModeChoice, 74
  - displayKeyBinds, 74
  - DOWN, 68
  - EASY, 68
  - EXIT, 69
  - FIRE, 68
  - GAME\_LOOP, 69
  - gameLoop, 69
  - GameMode, 68
  - getPreviousScene, 70
  - getRenderWindow, 70
  - HARD, 68
  - IN\_GAME\_MENU, 69
  - inGameMenu, 70
  - keyBinds, 74
  - LEFT, 68
  - MAIN\_MENU, 68
  - mainMenu, 71
  - MEDIUM, 68
  - NORMAL, 68
  - PAUSE, 68
  - previousScene, 74
  - PROTANOPIA, 68
  - QUIT, 68
  - render, 71
  - RIGHT, 68
  - Scene, 68
  - Scenes, 69
  - setDaltonism, 71
  - setGameMode, 72
  - setScene, 72
  - SETTINGS\_MENU, 69
  - settingsMenu, 72
  - shouldQuit, 72
  - TRITANOPIA, 68
  - UP, 68
- scenes.cpp
  - createDaltonismChoiceButtons, 91
  - createGameModeChoiceButtons, 91
  - createKeyBindingButtons, 91
  - handleEvents, 91
  - waitForKey, 91
- score
  - ScoreComponent, 75
- ScoreComponent, 74
  - score, 75
- Send
  - r\_type::net::AClient< T >, 14
  - r\_type::net::IClient< T >, 49
- Server
  - r\_type::net::Server, 75
- setDaltonism
  - Scenes, 71
- setGameBgOffset
  - UpdateSystem, 82
- setGameMode
  - Scenes, 72
- setPlayerId
  - r\_type::net::AClient< T >, 14
- setScene
  - Scenes, 72
- SETTINGS\_MENU
  - Scenes, 69
- settingsMenu
  - Scenes, 72
- Ship1
  - sprite\_path.hpp, 101
- Ship2
  - sprite\_path.hpp, 101
- Ship3
  - sprite\_path.hpp, 101
- Ship4
  - sprite\_path.hpp, 101
- SHOOT
  - input\_component.hpp, 95
- shouldQuit
  - Scenes, 72
- speed
  - VelocityComponent, 84
- sprite
  - SpriteComponent, 77
- sprite\_path.cpp
  - SpriteFactory, 103
- sprite\_path.hpp
  - Background, 101
  - Boss, 101
  - BossBullet, 101
  - Enemy1, 101
  - Enemy2, 101
  - Enemy3, 101
  - Enemy4, 101
  - Enemy5, 101
  - Enemy6, 101
  - Explosion, 101
  - Missile, 101
  - Monster1, 101
  - Monster2, 101
  - Monster3, 101

- Monster4, [101](#)
- Monster5, [101](#)
- NONE, [101](#)
- PowerUp, [101](#)
- Ship1, [101](#)
- Ship2, [101](#)
- Ship3, [101](#)
- Ship4, [101](#)
- SpriteFactory, [101](#)
- SpritePath, [100](#)
- SpriteComponent, [77](#)
  - sprite, [77](#)
  - SpriteComponent, [77](#)
- spriteData
  - EntityInformation, [42](#)
- SpriteDataComponent, [78](#)
  - dimension, [78](#)
  - offSet, [78](#)
  - scale, [78](#)
  - spritePath, [78](#)
- SpriteFactory
  - sprite\_path.cpp, [103](#)
  - sprite\_path.hpp, [101](#)
- SpritePath
  - sprite\_path.hpp, [100](#)
- spritePath
  - SpriteDataComponent, [78](#)
- Start
  - r\_type::net::AServer< T >, [23](#)
- Stop
  - r\_type::net::AServer< T >, [23](#)
- TextComponent, [79](#)
  - \_text, [79](#)
  - TextComponent, [79](#)
- TextureManager, [80](#)
  - getTexture, [80](#)
  - textures, [80](#)
- textures
  - TextureManager, [80](#)
- thrContext
  - r\_type::net::AClient< T >, [15](#)
- TRITANOPIA
  - Scenes, [68](#)
- uniqueID
  - EntityInformation, [42](#)
- UP
  - input\_component.hpp, [95](#)
  - Scenes, [68](#)
- Update
  - r\_type::net::AServer< T >, [23](#)
- update
  - UpdateSystem, [82](#)
- updateBackground
  - UpdateSystem, [82](#)
- updateEntity
  - r\_type::net::AClient< T >, [14](#)
  - r\_type::net::Client, [29](#)
- UpdateEntityPosition
  - r\_type::net::AServer< T >, [24](#)
- updateGame
  - Rtype, [65](#)
- updateSpritePosition
  - UpdateSystem, [82](#)
- UpdateSystem, [81](#)
  - \_window, [83](#)
  - gameBgOffset, [83](#)
  - getGameBgOffset, [82](#)
  - setGameBgOffset, [82](#)
  - update, [82](#)
  - updateBackground, [82](#)
  - updateSpritePosition, [82](#)
  - UpdateSystem, [81](#)
- Vector< T >, [83](#)
  - x, [83](#)
  - y, [83](#)
- VelocityComponent, [84](#)
  - speed, [84](#)
- vf2d, [84](#)
  - x, [84](#)
  - y, [84](#)
- vPos
  - EntityInformation, [42](#)
- w
  - HitboxComponent, [47](#)
- WaitForClientMessage
  - r\_type::net::AServer< T >, [24](#)
- waitForKey
  - scenes.cpp, [91](#)
- WeaponComponent, [85](#)
  - bullet\_lifetime, [85](#)
  - bullet\_speed, [85](#)
  - damage, [85](#)
  - fire\_rate, [85](#)
- what
  - componentNotFound, [33](#)
  - entityNotFound, [45](#)
  - failedToLoadTexture, [46](#)
- x
  - labelComponent, [58](#)
  - PositionComponent, [61](#)
  - Vector< T >, [83](#)
  - vf2d, [84](#)
- y
  - labelComponent, [58](#)
  - PositionComponent, [61](#)
  - Vector< T >, [83](#)
  - vf2d, [84](#)