

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 AbstractScenes Class Reference	11
6.1.1 Detailed Description	11
6.2 r_type::net::AClient< T > Class Template Reference	11
6.2.1 Constructor & Destructor Documentation	12
6.2.1.1 AClient()	12
6.2.1.2 ~AClient()	12
6.2.2 Member Function Documentation	13
6.2.2.1 Connect()	13
6.2.2.2 Disconnect()	13
6.2.2.3 getConnection()	13
6.2.2.4 getPlayerId()	14
6.2.2.5 getWindowSize()	14
6.2.2.6 Incoming()	14
6.2.2.7 isConnected()	14
6.2.2.8 Send()	14
6.2.2.9 setPlayerId()	15
6.2.2.10 setWindowSize()	15
6.2.3 Member Data Documentation	15
6.2.3.1 m_connection	15
6.2.3.2 m_context	15
6.2.3.3 m_qMessagesIn	15
6.2.3.4 playerId	16
6.2.3.5 thrContext	16
6.2.3.6 windowSize	16
6.3 AllyComponent Struct Reference	16
6.4 AllyMissileComponent Struct Reference	16
6.5 AnimationComponent Struct Reference	16

6.5.1 Constructor & Destructor Documentation	17
6.5.1.1 AnimationComponent()	17
6.5.2 Member Data Documentation	17
6.5.2.1 dimension	17
6.5.2.2 offset	17
6.6 AnimationSystem Class Reference	17
6.6.1 Constructor & Destructor Documentation	18
6.6.1.1 AnimationSystem()	18
6.6.2 Member Function Documentation	18
6.6.2.1 animateBasicMonster()	18
6.6.2.2 animatePlayer()	18
6.6.2.3 animateWeapon()	19
6.6.2.4 AnimationEntities()	19
6.6.3 Member Data Documentation	19
6.6.3.1 _componentManager	19
6.6.3.2 _entityManager	20
6.7 AScenes Class Reference	20
6.7.1 Member Enumeration Documentation	21
6.7.1.1 Actions	22
6.7.1.2 DaltonismMode	22
6.7.1.3 GameMode	23
6.7.1.4 Scene	23
6.7.1.5 SpriteType	23
6.7.2 Constructor & Destructor Documentation	24
6.7.2.1 AScenes()	24
6.7.2.2 ~AScenes()	24
6.7.3 Member Function Documentation	24
6.7.3.1 getDaltonism()	24
6.7.3.2 getDisplayDaltonismChoice()	25
6.7.3.3 getDisplayGameModeChoice()	25
6.7.3.4 getDisplayKeyBindsChoice()	25
6.7.3.5 getIp()	25
6.7.3.6 getPort()	25
6.7.3.7 getPreviousScene()	25
6.7.3.8 setDaltonism()	25
6.7.3.9 setDisplayDaltonismChoice()	26
6.7.3.10 setDisplayGameModeChoice()	26
6.7.3.11 setDisplayKeyBindsChoice()	26
6.7.3.12 setGameMode()	26
6.7.3.13 setIp()	26
6.7.3.14 setPort()	27
6.7.3.15 setScene()	27

6.7.4 Member Data Documentation	27
6.7.4.1 _currentDaltonismMode	27
6.7.4.2 _currentGameMode	27
6.7.4.3 _currentScene	27
6.7.4.4 _displayDaltonismChoice	27
6.7.4.5 _displayGameModeChoice	28
6.7.4.6 _displayKeyBindsChoice	28
6.7.4.7 _ip	28
6.7.4.8 _port	28
6.7.4.9 _previousScene	28
6.7.4.10 buttons	28
6.7.4.11 filter	28
6.7.4.12 keyBinds	29
6.8 r_type::net::AServer< T > Class Template Reference	29
6.8.1 Detailed Description	32
6.8.2 Constructor & Destructor Documentation	32
6.8.2.1 AServer()	32
6.8.2.2 ~AServer()	32
6.8.3 Member Function Documentation	33
6.8.3.1 FormatEntityInformation()	33
6.8.3.2 getClientById()	33
6.8.3.3 GetClientInfoBarId()	33
6.8.3.4 GetClientPlayerId()	33
6.8.3.5 GetClock()	34
6.8.3.6 GetComponentManager()	34
6.8.3.7 GetEntityFactory()	35
6.8.3.8 GetEntityManager()	35
6.8.3.9 GetPlayerClientId()	35
6.8.3.10 InitiateBackground()	35
6.8.3.11 InitiateEnemyMissile()	36
6.8.3.12 InitiatePlayer()	36
6.8.3.13 InitiatePlayerMissile()	36
6.8.3.14 InitiateWeaponForce()	37
6.8.3.15 InitInfoBar()	37
6.8.3.16 MessageAllClients()	37
6.8.3.17 MessageClient()	37
6.8.3.18 OnClientConnect()	38
6.8.3.19 OnClientDisconnect()	38
6.8.3.20 OnClientValidated()	38
6.8.3.21 OnMessage()	39
6.8.3.22 RemoveEntity()	39
6.8.3.23 RemoveInfoBar()	39

6.8.3.24 RemovePlayer()	40
6.8.3.25 SetClock()	40
6.8.3.26 Start()	40
6.8.3.27 Stop()	40
6.8.3.28 Update()	41
6.8.3.29 UpdateInfoBar()	41
6.8.3.30 UpdatePlayerPosition()	41
6.8.3.31 WaitForClientMessage()	42
6.8.4 Member Data Documentation	42
6.8.4.1 _asioContext	42
6.8.4.2 _asioSocket	42
6.8.4.3 _background	43
6.8.4.4 _clientEndpoint	43
6.8.4.5 _clientInfoBarID	43
6.8.4.6 _clientPlayerID	43
6.8.4.7 _clock	43
6.8.4.8 _componentManager	44
6.8.4.9 _deqConnections	44
6.8.4.10 _entityFactory	44
6.8.4.11 _entityManager	44
6.8.4.12 _level	44
6.8.4.13 _nbrOfPlayers	45
6.8.4.14 _nIDCounter	45
6.8.4.15 _playerConnected	45
6.8.4.16 _port	45
6.8.4.17 _qMessagesIn	45
6.8.4.18 _tempBuffer	45
6.8.4.19 _threadContext	46
6.9 AudioManager Class Reference	46
6.9.1 Member Function Documentation	46
6.9.1.1 getSoundBuffer()	46
6.9.2 Member Data Documentation	46
6.9.2.1 soundBuffers	47
6.10 AudioSystem Class Reference	47
6.10.1 Constructor & Destructor Documentation	47
6.10.1.1 AudioSystem()	47
6.10.2 Member Function Documentation	48
6.10.2.1 playBackgroundMusic()	48
6.10.2.2 playSoundEffect()	48
6.10.2.3 stopBackgroundMusic()	48
6.10.3 Member Data Documentation	48
6.10.3.1 _audioManager	48

6.10.3.2 _backgroundMusic . . . . .	48
6.10.3.3 _currentMusicFilePath . . . . .	48
6.10.3.4 _soundEffect . . . . .	49
6.11 AutoFireSystem Class Reference . . . . .	49
6.11.1 Constructor & Destructor Documentation . . . . .	49
6.11.1.1 AutoFireSystem() . . . . .	49
6.11.2 Member Function Documentation . . . . .	49
6.11.2.1 handleAutoFire() . . . . .	50
6.11.3 Member Data Documentation . . . . .	50
6.11.3.1 _componentManager . . . . .	50
6.11.3.2 _entityManager . . . . .	50
6.12 BackgroundComponent Struct Reference . . . . .	50
6.13 BasicMonsterComponent Struct Reference . . . . .	50
6.14 BindComponent Struct Reference . . . . .	50
6.14.1 Constructor & Destructor Documentation . . . . .	51
6.14.1.1 BindComponent() . . . . .	51
6.14.2 Member Data Documentation . . . . .	51
6.14.2.1 bind . . . . .	51
6.14.2.2 isHovered . . . . .	51
6.15 BossComponent Struct Reference . . . . .	51
6.16 r_type::net::Client Class Reference . . . . .	52
6.16.1 Member Function Documentation . . . . .	52
6.16.1.1 addEntity() . . . . .	52
6.16.1.2 animateEntity() . . . . .	53
6.16.1.3 initInfoBar() . . . . .	53
6.16.1.4 MessageAll() . . . . .	53
6.16.1.5 moveEntity() . . . . .	53
6.16.1.6 PingServer() . . . . .	53
6.16.1.7 removeEntity() . . . . .	54
6.16.1.8 updateInfoBar() . . . . .	54
6.17 CollisionSystem Class Reference . . . . .	54
6.17.1 Constructor & Destructor Documentation . . . . .	54
6.17.1.1 CollisionSystem() . . . . .	55
6.17.2 Member Function Documentation . . . . .	55
6.17.2.1 checkCollision() . . . . .	55
6.17.2.2 checkOffScreen() . . . . .	55
6.17.3 Member Data Documentation . . . . .	55
6.17.3.1 _componentManager . . . . .	55
6.17.3.2 _entityManager . . . . .	55
6.18 ComponentManager Class Reference . . . . .	56
6.18.1 Detailed Description . . . . .	56
6.18.2 Member Function Documentation . . . . .	56

6.18.2.1 addComponent()	56
6.18.2.2 getComponent()	57
6.18.2.3 getComponentMap()	57
6.18.2.4 removeEntityFromAllComponents()	58
6.18.2.5 removeEntityFromComponent()	58
6.18.3 Member Data Documentation	58
6.18.3.1 components	58
6.19 componentNotFound Class Reference	58
6.19.1 Detailed Description	59
6.19.2 Member Function Documentation	59
6.19.2.1 what()	59
6.20 CreatableClientObject Class Reference	59
6.20.1 Detailed Description	59
6.21 EnemyComponent Struct Reference	59
6.22 EnemyMissileComponent Struct Reference	60
6.23 Entity Class Reference	60
6.23.1 Detailed Description	60
6.23.2 Constructor & Destructor Documentation	60
6.23.2.1 Entity()	60
6.23.3 Member Function Documentation	61
6.23.3.1 getId()	61
6.23.4 Member Data Documentation	61
6.23.4.1 _id	61
6.24 EntityFactory Class Reference	61
6.24.1 Detailed Description	62
6.24.2 Member Function Documentation	62
6.24.2.1 createBackground()	63
6.24.2.2 createBasicMonster()	63
6.24.2.3 createButton()	64
6.24.2.4 createEnemyMissile()	64
6.24.2.5 createFilter()	65
6.24.2.6 createForceMissile()	65
6.24.2.7 createForceWeapon()	66
6.24.2.8 createInfoBar()	66
6.24.2.9 createPlayer()	66
6.24.2.10 createPlayerMissile()	67
6.24.2.11 createPowerUpBlueLaserCrystal()	67
6.24.2.12 createShooterEnemy()	67
6.24.2.13 createSmallButton()	68
6.25 EntityInformation Struct Reference	69
6.25.1 Detailed Description	69
6.25.2 Member Data Documentation	69



6.25.2.1 animationComponent . . . . .	69
6.25.2.2 ratio . . . . .	69
6.25.2.3 spriteData . . . . .	69
6.25.2.4 uniqueID . . . . .	69
6.25.2.5 vPos . . . . .	70
6.26 EntityManager Class Reference . . . . .	70
6.26.1 Detailed Description . . . . .	70
6.26.2 Member Function Documentation . . . . .	70
6.26.2.1 createEntity() . . . . .	70
6.26.2.2 getAllEntities() . . . . .	71
6.26.2.3 getEntity() . . . . .	71
6.26.2.4 removeEntity() . . . . .	71
6.26.3 Member Data Documentation . . . . .	71
6.26.3.1 entities . . . . .	72
6.26.3.2 entityNb . . . . .	72
6.27 entityNotFound Class Reference . . . . .	72
6.27.1 Detailed Description . . . . .	72
6.27.2 Member Function Documentation . . . . .	72
6.27.2.1 what() . . . . .	73
6.28 failedToLoadFont Class Reference . . . . .	73
6.28.1 Member Function Documentation . . . . .	73
6.28.1.1 what() . . . . .	73
6.29 failedToLoadSound Class Reference . . . . .	73
6.29.1 Member Function Documentation . . . . .	74
6.29.1.1 what() . . . . .	74
6.30 failedToLoadTexture Class Reference . . . . .	74
6.30.1 Detailed Description . . . . .	74
6.30.2 Member Function Documentation . . . . .	74
6.30.2.1 what() . . . . .	75
6.31 FontManager Class Reference . . . . .	75
6.31.1 Member Function Documentation . . . . .	75
6.31.1.1 getFont() . . . . .	75
6.31.1.2 releaseFont() . . . . .	75
6.31.2 Member Data Documentation . . . . .	75
6.31.2.1 fonts . . . . .	76
6.32 ForceMissileComponent Struct Reference . . . . .	76
6.32.1 Member Data Documentation . . . . .	76
6.32.1.1 forceld . . . . .	76
6.33 FrontComponent Struct Reference . . . . .	76
6.33.1 Constructor & Destructor Documentation . . . . .	77
6.33.1.1 FrontComponent() . . . . .	77
6.33.2 Member Data Documentation . . . . .	77

6.33.2.1 targetId	77
6.34 HealthComponent Struct Reference	77
6.34.1 Member Data Documentation	77
6.34.1.1 health	77
6.34.1.2 max_health	78
6.35 HitboxComponent Struct Reference	78
6.35.1 Member Data Documentation	78
6.35.1.1 h	78
6.35.1.2 w	78
6.36 r_type::net::IClient< T > Class Template Reference	78
6.36.1 Constructor & Destructor Documentation	79
6.36.1.1 IClient()	79
6.36.1.2 ~IClient()	79
6.36.2 Member Function Documentation	79
6.36.2.1 Connect()	79
6.36.2.2 Disconnect()	80
6.36.2.3 Incoming()	80
6.36.2.4 IsConnected()	80
6.36.2.5 Send()	80
6.37 IEntityFactory Class Reference	81
6.37.1 Detailed Description	82
6.37.2 Member Enumeration Documentation	82
6.37.2.1 EnemyType	82
6.37.3 Constructor & Destructor Documentation	82
6.37.3.1 ~IEntityFactory()	83
6.37.4 Member Function Documentation	83
6.37.4.1 createBackground()	83
6.37.4.2 createBasicMonster()	83
6.37.4.3 createButton()	84
6.37.4.4 createEnemyMissile()	84
6.37.4.5 createForceMissile()	85
6.37.4.6 createForceWeapon()	85
6.37.4.7 createInfoBar()	85
6.37.4.8 createPlayer()	86
6.37.4.9 createPlayerMissile()	86
6.37.4.10 createPowerUpBlueLaserCrystal()	87
6.37.4.11 createShooterEnemy()	87
6.37.4.12 createSmallButton()	87
6.38 InputComponent Struct Reference	88
6.38.1 Member Data Documentation	88
6.38.1.1 input	88
6.39 IScenes Class Reference	88

6.39.1 Detailed Description	89
6.39.2 Constructor & Destructor Documentation	89
6.39.2.1 ~IScenes()	89
6.39.3 Member Function Documentation	89
6.39.3.1 difficultyChoices()	89
6.39.3.2 gameLoop()	90
6.39.3.3 getRenderWindow()	90
6.39.3.4 inGameMenu()	90
6.39.3.5 mainMenu()	90
6.39.3.6 render()	90
6.39.3.7 settingsMenu()	91
6.39.3.8 shouldQuit()	91
6.40 ISystem Class Reference	91
6.40.1 Constructor & Destructor Documentation	91
6.40.1.1 ISystem()	92
6.40.1.2 ~ISystem()	92
6.41 labelComponent Struct Reference	92
6.41.1 Member Data Documentation	92
6.41.1.1 name	92
6.41.1.2 x	92
6.41.1.3 y	93
6.42 r_type::Level< T > Class Template Reference	93
6.42.1 Constructor & Destructor Documentation	94
6.42.1.1 Level()	94
6.42.1.2 ~Level()	94
6.42.2 Member Function Documentation	94
6.42.2.1 AnimationUpdate()	94
6.42.2.2 CollisionUpdate()	95
6.42.2.3 FireUpdate()	95
6.42.2.4 LevelOne()	96
6.42.2.5 MoveUpdate()	96
6.42.2.6 SetGameParameters()	97
6.42.2.7 SetSystem()	97
6.42.2.8 SpawnEntity()	97
6.42.2.9 Update()	98
6.42.3 Member Data Documentation	98
6.42.3.1 _animationSystem	99
6.42.3.2 _autoFireSystem	99
6.42.3.3 _basicMonsterSpawnTime	99
6.42.3.4 _collisionSystem	99
6.42.3.5 _gameParameters	99
6.42.3.6 _moveSystem	99

6.42.3.7 _shooterEnemySpawnTime . . . . .	100
6.42.3.8 _spawnTimeMonsterThree . . . . .	100
6.43 MovementComponent Struct Reference . . . . .	100
6.43.1 Member Data Documentation . . . . .	100
6.43.1.1 index . . . . .	100
6.43.1.2 movementType . . . . .	100
6.44 MoveSystem Class Reference . . . . .	101
6.44.1 Constructor & Destructor Documentation . . . . .	101
6.44.1.1 MoveSystem() . . . . .	101
6.44.2 Member Function Documentation . . . . .	101
6.44.2.1 moveEntities() . . . . .	101
6.44.3 Member Data Documentation . . . . .	101
6.44.3.1 _componentManager . . . . .	102
6.44.3.2 _entityManager . . . . .	102
6.45 OffsetComponent Struct Reference . . . . .	102
6.45.1 Member Data Documentation . . . . .	102
6.45.1.1 offset . . . . .	102
6.46 OnClickComponent Struct Reference . . . . .	102
6.46.1 Constructor & Destructor Documentation . . . . .	103
6.46.1.1 OnClickComponent() . . . . .	103
6.46.2 Member Data Documentation . . . . .	103
6.46.2.1 isClicked . . . . .	103
6.46.2.2 onClick . . . . .	103
6.47 PlayerComponent Struct Reference . . . . .	103
6.48 playerIdNotFound Class Reference . . . . .	104
6.48.1 Member Function Documentation . . . . .	104
6.48.1.1 what() . . . . .	104
6.49 PlayerMissileComponent Struct Reference . . . . .	104
6.49.1 Member Data Documentation . . . . .	104
6.49.1.1 playerId . . . . .	105
6.50 PositionComponent Struct Reference . . . . .	105
6.50.1 Constructor & Destructor Documentation . . . . .	105
6.50.1.1 PositionComponent() . . . . .	105
6.50.2 Member Data Documentation . . . . .	105
6.50.2.1 x . . . . .	105
6.50.2.2 y . . . . .	106
6.51 PowerUpComponent Struct Reference . . . . .	106
6.52 RectangleShapeComponent Struct Reference . . . . .	106
6.52.1 Constructor & Destructor Documentation . . . . .	106
6.52.1.1 RectangleShapeComponent() . . . . .	106
6.52.2 Member Data Documentation . . . . .	106
6.52.2.1 rectangleShape . . . . .	107

6.53 RenderSystem Class Reference . . . . .	107
6.53.1 Constructor & Destructor Documentation . . . . .	107
6.53.1.1 RenderSystem() . . . . .	107
6.53.2 Member Function Documentation . . . . .	107
6.53.2.1 render() . . . . .	108
6.53.3 Member Data Documentation . . . . .	108
6.53.3.1 _componentManager . . . . .	108
6.53.3.2 _font . . . . .	108
6.53.3.3 _window . . . . .	108
6.54 Scenes Class Reference . . . . .	108
6.54.1 Detailed Description . . . . .	109
6.54.2 Constructor & Destructor Documentation . . . . .	109
6.54.2.1 Scenes() . . . . .	109
6.54.2.2 ~Scenes() . . . . .	110
6.54.3 Member Function Documentation . . . . .	110
6.54.3.1 difficultyChoices() . . . . .	110
6.54.3.2 gameLoop() . . . . .	110
6.54.3.3 getRenderWindow() . . . . .	110
6.54.3.4 HandleMessage() . . . . .	111
6.54.3.5 inGameMenu() . . . . .	111
6.54.3.6 mainMenu() . . . . .	112
6.54.3.7 render() . . . . .	112
6.54.3.8 run() . . . . .	112
6.54.3.9 settingsMenu() . . . . .	113
6.54.3.10 shouldQuit() . . . . .	113
6.54.3.11 StopGameLoop() . . . . .	113
6.54.4 Member Data Documentation . . . . .	113
6.54.4.1 _networkClient . . . . .	113
6.54.4.2 _window . . . . .	114
6.55 ScoreComponent Struct Reference . . . . .	114
6.55.1 Member Data Documentation . . . . .	114
6.55.1.1 score . . . . .	114
6.56 r_type::net::Server Class Reference . . . . .	114
6.56.1 Constructor & Destructor Documentation . . . . .	115
6.56.1.1 Server() . . . . .	115
6.56.1.2 ~Server() . . . . .	115
6.56.2 Member Function Documentation . . . . .	115
6.56.2.1 OnClientConnect() . . . . .	115
6.56.2.2 OnClientDisconnect() . . . . .	116
6.56.2.3 OnMessage() . . . . .	116
6.57 ShaderComponent Struct Reference . . . . .	116
6.57.1 Constructor & Destructor Documentation . . . . .	117

6.57.1.1 ShaderComponent()	117
6.57.2 Member Data Documentation	117
6.57.2.1 shader	117
6.58 ShootComponent Struct Reference	117
6.58.1 Constructor & Destructor Documentation	118
6.58.1.1 ShootComponent()	118
6.58.2 Member Data Documentation	118
6.58.2.1 canShoot	118
6.58.2.2 cooldownTime	118
6.58.2.3 nextShootTime	118
6.59 SpriteComponent Struct Reference	118
6.59.1 Constructor & Destructor Documentation	119
6.59.1.1 SpriteComponent()	119
6.59.2 Member Data Documentation	119
6.59.2.1 hitboxX	119
6.59.2.2 hitboxY	119
6.59.2.3 sprite	119
6.59.2.4 type	120
6.60 SpriteDataComponent Struct Reference	120
6.60.1 Member Data Documentation	120
6.60.1.1 scale	120
6.60.1.2 spritePath	120
6.60.1.3 type	120
6.61 TextComponent Struct Reference	121
6.61.1 Constructor & Destructor Documentation	121
6.61.1.1 TextComponent()	121
6.61.2 Member Data Documentation	121
6.61.2.1 text	121
6.62 TextDataComponent Struct Reference	121
6.62.1 Member Data Documentation	122
6.62.1.1 categoryIds	122
6.62.1.2 categorySize	122
6.62.1.3 categoryTexts	122
6.62.1.4 charSize	122
6.62.1.5 fontPath	122
6.63 TextureManager Class Reference	123
6.63.1 Member Function Documentation	123
6.63.1.1 getTexture()	123
6.63.1.2 releaseTexture()	123
6.63.2 Member Data Documentation	124
6.63.2.1 textures	124
6.64 UIEntityInformation Struct Reference	124

6.64.1 Member Data Documentation	124
6.64.1.1 lives	124
6.64.1.2 score	124
6.64.1.3 spriteData	125
6.64.1.4 textData	125
6.64.1.5 uniqueID	125
6.65 UpdateSystem Class Reference	125
6.65.1 Constructor & Destructor Documentation	125
6.65.1.1 UpdateSystem()	126
6.65.2 Member Function Documentation	126
6.65.2.1 updateSpritePositions()	126
6.65.3 Member Data Documentation	126
6.65.3.1 _componentManager	126
6.65.3.2 _entityManager	126
6.65.3.3 _window	126
6.66 VelocityComponent Struct Reference	126
6.66.1 Member Data Documentation	127
6.66.1.1 x	127
6.66.1.2 y	127
6.67 vf2d Struct Reference	127
6.67.1 Detailed Description	127
6.67.2 Member Data Documentation	127
6.67.2.1 x	128
6.67.2.2 y	128
6.68 WeaponComponent Struct Reference	128
6.68.1 Constructor & Destructor Documentation	128
6.68.1.1 WeaponComponent()	128
6.68.2 Member Data Documentation	128
6.68.2.1 level	128
6.68.2.2 playerId	128
<b>7 File Documentation</b>	<b>129</b>
7.1 /home/runner/work/R-Type/R-Type/Client/Interface/Include/mainmenu.hpp File Reference	129
7.1.1 Function Documentation	129
7.1.1.1 MainMenu()	129
7.2 /home/runner/work/R-Type/R-Type/Client/Interface/Include/Net/a_client.hpp File Reference	129
7.3 /home/runner/work/R-Type/R-Type/Client/Interface/Include/Net/client.hpp File Reference	130
7.4 /home/runner/work/R-Type/R-Type/Client/Interface/Include/Net/i_client.hpp File Reference	130
7.5 /home/runner/work/R-Type/R-Type/Client/Interface/Include/scenes.hpp File Reference	131
7.5.1 Function Documentation	131
7.5.1.1 keyToString()	131
7.6 /home/runner/work/R-Type/R-Type/Client/Src/keyToString.cpp File Reference	131

7.6.1 Function Documentation	131
7.6.1.1 keyToString()	132
7.7 /home/runner/work/R-Type/R-Type/Client/Src/main.cpp File Reference	132
7.7.1 Function Documentation	132
7.7.1.1 isValidIPv4()	132
7.7.1.2 isValidPort()	132
7.7.1.3 main()	132
7.8 /home/runner/work/R-Type/R-Type/Server/Src/main.cpp File Reference	133
7.8.1 Function Documentation	133
7.8.1.1 isValidPort()	133
7.8.1.2 main()	133
7.8.1.3 signal_handler()	133
7.8.2 Variable Documentation	134
7.8.2.1 loopRunning	134
7.9 /home/runner/work/R-Type/R-Type/Client/Src/scenes.cpp File Reference	134
7.9.1 Function Documentation	134
7.9.1.1 createDaltonismChoiceButtons()	135
7.9.1.2 createKeyBindingButtons()	135
7.9.1.3 handleEvents()	135
7.9.1.4 reloadFilter()	136
7.9.1.5 waitForKey()	136
7.10 /home/runner/work/R-Type/R-Type/ECS/Interface/Include/a_scenes.hpp File Reference	136
7.11 /home/runner/work/R-Type/R-Type/ECS/Interface/Include/audio_manager.hpp File Reference	136
7.12 /home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/ally_component.hpp File Reference	136
7.13 /home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/ally_missile_component.hpp File Reference	137
7.14 /home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/animation_component.hpp File Reference	137
7.14.1 Function Documentation	137
7.14.1.1 operator"!="()	137
7.15 /home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/background_component.hpp File Reference	138
7.16 /home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/basic_monster_component.hpp File Reference	138
7.17 /home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/bind_component.hpp File Reference	138
7.18 /home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/boss_component.hpp File Reference	138
7.19 /home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/component_manager.hpp File Reference	139
7.20 /home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/components.hpp File Reference	139
7.21 /home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/enemy_component.hpp File Reference	140



7.22	/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/enemy_missile_component.hpp File Reference	140
7.23	/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/force_missile_component.hpp File Reference	140
7.24	/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/front_component.hpp File Reference	140
7.25	/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/health_component.hpp File Reference	140
7.26	/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/hitbox_component.hpp File Reference	141
7.27	/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/input_component.hpp File Reference	141
7.27.1	Enumeration Type Documentation	141
7.27.1.1	InputType	141
7.28	/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/label_component.hpp File Reference	141
7.29	/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/movement_component.hpp File Reference	142
7.29.1	Enumeration Type Documentation	142
7.29.1.1	MovementType	142
7.30	/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/offset_component.hpp File Reference	142
7.31	/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/on_click_component.hpp File Reference	143
7.32	/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/player_component.hpp File Reference	143
7.33	/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/player_missile_component.hpp File Reference	143
7.34	/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/position_component.hpp File Reference	143
7.35	/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/power_up_component.hpp File Reference	143
7.36	/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/rectangleShapeComponent.hpp File Reference	144
7.37	/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/score_component.hpp File Reference	144
7.38	/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/shader_component.hpp File Reference	144
7.39	/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/shoot_component.hpp File Reference	144
7.40	/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/sprite_component.hpp File Reference	145
7.41	/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/sprite_data_component.hpp File Reference	145
7.41.1	Function Documentation	145
7.41.1.1	operator<<()	145
7.42	/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/text_component.hpp File Reference	146

7.43	/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/text_data_component.hpp File Reference	146
7.44	/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/velocity_component.hpp File Reference	146
7.45	/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/weapon_component.hpp File Reference	146
7.46	/home/runner/work/R-Type/R-Type/ECS/Interface/Include/creatable_client_object.hpp File Reference	146
7.46.1	Enumeration Type Documentation	147
7.46.1.1	CreatableClientObject	147
7.47	/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Entities/entity.hpp File Reference	147
7.48	/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Entities/entity_factory.hpp File Reference	147
7.49	/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Entities/entity_manager.hpp File Reference	148
7.50	/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Entities/i_entity_factory.hpp File Reference	148
7.51	/home/runner/work/R-Type/R-Type/ECS/Interface/Include/entity_struct.hpp File Reference	148
7.52	/home/runner/work/R-Type/R-Type/ECS/Interface/Include/error_handling.hpp File Reference	149
7.53	/home/runner/work/R-Type/R-Type/ECS/Interface/Include/font_manager.hpp File Reference	149
7.54	/home/runner/work/R-Type/R-Type/ECS/Interface/Include/font_path.hpp File Reference	149
7.54.1	Enumeration Type Documentation	150
7.54.1.1	FontPath	150
7.54.2	Function Documentation	150
7.54.2.1	FontFactory()	150
7.54.2.2	operator<<()	150
7.55	/home/runner/work/R-Type/R-Type/ECS/Interface/Include/game_text.hpp File Reference	150
7.55.1	Enumeration Type Documentation	151
7.55.1.1	GameText	151
7.55.2	Function Documentation	151
7.55.2.1	GameTextFactory()	151
7.55.2.2	operator<<()	151
7.56	/home/runner/work/R-Type/R-Type/ECS/Interface/Include/hitbox_tmp.hpp File Reference	151
7.56.1	Function Documentation	152
7.56.1.1	CheckEntityMovement()	152
7.56.1.2	CheckEntityPosition()	152
7.57	/home/runner/work/R-Type/R-Type/ECS/Interface/Include/i_scenes.hpp File Reference	152
7.58	/home/runner/work/R-Type/R-Type/ECS/Interface/Include/macros.hpp File Reference	152
7.58.1	Macro Definition Documentation	153
7.58.1.1	SCREEN_HEIGHT	153
7.58.1.2	SCREEN_WIDTH	153
7.59	/home/runner/work/R-Type/R-Type/ECS/Interface/Include/sound_path.hpp File Reference	153
7.59.1	Enumeration Type Documentation	153
7.59.1.1	ActionType	153
7.59.2	Function Documentation	154
7.59.2.1	SoundFactory()	154
7.60	/home/runner/work/R-Type/R-Type/ECS/Interface/Include/sprite_path.hpp File Reference	154

7.60.1 Enumeration Type Documentation . . . . .	154
7.60.1.1 SpritePath . . . . .	155
7.60.2 Function Documentation . . . . .	155
7.60.2.1 operator<<() . . . . .	155
7.60.2.2 SpriteFactory() . . . . .	155
7.61 /home/runner/work/R-Type/R-Type/ECS/Interface/Include/Systems/animation_system.hpp File Reference . . . . .	156
7.61.1 Enumeration Type Documentation . . . . .	156
7.61.1.1 AnimationBasicMonster . . . . .	156
7.61.1.2 AnimationShip . . . . .	157
7.61.1.3 AnimationWeapon1 . . . . .	157
7.61.2 Function Documentation . . . . .	157
7.61.2.1 animationShipFactory() . . . . .	157
7.61.2.2 operator!=(()) . . . . .	158
7.62 /home/runner/work/R-Type/R-Type/ECS/Interface/Include/Systems/audio_system.hpp File Reference . . . . .	159
7.63 /home/runner/work/R-Type/R-Type/ECS/Interface/Include/Systems/auto_fire_system.hpp File Reference . . . . .	159
7.64 /home/runner/work/R-Type/R-Type/ECS/Interface/Include/Systems/button_system.hpp File Reference . . . . .	159
7.65 /home/runner/work/R-Type/R-Type/ECS/Interface/Include/Systems/collision_system.hpp File Reference . . . . .	159
7.66 /home/runner/work/R-Type/R-Type/ECS/Interface/Include/Systems/i_system.hpp File Reference . . . . .	160
7.67 /home/runner/work/R-Type/R-Type/ECS/Interface/Include/Systems/move_system.hpp File Reference . . . . .	160
7.68 /home/runner/work/R-Type/R-Type/ECS/Interface/Include/Systems/render_system.hpp File Reference . . . . .	160
7.69 /home/runner/work/R-Type/R-Type/ECS/Interface/Include/Systems/systems.hpp File Reference . . . . .	160
7.70 /home/runner/work/R-Type/R-Type/ECS/Interface/Include/Systems/update_system.hpp File Reference . . . . .	161
7.71 /home/runner/work/R-Type/R-Type/ECS/Interface/Include/texture_manager.hpp File Reference . . . . .	161
7.72 /home/runner/work/R-Type/R-Type/ECS/Src/a_scenes.cpp File Reference . . . . .	161
7.73 /home/runner/work/R-Type/R-Type/ECS/Src/Entities/entity_factory.cpp File Reference . . . . .	161
7.73.1 Function Documentation . . . . .	162
7.73.1.1 operator<<() [1/3] . . . . .	162
7.73.1.2 operator<<() [2/3] . . . . .	162
7.73.1.3 operator<<() [3/3] . . . . .	162
7.74 /home/runner/work/R-Type/R-Type/ECS/Src/font_path.cpp File Reference . . . . .	162
7.74.1 Function Documentation . . . . .	162
7.74.1.1 FontFactory() . . . . .	163
7.75 /home/runner/work/R-Type/R-Type/ECS/Src/game_text.cpp File Reference . . . . .	163
7.75.1 Function Documentation . . . . .	163
7.75.1.1 GameTextFactory() . . . . .	163
7.76 /home/runner/work/R-Type/R-Type/ECS/Src/hitbox_tmp.cpp File Reference . . . . .	163
7.76.1 Function Documentation . . . . .	163
7.76.1.1 CheckCollisionLogic() . . . . .	164
7.76.1.2 CheckEntityMovement() . . . . .	164
7.76.1.3 CheckEntityPosition() . . . . .	164

7.77 /home/runner/work/R-Type/R-Type/ECS/Src/sound_path.cpp File Reference . . . . .	164
7.77.1 Function Documentation . . . . .	164
7.77.1.1 SoundFactory() . . . . .	164
7.78 /home/runner/work/R-Type/R-Type/ECS/Src/sprite_path.cpp File Reference . . . . .	165
7.78.1 Function Documentation . . . . .	165
7.78.1.1 SpriteFactory() . . . . .	165
7.79 /home/runner/work/R-Type/R-Type/ECS/Src/Systems/animation_system.cpp File Reference . . . . .	165
7.79.1 Function Documentation . . . . .	165
7.79.1.1 animationBasicMonsterFactory() . . . . .	165
7.79.1.2 animationShipFactory() . . . . .	166
7.79.1.3 animationWeapon1Factory() . . . . .	166
7.79.1.4 operator"!=()" . . . . .	166
7.80 /home/runner/work/R-Type/R-Type/ECS/Src/Systems/audio_system.cpp File Reference . . . . .	167
7.81 /home/runner/work/R-Type/R-Type/ECS/Src/Systems/auto_fire_system.cpp File Reference . . . . .	167
7.82 /home/runner/work/R-Type/R-Type/ECS/Src/Systems/collision_system.cpp File Reference . . . . .	167
7.83 /home/runner/work/R-Type/R-Type/ECS/Src/Systems/move_system.cpp File Reference . . . . .	167
7.84 /home/runner/work/R-Type/R-Type/ECS/Src/Systems/render_system.cpp File Reference . . . . .	167
7.85 /home/runner/work/R-Type/R-Type/ECS/Src/Systems/update_system.cpp File Reference . . . . .	167
7.86 /home/runner/work/R-Type/R-Type/Server/Interface/Include/level.hpp File Reference . . . . .	168
7.87 /home/runner/work/R-Type/R-Type/Server/Interface/Include/Net/a_server.hpp File Reference . . . . .	168
7.88 /home/runner/work/R-Type/R-Type/Server/Interface/Include/Net/server.hpp File Reference . . . . .	169
7.89 /home/runner/work/R-Type/R-Type/Server/Interface/Include/r_type-server.hpp File Reference . . . . .	169
7.90 /home/runner/work/R-Type/R-Type/Server/Src/r_type-server.cpp File Reference . . . . .	169
7.91 /home/runner/work/R-Type/R-Type/Server/Src/server.cpp File Reference . . . . .	169

# 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:

AbstractScenes . . . . .	11
AllyComponent . . . . .	16
AllyMissileComponent . . . . .	16
AnimationComponent . . . . .	16
AudioManager . . . . .	46
BackgroundComponent . . . . .	50
BasicMonsterComponent . . . . .	50
BindComponent . . . . .	50
BossComponent . . . . .	51
ComponentManager . . . . .	56
CreatableClientObject . . . . .	59
EnemyComponent . . . . .	59
EnemyMissileComponent . . . . .	60
Entity . . . . .	60
EntityInformation . . . . .	69
EntityManager . . . . .	70
std::exception	
componentNotFound . . . . .	58
entityNotFound . . . . .	72
failedToLoadFont . . . . .	73
failedToLoadSound . . . . .	73
failedToLoadTexture . . . . .	74
playerIdNotFound . . . . .	104
FontManager . . . . .	75
ForceMissileComponent . . . . .	76
FrontComponent . . . . .	76
HealthComponent . . . . .	77
HitboxComponent . . . . .	78
r_type::net::IClient< T > . . . . .	78
r_type::net::AClient< TypeMessage > . . . . .	11
r_type::net::Client . . . . .	52
r_type::net::AClient< T > . . . . .	11
IEntityFactory . . . . .	81
EntityFactory . . . . .	61
ILevel	

r_type::Level< T > . . . . .	93
InputComponent . . . . .	88
IScenes . . . . .	88
AScenes . . . . .	20
Scenes . . . . .	108
r_type::net::IServer	
r_type::net::AServer< TypeMessage > . . . . .	29
r_type::net::Server . . . . .	114
r_type::net::AServer< T > . . . . .	29
ISystem . . . . .	91
AnimationSystem . . . . .	17
AudioSystem . . . . .	47
AutoFireSystem . . . . .	49
CollisionSystem . . . . .	54
MoveSystem . . . . .	101
RenderSystem . . . . .	107
UpdateSystem . . . . .	125
labelComponent . . . . .	92
MovementComponent . . . . .	100
OffsetComponent . . . . .	102
OnClickComponent . . . . .	102
PlayerComponent . . . . .	103
PlayerMissileComponent . . . . .	104
PositionComponent . . . . .	105
PowerUpComponent . . . . .	106
RectangleShapeComponent . . . . .	106
ScoreComponent . . . . .	114
ShaderComponent . . . . .	116
ShootComponent . . . . .	117
SpriteComponent . . . . .	118
SpriteDataComponent . . . . .	120
TextComponent . . . . .	121
TextDataComponent . . . . .	121
TextureManager . . . . .	123
UIEntityInformation . . . . .	124
VelocityComponent . . . . .	126
vf2d . . . . .	127
WeaponComponent . . . . .	128



## Chapter 3

# Class Index

### 3.1 Class List

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

<a href="#">AbstractScenes</a>	
An abstract class that provides a base for managing different scenes in a game	11
<a href="#">r_type::net::AClient&lt; T &gt;</a>	11
<a href="#">AllyComponent</a>	16
<a href="#">AllyMissileComponent</a>	16
<a href="#">AnimationComponent</a>	16
<a href="#">AnimationSystem</a>	17
<a href="#">AScenes</a>	20
<a href="#">r_type::net::AServer&lt; T &gt;</a>	
AServer class template for managing server operations	29
<a href="#">AudioManager</a>	46
<a href="#">AudioSystem</a>	47
<a href="#">AutoFireSystem</a>	49
<a href="#">BackgroundComponent</a>	50
<a href="#">BasicMonsterComponent</a>	50
<a href="#">BindComponent</a>	50
<a href="#">BossComponent</a>	51
<a href="#">r_type::net::Client</a>	52
<a href="#">CollisionSystem</a>	54
<a href="#">ComponentManager</a>	
Manages the components of entities in an ECS system	56
<a href="#">componentNotFound</a>	
Exception class for when a component is not found	58
<a href="#">CreatableClientObject</a>	
Enum class for the creatable client object	59
<a href="#">EnemyComponent</a>	59
<a href="#">EnemyMissileComponent</a>	60
<a href="#">Entity</a>	
Represents an entity in the ECS system	60
<a href="#">EntityFactory</a>	
A class responsible for creating different types of entities	61
<a href="#">EntityInformation</a>	
Represents information about an entity	69
<a href="#">EntityManager</a>	
Class responsible for managing entities in the ECS system	70

<a href="#">entityNotFound</a>	
Exception class for entity not found error	72
<a href="#">failedToLoadFont</a>	73
<a href="#">failedToLoadSound</a>	73
<a href="#">failedToLoadTexture</a>	
Exception class for failed texture loading	74
<a href="#">FontManager</a>	75
<a href="#">ForceMissileComponent</a>	76
<a href="#">FrontComponent</a>	76
<a href="#">HealthComponent</a>	77
<a href="#">HitboxComponent</a>	78
<a href="#">r_type::net::IClient&lt; T &gt;</a>	78
<a href="#">IEntityFactory</a>	
The interface for an entity factory	81
<a href="#">InputComponent</a>	88
<a href="#">IScenes</a>	
Interface for managing different scenes in a game	88
<a href="#">ISystem</a>	91
<a href="#">labelComponent</a>	92
<a href="#">r_type::Level&lt; T &gt;</a>	93
<a href="#">MovementComponent</a>	100
<a href="#">MoveSystem</a>	101
<a href="#">OffsetComponent</a>	102
<a href="#">OnClickComponent</a>	102
<a href="#">PlayerComponent</a>	103
<a href="#">playerIdNotFound</a>	104
<a href="#">PlayerMissileComponent</a>	104
<a href="#">PositionComponent</a>	105
<a href="#">PowerUpComponent</a>	106
<a href="#">RectangleShapeComponent</a>	106
<a href="#">RenderSystem</a>	107
<a href="#">Scenes</a>	
Represents a class that manages different scenes in a game	108
<a href="#">ScoreComponent</a>	114
<a href="#">r_type::net::Server</a>	114
<a href="#">ShaderComponent</a>	116
<a href="#">ShootComponent</a>	117
<a href="#">SpriteComponent</a>	118
<a href="#">SpriteDataComponent</a>	120
<a href="#">TextComponent</a>	121
<a href="#">TextDataComponent</a>	121
<a href="#">TextureManager</a>	123
<a href="#">UIEntityInformation</a>	124
<a href="#">UpdateSystem</a>	125
<a href="#">VelocityComponent</a>	126
<a href="#">vf2d</a>	
Represents a 2D vector with x and y coordinates	127
<a href="#">WeaponComponent</a>	128

## 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	129
/home/runner/work/R-Type/R-Type/Client/Interface/Include/scenes.hpp	131
/home/runner/work/R-Type/R-Type/Client/Interface/Include/Net/a_client.hpp	129
/home/runner/work/R-Type/R-Type/Client/Interface/Include/Net/client.hpp	130
/home/runner/work/R-Type/R-Type/Client/Interface/Include/Net/i_client.hpp	130
/home/runner/work/R-Type/R-Type/Client/Src/keyToString.cpp	131
/home/runner/work/R-Type/R-Type/Client/Src/main.cpp	132
/home/runner/work/R-Type/R-Type/Client/Src/scenes.cpp	134
/home/runner/work/R-Type/R-Type/ECS/Interface/Include/a_scenes.hpp	136
/home/runner/work/R-Type/R-Type/ECS/Interface/Include/audio_manager.hpp	136
/home/runner/work/R-Type/R-Type/ECS/Interface/Include/creatable_client_object.hpp	146
/home/runner/work/R-Type/R-Type/ECS/Interface/Include/entity_struct.hpp	148
/home/runner/work/R-Type/R-Type/ECS/Interface/Include/error_handling.hpp	149
/home/runner/work/R-Type/R-Type/ECS/Interface/Include/font_manager.hpp	149
/home/runner/work/R-Type/R-Type/ECS/Interface/Include/font_path.hpp	149
/home/runner/work/R-Type/R-Type/ECS/Interface/Include/game_text.hpp	150
/home/runner/work/R-Type/R-Type/ECS/Interface/Include/hitbox_tmp.hpp	151
/home/runner/work/R-Type/R-Type/ECS/Interface/Include/i_scenes.hpp	152
/home/runner/work/R-Type/R-Type/ECS/Interface/Include/macros.hpp	152
/home/runner/work/R-Type/R-Type/ECS/Interface/Include/sound_path.hpp	153
/home/runner/work/R-Type/R-Type/ECS/Interface/Include/sprite_path.hpp	154
/home/runner/work/R-Type/R-Type/ECS/Interface/Include/texture_manager.hpp	161
/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/ally_component.hpp	136
/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/ally_missile_component.hpp	137
/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/animation_component.hpp	137
/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/background_component.hpp	138
/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/basic_monster_component.hpp	138
/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/bind_component.hpp	138
/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/boss_component.hpp	138
/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/component_manager.hpp	139
/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/components.hpp	139
/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/enemy_component.hpp	140
/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/enemy_missile_component.hpp	140
/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/force_missile_component.hpp	140
/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/front_component.hpp	140

/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/health_component.hpp	140
/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/hitbox_component.hpp	141
/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/input_component.hpp	141
/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/label_component.hpp	141
/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/movement_component.hpp	142
/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/offset_component.hpp	142
/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/on_click_component.hpp	143
/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/player_component.hpp	143
/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/player_missile_component.hpp	143
/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/position_component.hpp	143
/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/power_up_component.hpp	143
/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/rectangleShapeComponent.hpp	144
/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/score_component.hpp	144
/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/shader_component.hpp	144
/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/shoot_component.hpp	144
/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/sprite_component.hpp	145
/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/sprite_data_component.hpp	145
/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/text_component.hpp	146
/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/text_data_component.hpp	146
/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/velocity_component.hpp	146
/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/weapon_component.hpp	146
/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Entities/entity.hpp	147
/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Entities/entity_factory.hpp	147
/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Entities/entity_manager.hpp	148
/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Entities/i_entity_factory.hpp	148
/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Systems/animation_system.hpp	156
/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Systems/audio_system.hpp	159
/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Systems/auto_fire_system.hpp	159
/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Systems/button_system.hpp	159
/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Systems/collision_system.hpp	159
/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Systems/i_system.hpp	160
/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Systems/move_system.hpp	160
/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Systems/render_system.hpp	160
/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Systems/systems.hpp	160
/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Systems/update_system.hpp	161
/home/runner/work/R-Type/R-Type/ECS/Src/a_scenes.cpp	161
/home/runner/work/R-Type/R-Type/ECS/Src/font_path.cpp	162
/home/runner/work/R-Type/R-Type/ECS/Src/game_text.cpp	163
/home/runner/work/R-Type/R-Type/ECS/Src/hitbox_tmp.cpp	163
/home/runner/work/R-Type/R-Type/ECS/Src/sound_path.cpp	164
/home/runner/work/R-Type/R-Type/ECS/Src/sprite_path.cpp	165
/home/runner/work/R-Type/R-Type/ECS/Src/Entities/entity_factory.cpp	161
/home/runner/work/R-Type/R-Type/ECS/Src/Systems/animation_system.cpp	165
/home/runner/work/R-Type/R-Type/ECS/Src/Systems/audio_system.cpp	167
/home/runner/work/R-Type/R-Type/ECS/Src/Systems/auto_fire_system.cpp	167
/home/runner/work/R-Type/R-Type/ECS/Src/Systems/collision_system.cpp	167
/home/runner/work/R-Type/R-Type/ECS/Src/Systems/move_system.cpp	167
/home/runner/work/R-Type/R-Type/ECS/Src/Systems/render_system.cpp	167
/home/runner/work/R-Type/R-Type/ECS/Src/Systems/update_system.cpp	167
/home/runner/work/R-Type/R-Type/Server/Interface/Include/level.hpp	168
/home/runner/work/R-Type/R-Type/Server/Interface/Include/r_type-server.hpp	169
/home/runner/work/R-Type/R-Type/Server/Interface/Include/Net/a_server.hpp	168
/home/runner/work/R-Type/R-Type/Server/Interface/Include/Net/server.hpp	169
/home/runner/work/R-Type/R-Type/Server/Src/main.cpp	133
/home/runner/work/R-Type/R-Type/Server/Src/r_type-server.cpp	169
/home/runner/work/R-Type/R-Type/Server/Src/server.cpp	169

## Chapter 5

# Namespace Documentation

### 5.1 `r_type` Namespace Reference

#### Namespaces

- [net](#)

#### Classes

- class [Level](#)

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

#### Classes

- class [AClient](#)
- class [Client](#)
- class [IClient](#)
- class [AServer](#)  
*[AServer](#) class template for managing server operations.*
- class [Server](#)



## Chapter 6

# Class Documentation

### 6.1 AbstractScenes Class Reference

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

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

#### 6.1.1 Detailed Description

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

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

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

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

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

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

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



## Public Member Functions

- [AClient](#) ()
- virtual [~AClient](#) ()
- bool [Connect](#) (const std::string &host, const uint16\_t port)  
*Connects to a remote host using UDP protocol.*
- void [Disconnect](#) ()  
*Disconnects the client from the server.*
- bool [IsConnected](#) ()  
*Checks if the client is connected to the server.*
- void [Send](#) (const Message< T > &msg)  
*Send message to server.*
- ThreadSafeQueue< OwnedMessage< T > > & [Incoming](#) ()  
*get incoming messages*
- const std::unique\_ptr< Connection< T > > & [getConnection](#) ()
- void [setPlayerId](#) (int id)
- uint32\_t [getPlayerId](#) ()
- void [setWindowSize](#) (sf::Vector2u size)
- sf::Vector2u [getWindowSize](#) ()

## 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
- sf::Vector2u [windowSize](#)

## 6.2.1 Constructor & Destructor Documentation

### 6.2.1.1 AClient()

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

### 6.2.1.2 ~AClient()

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



## 6.2.2 Member Function Documentation

### 6.2.2.1 `Connect()`

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

Connects to a remote host using UDP protocol.

#### Parameters

<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 >`.

### 6.2.2.2 `Disconnect()`

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

Disconnects the client from the server.

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

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

### 6.2.2.3 `getConnection()`

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

#### 6.2.2.4 getPlayerId()

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

#### 6.2.2.5 getWindowSize()

```
template<typename T >
sf::Vector2u r_type::net::AClient< T >::getWindowSize ( ) [inline]
```

#### 6.2.2.6 Incoming()

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

get incoming messages

##### Returns

ThreadSafeQueue<OwnedMessage<T>>&

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

#### 6.2.2.7 isConnected()

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

Checks if the client is connected to the server.

##### Returns

true

false

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

#### 6.2.2.8 Send()

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

Send message to server.

## Parameters

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

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

### 6.2.2.9 `setPlayerId()`

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

### 6.2.2.10 `setWindowSize()`

```
template<typename T >
void r_type::net::AClient< T >::setWindowSize (
    sf::Vector2u size ) [inline]
```

## 6.2.3 Member Data Documentation

### 6.2.3.1 `m_connection`

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

### 6.2.3.2 `m_context`

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

### 6.2.3.3 `m_qMessagesIn`

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

#### 6.2.3.4 playerId

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

#### 6.2.3.5 thrContext

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

#### 6.2.3.6 windowSize

```
template<typename T >
sf::Vector2u r_type::net::AClient< T >::windowSize [private]
```

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

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

### 6.3 AllyComponent Struct Reference

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

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

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

### 6.4 AllyMissileComponent Struct Reference

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

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

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

### 6.5 AnimationComponent Struct Reference

```
#include <animation_component.hpp>
```

## Public Member Functions

- [AnimationComponent](#) ([vf2d \\_offset](#), [vf2d \\_dimension](#))

## Public Attributes

- [vf2d offset](#)
- [vf2d dimension](#)

## 6.5.1 Constructor & Destructor Documentation

### 6.5.1.1 AnimationComponent()

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

## 6.5.2 Member Data Documentation

### 6.5.2.1 dimension

[vf2d](#) `AnimationComponent::dimension`

### 6.5.2.2 offset

[vf2d](#) `AnimationComponent::offset`

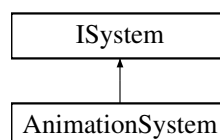
The documentation for this struct was generated from the following file:

- `/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/animation\_component.hpp`

## 6.6 AnimationSystem Class Reference

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

Inheritance diagram for AnimationSystem:



## Public Member Functions

- [AnimationSystem](#) ([ComponentManager](#) &componentManager, [EntityManager](#) &entityManager)
- void [AnimationEntities](#) ([ComponentManager](#) &componentManager, [EntityManager](#) &entityManager, float deltaTime)  
*Animates entities.*
- void [animatePlayer](#) (std::optional< [VelocityComponent](#) \* > &velocity, std::optional< [AnimationComponent](#) \* > &animation)
- void [animateBasicMonster](#) (std::optional< [AnimationComponent](#) \* > &animation)
- void [animateWeapon](#) (std::optional< [AnimationComponent](#) \* > &animation)

## Private Attributes

- [ComponentManager](#) &\_componentManager  
*Reference to the [ComponentManager](#) instance.*
- [EntityManager](#) &\_entityManager  
*Reference to the [EntityManager](#) instance.*

## 6.6.1 Constructor & Destructor Documentation

### 6.6.1.1 AnimationSystem()

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

## 6.6.2 Member Function Documentation

### 6.6.2.1 animateBasicMonster()

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

### 6.6.2.2 animatePlayer()

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

### 6.6.2.3 animateWeapon()

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

### 6.6.2.4 AnimationEntities()

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

Animates entities.

Updates the animation states of entities based on their components.

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

#### Parameters

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

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

#### Parameters

<i>componentManager</i>	Reference to the <a href="#">ComponentManager</a> that handles components.
<i>entityManager</i>	Reference to the <a href="#">EntityManager</a> that handles entities.
<i>deltaTime</i>	The time elapsed since the last update, used for time-based animations.

## 6.6.3 Member Data Documentation

### 6.6.3.1 \_componentManager

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

Reference to the [ComponentManager](#) instance.

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

### 6.6.3.2 \_entityManager

`EntityManager& AnimationSystem::_entityManager [private]`

Reference to the [EntityManager](#) instance.

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

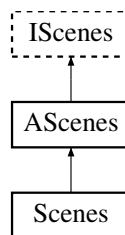
The documentation for this class was generated from the following files:

- /home/runner/work/R-Type/R-Type/ECS/Interface/Include/Systems/[animation\\_system.hpp](#)
- /home/runner/work/R-Type/R-Type/ECS/Src/Systems/[animation\\_system.cpp](#)

## 6.7 AScenes Class Reference

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

Inheritance diagram for AScenes:



### Public Types

- enum class [Scene](#) {  
[MAIN\\_MENU](#) , [GAME\\_LOOP](#) , [SETTINGS\\_MENU](#) , [IN\\_GAME\\_MENU](#) ,  
[EXIT](#) }  
*Represents the different scenes in the R-Type client application.*
- enum class [GameMode](#) { [EASY](#) , [MEDIUM](#) , [HARD](#) }  
*Enumeration to represent different game difficulty levels.*
- enum class [DaltonismMode](#) { [NORMAL](#) , [TRITANOPIA](#) , [DEUTERANOPIA](#) , [PROTANOPIA](#) }  
*Enum representing different modes of color blindness (Daltonism).*
- enum class [Actions](#) {  
[UP](#) , [DOWN](#) , [LEFT](#) , [RIGHT](#) ,  
[FIRE](#) , [PAUSE](#) , [QUIT](#) }  
*Enumeration representing possible actions in the game.*
- enum class [SpriteType](#) {  
[BACKGROUND](#) , [PLAYER](#) , [ALLY](#) , [ENEMY](#) ,  
[FILTER](#) , [WEAPON](#) , [POWER\\_UP](#) , [UI](#) ,  
[OTHER](#) }  
*Enumeration representing the type of sprite in the game.*



## Public Member Functions

- [AScenes](#) (std::string ip, int port)
- [~AScenes](#) ()=default
- void [setScene](#) ([Scene](#) scene)  
*Set the Scene object.*
- [AScenes::Scene](#) [getPreviousScene](#) ()  
*Get the Previous Scene object.*
- [DaltonismMode](#) [getDaltonism](#) () const  
*Get the Daltonism object.*
- void [setDaltonism](#) ([DaltonismMode](#) const mode)  
*Set the Daltonism object.*
- void [setGameMode](#) ([GameMode](#) const mode)  
*Set the Game Mode object.*
- void [setDisplayDaltonismChoice](#) (bool const displayDaltonismChoice)
- bool [getDisplayDaltonismChoice](#) () const
- void [setDisplayGameModeChoice](#) (bool const displayGameModeChoice)
- bool [getDisplayGameModeChoice](#) () const
- void [setDisplayKeyBindsChoice](#) (bool const displayKeyBindsChoice)
- bool [getDisplayKeyBindsChoice](#) () const
- void [setIp](#) (std::string ip)
- void [setPort](#) (int port)
- std::string [getIp](#) () const
- int [getPort](#) () const

## Public Attributes

- std::map< [Actions](#), sf::Keyboard::Key > [keyBinds](#)  
*A map that binds game actions to specific keyboard keys.*
- std::vector< std::shared\_ptr< [Entity](#) > > [buttons](#)
- std::shared\_ptr< [Entity](#) > [filter](#)

## Protected Attributes

- [GameMode](#) [\\_currentGameMode](#) = [GameMode::MEDIUM](#)
- [DaltonismMode](#) [\\_currentDaltonismMode](#) = [DaltonismMode::NORMAL](#)
- [Scene](#) [\\_currentScene](#) = [Scene::MAIN\\_MENU](#)
- [Scene](#) [\\_previousScene](#) = [Scene::MAIN\\_MENU](#)
- bool [\\_displayDaltonismChoice](#) = false
- bool [\\_displayGameModeChoice](#) = false
- bool [\\_displayKeyBindsChoice](#) = false
- std::string [\\_ip](#)  
*The IP address of the server.*
- int [\\_port](#)  
*The port number of the server.*

### 6.7.1 Member Enumeration Documentation

### 6.7.1.1 Actions

```
enum AScenes::Actions [strong]
```

Enumeration representing possible actions in the game.

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

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

#### Enumerator

UP	
DOWN	
LEFT	
RIGHT	
FIRE	
PAUSE	
QUIT	

### 6.7.1.2 DaltonismMode

```
enum AScenes::DaltonismMode [strong]
```

Enum representing different modes of color blindness (Daltonism).

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

#### Enumerator

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

### 6.7.1.3 GameMode

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

Enumeration to represent different game difficulty levels.

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

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

#### Enumerator

EASY	
MEDIUM	
HARD	

### 6.7.1.4 Scene

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

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

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

#### Enumerator

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

### 6.7.1.5 SpriteType

```
enum AScenes::SpriteType [strong]
```

Enumeration representing the type of sprite in the game.

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

- BACKGROUND: Represents a background sprite.

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

#### Enumerator

BACKGROUND	
PLAYER	
ALLY	
ENEMY	
FILTER	
WEAPON	
POWER_UP	
UI	
OTHER	

## 6.7.2 Constructor & Destructor Documentation

### 6.7.2.1 AScenes()

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

### 6.7.2.2 ~AScenes()

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

## 6.7.3 Member Function Documentation

### 6.7.3.1 getDaltonism()

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

Get the Daltonism object.

#### Returns

DaltonismMode

### 6.7.3.2 getDisplayDaltonismChoice()

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

### 6.7.3.3 getDisplayGameModeChoice()

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

### 6.7.3.4 getDisplayKeyBindsChoice()

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

### 6.7.3.5 getIp()

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

### 6.7.3.6 getPort()

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

### 6.7.3.7 getPreviousScene()

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

Get the Previous Scene object.

#### Returns

Scene

### 6.7.3.8 setDaltonism()

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

Set the Daltonism object.

**Parameters**

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

**6.7.3.9 setDisplayDaltonismChoice()**

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

**6.7.3.10 setDisplayGameModeChoice()**

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

**6.7.3.11 setDisplayKeyBindsChoice()**

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

**6.7.3.12 setGameMode()**

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

Set the Game Mode object.

**Parameters**

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

**6.7.3.13 setIp()**

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

#### 6.7.3.14 setPort()

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

#### 6.7.3.15 setScene()

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

Set the Scene object.

Parameters

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

### 6.7.4 Member Data Documentation

#### 6.7.4.1 \_currentDaltonismMode

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

#### 6.7.4.2 \_currentGameMode

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

#### 6.7.4.3 \_currentScene

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

#### 6.7.4.4 \_displayDaltonismChoice

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

#### 6.7.4.5 \_displayGameModeChoice

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

#### 6.7.4.6 \_displayKeyBindsChoice

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

#### 6.7.4.7 \_ip

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

The IP address of the server.

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

#### 6.7.4.8 \_port

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

The port number of the server.

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

#### 6.7.4.9 \_previousScene

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

#### 6.7.4.10 buttons

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

#### 6.7.4.11 filter

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



## 6.7.4.12 keyBinds

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

**Initial value:**

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

A map that binds game actions to specific keyboard keys.

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

The key bindings are as follows:

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

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

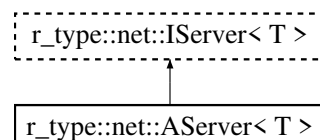
- /home/runner/work/R-Type/R-Type/ECS/Interface/Include/a\_scenes.hpp
- /home/runner/work/R-Type/R-Type/ECS/Src/a\_scenes.cpp

## 6.8 r\_type::net::AServer&lt; T &gt; Class Template Reference

[AServer](#) class template for managing server operations.

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

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



## Public Member Functions

- [AServer](#) (uint16\_t port)  
*Constructs an [AServer](#) object with the specified port.*
- [~AServer](#) ()  
*Destructor for the [AServer](#) class.*
- bool [Start](#) ()  
*Start the server.*
- void [Stop](#) ()  
*Stops the server.*
- void [WaitForClientMessage](#) ()  
*Waits for a client message asynchronously.*
- void [MessageClient](#) (std::shared\_ptr< Connection< T >> client, const Message< T > &msg)  
*Sends a message to a specific client if the client is connected.*
- void [MessageAllClients](#) (const Message< T > &msg, std::shared\_ptr< Connection< T >> plgnore← Client=nullptr)  
*Sends a message to all connected clients, optionally ignoring a specified client.*
- [UIEntityInformation UpdateInfoBar](#) (int playerId)
- void [Update](#) (size\_t nMaxMessages=-1, bool bWait=false)  
*Updates the server state, processes incoming messages, and updates the game level.*
- void [UpdatePlayerPosition](#) (PlayerMovement direction, uint32\_t entityId) override  
*Updates the position of an entity based on the message received and the client ID.*
- uint32\_t [GetClientPlayerId](#) (uint32\_t id)  
*Retrieves the entity ID associated with a client ID.*
- uint32\_t [GetPlayerClientId](#) (uint32\_t id)
- uint32\_t [GetClientInfoBarId](#) (uint32\_t id)
- void [RemovePlayer](#) (uint32\_t id)  
*Removes a player from the game based on the client ID.*
- void [RemoveEntity](#) (uint32\_t id)  
*Removes entities associated with a player.*
- void [RemoveInfoBar](#) (uint32\_t infoBarId)
- [EntityInformation InitiatePlayer](#) (int clientId)  
*Initializes a new player entity and assigns a random position.*
- [UIEntityInformation InitInfoBar](#) (int clientId)
- [EntityInformation FormatEntityInformation](#) (uint32\_t entityId)  
*Formats the information of a given entity into an [EntityInformation](#) structure.*
- [EntityInformation InitiatePlayerMissile](#) (int entityId)  
*Initializes a missile entity associated with a player.*
- [EntityInformation InitiateEnemyMissile](#) (int enemyId)
- [EntityInformation InitiateWeaponForce](#) (int entityId)
- [EntityInformation InitiateBackground](#) ()  
*Initializes a background entity.*
- std::shared\_ptr< Connection< T >> [getClientById](#) (const std::deque< std::shared\_ptr< Connection< T >>> &connections, uint32\_t clientId)
- virtual void [OnClientValidated](#) (std::shared\_ptr< Connection< T >> client)  
*Callback function that is called when a client has been successfully validated.*
- [ComponentManager GetComponentManager](#) () override  
*Retrieves the component manager associated with the server.*
- [EntityManager & GetEntityManager](#) () override  
*Retrieves the entity manager associated with the server.*
- [EntityFactory & GetEntityFactory](#) () override  
*Retrieves the entity factory associated with the server.*

- `std::chrono::system_clock::time_point` [GetClock](#) () override  
*Retrieves the current clock time of the server.*
- `void` [SetClock](#) (`std::chrono::system_clock::time_point` clock)  
*Set the Clock object.*

## Public Attributes

- `ThreadSafeQueue< OwnedMessage< T > >` [\\_qMessagesIn](#)  
*Thread-safe queue to store incoming messages.*
- `std::deque< std::shared_ptr< Connection< T > > >` [\\_deqConnections](#)  
*A deque that holds shared pointers to Connection objects.*
- `asio::io_context` [\\_asioContext](#)  
*The io\_context object provides I/O services, such as sockets, that the server will use.*
- `std::thread` [\\_threadContext](#)  
*Thread object for managing the server's context operations.*
- `asio::ip::udp::socket` [\\_asioSocket](#)  
*A socket for sending and receiving UDP datagrams.*
- `asio::ip::udp::endpoint` [\\_clientEndpoint](#)  
*Represents the endpoint of a client in a UDP connection.*
- `std::array< uint8_t, 1024 >` [\\_tempBuffer](#)  
*Temporary buffer used for storing data.*
- `uint32_t` [\\_nIDCounter](#) = 10000  
*Counter for generating unique network IDs.*
- [ComponentManager](#) [\\_componentManager](#)  
*Manages and maintains the lifecycle of various components within the server.*
- [EntityManager](#) [\\_entityManager](#)  
*Manages the lifecycle and operations of entities within the server.*
- [EntityFactory](#) [\\_entityFactory](#)  
*An instance of [EntityFactory](#) used to create and manage game entities.*
- `std::unordered_map< uint32_t, uint32_t >` [\\_clientPlayerID](#)  
*A container that maps client IDs to player IDs.*
- `std::unordered_map< uint32_t, uint32_t >` [\\_clientInfoBarID](#)
- `int` [\\_nbrOfPlayers](#) = 0  
*Number of players currently connected to the server.*
- `std::chrono::system_clock::time_point` [\\_clock](#) = `std::chrono::system_clock::now()`  
*Stores the current time point from the system clock.*
- `bool` [\\_playerConnected](#) = false
- [EntityInformation](#) [\\_background](#)  
*Holds information about the background entity.*
- `int` [\\_port](#)
- `r_type::Level< T >` [\\_level](#)

## Protected Member Functions

- `virtual bool` [OnClientConnect](#) (`std::shared_ptr< Connection< T > >` client)  
*on client connect event*
- `virtual void` [OnClientDisconnect](#) (`std::shared_ptr< Connection< T > >` client)  
*on client disconnect event*
- `virtual void` [OnMessage](#) (`std::shared_ptr< Connection< T > >` client, `Message< T >` &msg)  
*on message event*

### 6.8.1 Detailed Description

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

[AServer](#) class template for managing server operations.

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

#### Template Parameters

<i>T</i>	The type of data that the server handles.
----------	---

### 6.8.2 Constructor & Destructor Documentation

#### 6.8.2.1 AServer()

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

Constructs an [AServer](#) object with the specified port.

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

#### Parameters

<i>port</i>	The port number on which the server will listen for incoming connections.
-------------	---

#### 6.8.2.2 ~AServer()

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

Destructor for the [AServer](#) class.

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

### 6.8.3 Member Function Documentation

#### 6.8.3.1 `FormatEntityInformation()`

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

Formats the information of a given entity into an [EntityInformation](#) structure.

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

##### Parameters

<i>entity</i>	The entity whose information is to be formatted.
---------------	--

##### Returns

[EntityInformation](#) The formatted information of the entity.

#### 6.8.3.2 `getClientById()`

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

#### 6.8.3.3 `GetClientInfoBarId()`

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

#### 6.8.3.4 `GetClientPlayerId()`

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

Retrieves the entity ID associated with a client ID.

#### Parameters

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

#### Returns

uint32\_t The entity ID associated with the client.

### 6.8.3.5 GetClock()

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

Retrieves the current clock time of the server.

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

#### Returns

std::chrono::system\_clock::time\_point The current time point of the server's clock.

### 6.8.3.6 GetComponentManager()

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

Retrieves the component manager associated with the server.

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

#### Returns

[ComponentManager](#)& A reference to the component manager instance.

### 6.8.3.7 GetEntityFactory()

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

Retrieves the entity factory associated with the server.

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

#### Returns

[EntityFactory](#)& A reference to the entity factory instance.

### 6.8.3.8 GetEntityManager()

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

Retrieves the entity manager associated with the server.

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

#### Returns

[EntityManager](#)& A reference to the entity manager instance.

### 6.8.3.9 GetPlayerClientId()

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

### 6.8.3.10 InitiateBackground()

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

Initializes a background entity.

The function creates and returns information about the background entity.

#### Returns

[EntityInformation](#) The information of the background entity.

### 6.8.3.11 InitiateEnemyMissile()

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

### 6.8.3.12 InitiatePlayer()

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

Initializes a new player entity and assigns a random position.

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

#### Parameters

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

#### Returns

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

### 6.8.3.13 InitiatePlayerMissile()

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

Initializes a missile entity associated with a player.

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

#### Parameters

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

#### Returns

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



**6.8.3.14 InitiateWeaponForce()**

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

**6.8.3.15 InitInfoBar()**

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

**6.8.3.16 MessageAllClients()**

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

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

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

**Template Parameters**

<i>T</i>	The type of the message.
----------	--------------------------

**Parameters**

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

**6.8.3.17 MessageClient()**

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

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

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

### Template Parameters

<i>T</i>	The type of the message.
----------	--------------------------

### Parameters

<i>client</i>	A shared pointer to the client connection.
<i>msg</i>	The message to be sent to the client.

#### 6.8.3.18 OnClientConnect()

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

on client connect event

### Parameters

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

### Returns

true

false

#### 6.8.3.19 OnClientDisconnect()

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

on client disconnect event

### Parameters

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

#### 6.8.3.20 OnClientValidated()

```
template<typename T >
```

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

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

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

#### Parameters

<i>client</i>	A shared pointer to the validated client connection.
---------------	--

### 6.8.3.21 OnMessage()

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

on message event

#### Parameters

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

### 6.8.3.22 RemoveEntity()

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

Removes entities associated with a player.

#### Parameters

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

### 6.8.3.23 RemoveInfoBar()

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

#### 6.8.3.24 RemovePlayer()

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

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

##### Parameters

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

#### 6.8.3.25 SetClock()

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

Set the Clock object.

##### Parameters

<i>clock</i>	
--------------	--

#### 6.8.3.26 Start()

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

Start the server.

##### Returns

true  
false

#### 6.8.3.27 Stop()

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

Stops the server.

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

### 6.8.3.28 `Update()`

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

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

This function performs several tasks:

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

#### Parameters

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

### 6.8.3.29 `UpdateInfoBar()`

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

### 6.8.3.30 `UpdatePlayerPosition()`

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

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

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

## Parameters

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

**6.8.3.31 WaitForClientMessage()**

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

Waits for a client message asynchronously.

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

**6.8.4 Member Data Documentation****6.8.4.1 \_asioContext**

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

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

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

**6.8.4.2 \_asioSocket**

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

A socket for sending and receiving UDP datagrams.

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

#### 6.8.4.3 `_background`

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

Holds information about the background entity.

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

#### 6.8.4.4 `_clientEndpoint`

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

Represents the endpoint of a client in a UDP connection.

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

#### 6.8.4.5 `_clientInfoBarID`

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

#### 6.8.4.6 `_clientPlayerID`

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

A container that maps client IDs to player IDs.

left: client ID right: player ID

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

#### 6.8.4.7 `_clock`

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

Stores the current time point from the system clock.

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

#### 6.8.4.8 `_componentManager`

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

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

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

#### 6.8.4.9 `_deqConnections`

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

A deque that holds shared pointers to Connection objects.

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

##### Template Parameters

<code>T</code>	The type of data that the Connection handles.
----------------	---

#### 6.8.4.10 `_entityFactory`

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

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

#### 6.8.4.11 `_entityManager`

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

Manages the lifecycle and operations of entities within the server.

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

#### 6.8.4.12 `_level`

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



#### 6.8.4.13 `_nbrOfPlayers`

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

Number of players currently connected to the server.

#### 6.8.4.14 `_nIDCounter`

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

Counter for generating unique network IDs.

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

#### 6.8.4.15 `_playerConnected`

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

#### 6.8.4.16 `_port`

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

#### 6.8.4.17 `_qMessagesIn`

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

Thread-safe queue to store incoming messages.

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

#### 6.8.4.18 `_tempBuffer`

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

Temporary buffer used for storing data.

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

#### 6.8.4.19 `_threadContext`

```
template<typename T >
std::thread r\_type::net::AServer< T >::_threadContext
```

Thread object for managing the server's context operations.

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

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

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

## 6.9 AudioManager Class Reference

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

### Public Member Functions

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

### Private Attributes

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

### 6.9.1 Member Function Documentation

#### 6.9.1.1 `getSoundBuffer()`

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

### 6.9.2 Member Data Documentation

### 6.9.2.1 soundBuffers

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

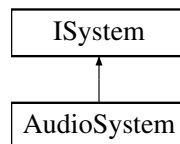
The documentation for this class was generated from the following file:

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

## 6.10 AudioSystem Class Reference

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

Inheritance diagram for AudioSystem:



### Public Member Functions

- [AudioSystem](#) (std::shared\_ptr< [AudioManager](#) > audioManager)
- void [playBackgroundMusic](#) (const std::string &filePath)
- void [stopBackgroundMusic](#) ()
- void [playSoundEffect](#) (const std::string &filePath)

### Private Attributes

- std::shared\_ptr< [AudioManager](#) > [\\_audioManager](#)
- sf::Music [\\_backgroundMusic](#)
- std::string [\\_currentMusicFilePath](#)
- sf::Sound [\\_soundEffect](#)

## 6.10.1 Constructor & Destructor Documentation

### 6.10.1.1 AudioSystem()

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

## 6.10.2 Member Function Documentation

### 6.10.2.1 playBackgroundMusic()

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

### 6.10.2.2 playSoundEffect()

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

### 6.10.2.3 stopBackgroundMusic()

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

## 6.10.3 Member Data Documentation

### 6.10.3.1 \_audioManager

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

### 6.10.3.2 \_backgroundMusic

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

### 6.10.3.3 \_currentMusicFilePath

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

### 6.10.3.4 \_soundEffect

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

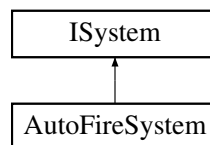
The documentation for this class was generated from the following files:

- /home/runner/work/R-Type/R-Type/ECS/Interface/Include/Systems/[audio\\_system.hpp](#)
- /home/runner/work/R-Type/R-Type/ECS/Src/Systems/[audio\\_system.cpp](#)

## 6.11 AutoFireSystem Class Reference

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

Inheritance diagram for AutoFireSystem:



### Public Member Functions

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

### Private Attributes

- [ComponentManager](#) & [\\_componentManager](#)
- [EntityManager](#) & [\\_entityManager](#)

### 6.11.1 Constructor & Destructor Documentation

#### 6.11.1.1 AutoFireSystem()

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

### 6.11.2 Member Function Documentation

### 6.11.2.1 handleAutoFire()

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

## 6.11.3 Member Data Documentation

### 6.11.3.1 \_componentManager

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

### 6.11.3.2 \_entityManager

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

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

- /home/runner/work/R-Type/R-Type/ECS/Interface/Include/Systems/[auto\\_fire\\_system.hpp](#)
- /home/runner/work/R-Type/R-Type/ECS/Src/Systems/[auto\\_fire\\_system.cpp](#)

## 6.12 BackgroundComponent Struct Reference

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

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

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

## 6.13 BasicMonsterComponent Struct Reference

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

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

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

## 6.14 BindComponent Struct Reference

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

## Public Member Functions

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

## Public Attributes

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

### 6.14.1 Constructor & Destructor Documentation

#### 6.14.1.1 BindComponent()

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

### 6.14.2 Member Data Documentation

#### 6.14.2.1 bind

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

#### 6.14.2.2 isHovered

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

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

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

## 6.15 BossComponent Struct Reference

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

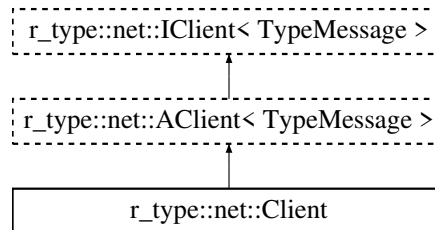
The documentation for this struct was generated from the following file:

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

## 6.16 r\_type::net::Client Class Reference

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

Inheritance diagram for r\_type::net::Client:



### Public Member Functions

- void [PingServer](#) ()  
*Send a message to the server to get the ping.*
- void [MessageAll](#) ()  
*Send a message to the server to all other clients.*
- sf::Vector2u [initInfoBar](#) (UIEntityInformation entity, ComponentManager &componentManager, TextureManager &textureManager, FontManager &fontManager, sf::Vector2u windowSize)
- void [updateInfoBar](#) (UIEntityInformation entity, ComponentManager &componentManager, TextureManager &textureManager)
- void [addEntity](#) (EntityInformation entity, ComponentManager &componentManager, TextureManager &textureManager, sf::Vector2u windowSize)
- void [removeEntity](#) (int entityId, ComponentManager &componentManager)
- void [moveEntity](#) (uint32\_t id, vf2d newPos, ComponentManager &componentManager, sf::Vector2u windowSize)
- void [animateEntity](#) (int entityId, AnimationComponent rect, ComponentManager &componentManager)

### Additional Inherited Members

#### 6.16.1 Member Function Documentation

##### 6.16.1.1 addEntity()

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



### 6.16.1.2 animateEntity()

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

### 6.16.1.3 initInfoBar()

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

### 6.16.1.4 MessageAll()

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

Send a message to the server to all other clients.

### 6.16.1.5 moveEntity()

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

### 6.16.1.6 PingServer()

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

Send a message to the server to get the ping.

### 6.16.1.7 removeEntity()

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

### 6.16.1.8 updateInfoBar()

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

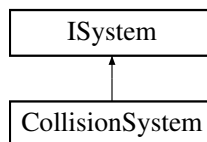
The documentation for this class was generated from the following file:

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

## 6.17 CollisionSystem Class Reference

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

Inheritance diagram for CollisionSystem:



### Public Member Functions

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

### Private Attributes

- [ComponentManager](#) & [\\_componentManager](#)
- [EntityManager](#) & [\\_entityManager](#)

### 6.17.1 Constructor & Destructor Documentation

### 6.17.1.1 CollisionSystem()

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

## 6.17.2 Member Function Documentation

### 6.17.2.1 checkCollision()

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

### 6.17.2.2 checkOffScreen()

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

## 6.17.3 Member Data Documentation

### 6.17.3.1 \_componentManager

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

### 6.17.3.2 \_entityManager

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

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

- [/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Systems/collision\\_system.hpp](#)
- [/home/runner/work/R-Type/R-Type/ECS/Src/Systems/collision\\_system.cpp](#)

## 6.18 ComponentManager Class Reference

Manages the components of entities in an ECS system.

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

### Public Member Functions

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

### Private Attributes

- `std::unordered_map< std::type_index, std::unordered_map< int, std::any > > components`  
*A component manager that stores components in an unordered map.*

#### 6.18.1 Detailed Description

Manages the components of entities in an ECS system.

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

#### 6.18.2 Member Function Documentation

##### 6.18.2.1 [addComponent\(\)](#)

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

Adds a component to an entity.

## Template Parameters

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

## 6.18.2.2 GetComponent()

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

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

## Template Parameters

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

## Parameters

<i>entity↵ Id</i>	The ID of the entity.
-----------------------	-----------------------

## Returns

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

## 6.18.2.3 GetComponentMap()

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

Retrieves the component map for the specified component type.

## Template Parameters

<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`.

**6.18.2.4 removeEntityFromAllComponents()**

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

**6.18.2.5 removeEntityFromComponent()**

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

**6.18.3 Member Data Documentation****6.18.3.1 components**

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

A component manager that stores components in an unordered map.

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

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

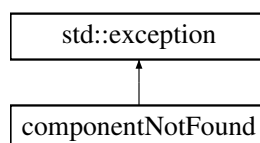
- [/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/component\\_manager.hpp](/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/component_manager.hpp)

**6.19 componentNotFound Class Reference**

Exception class for when a component is not found.

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

Inheritance diagram for `componentNotFound`:



## Private Member Functions

- `const char * what () const` noexcept override

### 6.19.1 Detailed Description

Exception class for when a component is not found.

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

### 6.19.2 Member Function Documentation

#### 6.19.2.1 what()

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

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

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

## 6.20 CreatableClientObject Class Reference

Enum class for the creatable client object.

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

### 6.20.1 Detailed Description

Enum class for the creatable client object.

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

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

## 6.21 EnemyComponent Struct Reference

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

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

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

## 6.22 EnemyMissileComponent Struct Reference

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

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

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

## 6.23 Entity Class Reference

Represents an entity in the ECS system.

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

### Public Member Functions

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

### Private Attributes

- int [\\_id](#)

### 6.23.1 Detailed Description

Represents an entity in the ECS system.

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

### 6.23.2 Constructor & Destructor Documentation

#### 6.23.2.1 Entity()

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

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



## Parameters

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

### 6.23.3 Member Function Documentation

#### 6.23.3.1 getId()

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

Returns the ID of the entity.

## Returns

The ID of the entity.

### 6.23.4 Member Data Documentation

#### 6.23.4.1 \_id

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

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

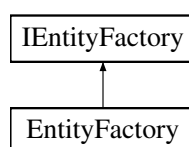
- /home/runner/work/R-Type/R-Type/ECS/Interface/Include/Entities/[entity.hpp](#)

## 6.24 EntityFactory Class Reference

A class responsible for creating different types of entities.

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

Inheritance diagram for EntityFactory:



## Public Member Functions

- [Entity createBackground](#) ([EntityManager](#) &entityManager, [ComponentManager](#) &componentManager) override  
*Creates a background entity.*
- [Entity createInfoBar](#) ([EntityManager](#) &entityManager, [ComponentManager](#) &componentManager) override  
*Creates a bar entity.*
- [Entity createPlayer](#) ([EntityManager](#) &entityManager, [ComponentManager](#) &componentManager, int nbrOf← Players) override  
*Creates a player entity.*
- [Entity createShooterEnemy](#) ([EntityManager](#) &entityManager, [ComponentManager](#) &componentManager, int posX, int posY) override  
*Creates a shooter enemy entity.*
- [Entity createBasicMonster](#) ([EntityManager](#) &entityManager, [ComponentManager](#) &componentManager, int posX, int posY) override  
*Creates a basic monster entity.*
- [Entity createPlayerMissile](#) ([EntityManager](#) &entityManager, [ComponentManager](#) &componentManager, uint32\_t entityId) override  
*Creates a player missile entity.*
- [Entity createForceWeapon](#) ([EntityManager](#) &entityManager, [ComponentManager](#) &componentManager, uint32\_t entityId) override
- [Entity createForceMissile](#) ([EntityManager](#) &entityManager, [ComponentManager](#) &componentManager, uint32\_t entityId) override
- [Entity createPowerUpBlueLaserCrystal](#) ([EntityManager](#) &entityManager, [ComponentManager](#) &component← Manager) override
- [Entity createButton](#) ([EntityManager](#) &entityManager, [ComponentManager](#) &componentManager, [TextureManager](#) &textureManager, [FontManager](#) &fontManager, std::string text, std::function< [IScenes](#) \*([AScenes](#) \*)> \*on← Click, float x=0, float y=0) override  
*Creates a button entity.*
- [Entity createSmallButton](#) ([EntityManager](#) &entityManager, [ComponentManager](#) &componentManager, [TextureManager](#) &textureManager, [FontManager](#) &fontManager, std::string text, std::function< [IScenes](#) \*([AScenes](#) \*, [AScenes::Actions](#))> \*onClick, float x=0, float y=0) override  
*Creates a small button entity.*
- [Entity createEnemyMissile](#) ([EntityManager](#) &entityManager, [ComponentManager](#) &componentManager, uint32\_t entityId) override  
*Creates an ally missile entity.*
- [Entity createFilter](#) ([EntityManager](#) &entityManager, [ComponentManager](#) &componentManager, [AScenes::DaltonismMode](#) mode)  
*Create a Filter object.*

## Additional Inherited Members

### 6.24.1 Detailed Description

A class responsible for creating different types of entities.

### 6.24.2 Member Function Documentation

### 6.24.2.1 createBackground()

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

Creates a background entity.

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

#### Parameters

<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](#).

### 6.24.2.2 createBasicMonster()

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

Creates a basic monster entity.

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

#### Parameters

<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](#).

### 6.24.2.3 createButton()

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

Creates a button entity.

This function creates a button entity with the specified parameters.

#### Parameters

<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](#).

### 6.24.2.4 createEnemyMissile()

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

Creates an ally missile entity.

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

#### Parameters

<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.

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.
<i>entityId</i>	The id of the entity that shoot the missile

#### Returns

The created enemy missile entity.

Implements [IEntityFactory](#).

#### 6.24.2.5 createFilter()

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

Create a Filter object.

#### Parameters

<i>entityManager</i>	
<i>componentManager</i>	
<i>mode</i>	

#### Returns

[Entity](#)

#### 6.24.2.6 createForceMissile()

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

Implements [IEntityFactory](#).

### 6.24.2.7 createForceWeapon()

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

Implements [IEntityFactory](#).

### 6.24.2.8 createInfoBar()

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

Creates a bar entity.

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

#### Parameters

<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 bar entity.

Implements [IEntityFactory](#).

### 6.24.2.9 createPlayer()

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

Creates a player entity.

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

#### Parameters

<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](#).

**6.24.2.10 createPlayerMissile()**

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

Creates a player missile entity.

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

**Parameters**

<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
<i>entityId</i>	The id of the entity that shoot the missile

**Returns**

The created player missile entity.

Implements [IEntityFactory](#).

**6.24.2.11 createPowerUpBlueLaserCrystal()**

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

Implements [IEntityFactory](#).

**6.24.2.12 createShooterEnemy()**

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

Creates a shooter enemy entity.

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

**Parameters**

<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](#).

**6.24.2.13 createSmallButton()**

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

Creates a small button entity.

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

**Parameters**

<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.

Implements [IEntityFactory](#).

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

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



## 6.25 EntityInformation Struct Reference

Represents information about an entity.

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

### Public Attributes

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

### 6.25.1 Detailed Description

Represents information about an entity.

### 6.25.2 Member Data Documentation

#### 6.25.2.1 animationComponent

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

#### 6.25.2.2 ratio

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

#### 6.25.2.3 spriteData

```
SpriteDataComponent EntityInformation::spriteData
```

#### 6.25.2.4 uniqueID

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

### 6.25.2.5 vPos

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

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

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

## 6.26 EntityManager Class Reference

Class responsible for managing entities in the ECS system.

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

### Public Member Functions

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

### Private Attributes

- int [entityNb](#) = 0  
*The number of entities in the entity manager.*
- std::vector< [Entity](#) > [entities](#)

### 6.26.1 Detailed Description

Class responsible for managing entities in the ECS system.

### 6.26.2 Member Function Documentation

#### 6.26.2.1 createEntity()

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

Create a [Entity](#) object.

Returns

[Entity](#)

### 6.26.2.2 getAllEntities()

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

Get all entities in the entity manager.

#### Returns

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

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

### 6.26.2.3 getEntity()

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

Get an entity by its ID.

#### Parameters

<i>entityId</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.

### 6.26.2.4 removeEntity()

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

Remove an entity from the entity manager.

#### Parameters

<i>entityId</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.

## 6.26.3 Member Data Documentation

### 6.26.3.1 entities

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

### 6.26.3.2 entityNb

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

The number of entities in the entity manager.

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

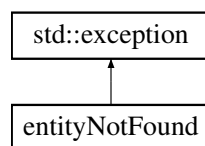
- /home/runner/work/R-Type/R-Type/ECS/Interface/Include/Entities/[entity\\_manager.hpp](#)

## 6.27 entityNotFound Class Reference

Exception class for entity not found error.

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

Inheritance diagram for entityNotFound:



### Private Member Functions

- `const char * what () const` noexcept override

### 6.27.1 Detailed Description

Exception class for entity not found error.

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

### 6.27.2 Member Function Documentation

### 6.27.2.1 what()

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

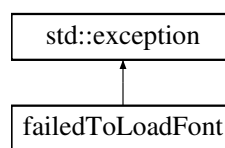
The documentation for this class was generated from the following file:

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

## 6.28 failedToLoadFont Class Reference

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

Inheritance diagram for failedToLoadFont:



### Private Member Functions

- `const char * what ( ) const noexcept override`

### 6.28.1 Member Function Documentation

#### 6.28.1.1 what()

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

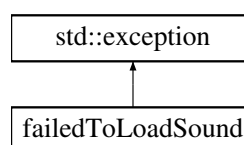
The documentation for this class was generated from the following file:

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

## 6.29 failedToLoadSound Class Reference

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

Inheritance diagram for failedToLoadSound:



## Private Member Functions

- `const char * what () const` noexcept override

### 6.29.1 Member Function Documentation

#### 6.29.1.1 `what()`

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

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

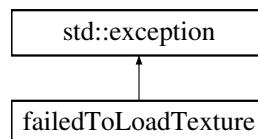
- `/home/runner/work/R-Type/R-Type/ECS/Interface/Include/error\_handling.hpp`

## 6.30 failedToLoadTexture Class Reference

Exception class for failed texture loading.

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

Inheritance diagram for failedToLoadTexture:



## Private Member Functions

- `const char * what () const` noexcept override

### 6.30.1 Detailed Description

Exception class for failed texture loading.

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

### 6.30.2 Member Function Documentation

### 6.30.2.1 what()

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

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

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

## 6.31 FontManager Class Reference

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

### Public Member Functions

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

### Private Attributes

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

## 6.31.1 Member Function Documentation

### 6.31.1.1 getFont()

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

### 6.31.1.2 releaseFont()

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

## 6.31.2 Member Data Documentation

### 6.31.2.1 fonts

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

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

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

## 6.32 ForceMissileComponent Struct Reference

```
#include <force_missile_component.hpp>
```

### Public Attributes

- uint32\_t [forceId](#)

### 6.32.1 Member Data Documentation

#### 6.32.1.1 forceId

```
uint32_t ForceMissileComponent::forceId
```

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

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

## 6.33 FrontComponent Struct Reference

```
#include <front_component.hpp>
```

### Public Member Functions

- [FrontComponent](#) (int \_targetId)

### Public Attributes

- int [targetId](#)



### 6.33.1 Constructor & Destructor Documentation

#### 6.33.1.1 FrontComponent()

```
FrontComponent::FrontComponent (
    int _targetId ) [inline]
```

### 6.33.2 Member Data Documentation

#### 6.33.2.1 targetId

```
int FrontComponent::targetId
```

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

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

## 6.34 HealthComponent Struct Reference

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

### Public Attributes

- int [max\\_health](#)
- int [health](#)

### 6.34.1 Member Data Documentation

#### 6.34.1.1 health

```
int HealthComponent::health
```

### 6.34.1.2 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)

## 6.35 HitboxComponent Struct Reference

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

### Public Attributes

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

### 6.35.1 Member Data Documentation

#### 6.35.1.1 h

```
int HitboxComponent::h
```

#### 6.35.1.2 w

```
int HitboxComponent::w
```

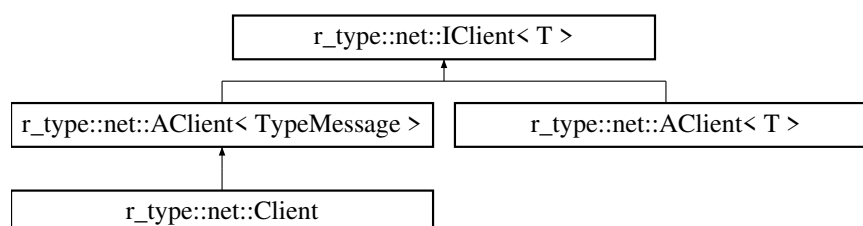
The documentation for this struct was generated from the following file:

- [/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)

## 6.36 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*

## 6.36.1 Constructor & Destructor Documentation

### 6.36.1.1 IClient()

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

### 6.36.1.2 ~IClient()

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

## 6.36.2 Member Function Documentation

### 6.36.2.1 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 >](#).

**6.36.2.2 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 >](#).

**6.36.2.3 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 >](#).

**6.36.2.4 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 >](#).

**6.36.2.5 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:

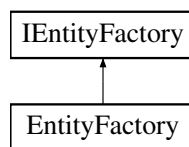
- [/home/runner/work/R-Type/R-Type/Client/Interface/Include/Net/i\\_client.hpp](#)

## 6.37 IEntityFactory Class Reference

The interface for an entity factory.

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

Inheritance diagram for IEntityFactory:



### Public Types

- enum [EnemyType](#) { [BasicMonster](#) , [ShooterEnemy](#) , [Boss](#) }

### Public Member Functions

- virtual [~IEntityFactory](#) ()=default  
*Destroy the IEntityFactory object.*
- virtual [Entity](#) [createBackground](#) ([EntityManager](#) &entityManager, [ComponentManager](#) &componentManager)=0  
*Creates a background entity.*
- virtual [Entity](#) [createInfoBar](#) ([EntityManager](#) &entityManager, [ComponentManager](#) &componentManager)=0  
*Creates a bar entity.*
- virtual [Entity](#) [createPlayer](#) ([EntityManager](#) &entityManager, [ComponentManager](#) &componentManager, int nbrOfPlayers)=0  
*Creates a player entity.*
- virtual [Entity](#) [createShooterEnemy](#) ([EntityManager](#) &entityManager, [ComponentManager](#) &componentManager, int posX, int posY)=0  
*Creates a shooter enemy entity.*
- virtual [Entity](#) [createBasicMonster](#) ([EntityManager](#) &entityManager, [ComponentManager](#) &componentManager, int posX, int posY)=0  
*Creates a basic monster entity.*
- virtual [Entity](#) [createPlayerMissile](#) ([EntityManager](#) &entityManager, [ComponentManager](#) &componentManager, uint32\_t entityId)=0

*Creates a player missile entity.*

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

*Creates an enemy missile entity.*

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

*Creates a button entity.*

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

### 6.37.1 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.

### 6.37.2 Member Enumeration Documentation

#### 6.37.2.1 EnemyType

```
enum IEntityManager::EnemyType
```

#### Enumerator

BasicMonster	
ShooterEnemy	
Boss	

### 6.37.3 Constructor & Destructor Documentation

### 6.37.3.1 ~IEntityFactory()

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

Destroy the [IEntityFactory](#) object.

## 6.37.4 Member Function Documentation

### 6.37.4.1 createBackground()

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

Creates a background entity.

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

#### Parameters

<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](#).

### 6.37.4.2 createBasicMonster()

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

Creates a basic monster entity.

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

#### Parameters

<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](#).

**6.37.4.3 createButton()**

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

Creates a button entity.

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

**Parameters**

<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](#).

**6.37.4.4 createEnemyMissile()**

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

Creates an enemy missile entity.

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



## Parameters

<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](#).

**6.37.4.5 createForceMissile()**

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

Implemented in [EntityFactory](#).

**6.37.4.6 createForceWeapon()**

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

Implemented in [EntityFactory](#).

**6.37.4.7 createInfoBar()**

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

Creates a bar entity.

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

## Parameters

<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 bar entity.

Implemented in [EntityFactory](#).

**6.37.4.8 createPlayer()**

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

Creates a player entity.

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

**Parameters**

<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](#).

**6.37.4.9 createPlayerMissile()**

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

Creates a player missile entity.

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

**Parameters**

<i>entityId</i>	The ID of the entity that shoot the missile.
<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](#).

**6.37.4.10 createPowerUpBlueLaserCrystal()**

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

Implemented in [EntityFactory](#).

**6.37.4.11 createShooterEnemy()**

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

Creates a shooter enemy entity.

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

**Parameters**

<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 shooter enemy entity.

Implemented in [EntityFactory](#).

**6.37.4.12 createSmallButton()**

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

```
std::string text,
std::function< IScenes *(AScenes *, AScenes::Actions)> * onClick,
float x = 0,
float y = 0 ) [pure virtual]
```

Implemented in [EntityFactory](#).

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

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

## 6.38 InputComponent Struct Reference

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

### Public Attributes

- [InputType input](#)

### 6.38.1 Member Data Documentation

#### 6.38.1.1 input

[InputType](#) InputComponent::input

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

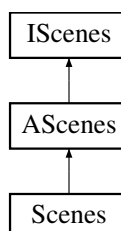
- [/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/input\\_component.hpp](#)

## 6.39 IScenes Class Reference

Interface for managing different scenes in a game.

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

Inheritance diagram for IScenes:



## Public Member Functions

- virtual [~IScenes](#) ()=default
- virtual void [mainMenu](#) ()=0  
*Displays the main menu and creates necessary entities.*
- virtual void [gameLoop](#) ()=0  
*Displays the main game loop and creates necessary entities.*
- virtual void [settingsMenu](#) ()=0  
*Displays the settings menu and creates necessary entities.*
- virtual void [inGameMenu](#) ()=0  
*Displays the in-game menu and creates necessary entities.*
- virtual void [difficultyChoices](#) ()=0  
*Displays the difficulty choices.*
- virtual void [render](#) ()=0  
*Displays the current scene and manages its components.*
- virtual bool [shouldQuit](#) ()=0  
*Checks if the game should quit.*
- virtual sf::RenderWindow \* [getRenderWindow](#) ()=0  
*Gets the render window.*

### 6.39.1 Detailed Description

Interface for managing different scenes in a game.

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

### 6.39.2 Constructor & Destructor Documentation

#### 6.39.2.1 [~IScenes\(\)](#)

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

### 6.39.3 Member Function Documentation

#### 6.39.3.1 [difficultyChoices\(\)](#)

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

Displays the difficulty choices.

Implemented in [Scenes](#).

### 6.39.3.2 gameLoop()

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

Displays the main game loop and creates necessary entities.

Implemented in [Scenes](#).

### 6.39.3.3 getRenderWindow()

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

Gets the render window.

#### Returns

Pointer to the sf::RenderWindow.

Implemented in [Scenes](#).

### 6.39.3.4 inGameMenu()

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

Displays the in-game menu and creates necessary entities.

Implemented in [Scenes](#).

### 6.39.3.5 mainMenu()

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

Displays the main menu and creates necessary entities.

Implemented in [Scenes](#).

### 6.39.3.6 render()

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

Displays the current scene and manages its components.

Implemented in [Scenes](#).

### 6.39.3.7 settingsMenu()

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

Displays the settings menu and creates necessary entities.

Implemented in [Scenes](#).

### 6.39.3.8 shouldQuit()

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

Checks if the game should quit.

#### Returns

True if the game should quit, false otherwise.

Implemented in [Scenes](#).

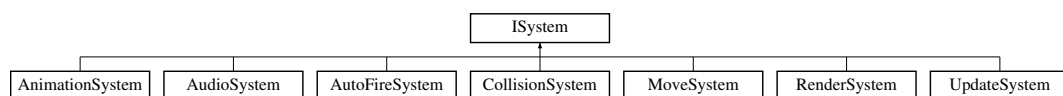
The documentation for this class was generated from the following file:

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

## 6.40 ISystem Class Reference

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

Inheritance diagram for ISystem:



### Public Member Functions

- [ISystem](#) ()=default
- virtual [~ISystem](#) ()=default

### 6.40.1 Constructor & Destructor Documentation

#### 6.40.1.1 ISystem()

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

#### 6.40.1.2 ~ISystem()

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

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

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

### 6.41 labelComponent Struct Reference

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

#### Public Attributes

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

#### 6.41.1 Member Data Documentation

##### 6.41.1.1 name

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

##### 6.41.1.2 x

```
int labelComponent::x
```



## 6.41.1.3 y

```
int labelComponent::y
```

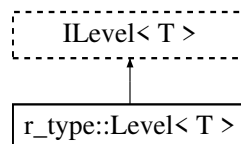
The documentation for this struct was generated from the following file:

- [/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)

## 6.42 r\_type::Level&lt; T &gt; Class Template Reference

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

Inheritance diagram for r\_type::Level< T >:



## Public Member Functions

- [Level](#) ()=default
- [~Level](#) ()=default
- void [Update](#) (r\_type::net::AServer< T > \*server, [ComponentManager](#) &componentManager, [EntityManager](#) &entityManager, std::chrono::system\_clock::time\_point newClock, bool \*bUpdateEntities) override  
*Updates the game state by processing entity movements, handling collisions, and sending messages to clients.*
- void [SetSystem](#) ([ComponentManager](#) &componentManager, [EntityManager](#) &entityManager) override  
*Initializes and sets up various systems for the level.*
- void [MoveUpdate](#) (r\_type::net::AServer< T > \*server, [ComponentManager](#) &componentManager, [EntityManager](#) &entityManager, std::chrono::system\_clock::time\_point newClock) override  
*Updates the positions of entities and notifies clients of any changes.*
- void [CollisionUpdate](#) (r\_type::net::AServer< T > \*server, [ComponentManager](#) &componentManager, [EntityManager](#) &entityManager, std::chrono::system\_clock::time\_point newClock) override  
*Updates the collision status of entities in the game.*
- void [AnimationUpdate](#) (r\_type::net::AServer< T > \*server, [ComponentManager](#) &componentManager, [EntityManager](#) &entityManager, std::chrono::system\_clock::time\_point newClock) override  
*Updates the animations of entities and sends messages to clients if animations have changed.*
- void [FireUpdate](#) (r\_type::net::AServer< T > \*server, [ComponentManager](#) &componentManager, [EntityManager](#) &entityManager, std::chrono::system\_clock::time\_point newClock) override  
*Updates the firing mechanism of entities in the game.*
- void [LevelOne](#) (r\_type::net::AServer< T > \*server, [ComponentManager](#) &componentManager, [EntityManager](#) &entityManager, std::chrono::system\_clock::time\_point newClock) override  
*Handles the spawning of entities for Level One.*
- void [SpawnEntity](#) (r\_type::net::AServer< T > \*server, [EntityManager](#) &entityManager, [ComponentManager](#) &componentManager, int nbrOfEnemy, [EnemyFactory::EnemyType](#) enemyType)  
*Spawns a specified number of enemy entities in the game.*
- void [SetGameParameters](#) (GameParameters gameParameters)  
*Sets the game difficulty based on the provided game parameters.*

## Protected Attributes

- `std::shared_ptr< MoveSystem > _moveSystem`
- `std::shared_ptr< CollisionSystem > _collisionSystem`
- `std::shared_ptr< AnimationSystem > _animationSystem`
- `std::shared_ptr< AutoFireSystem > _autoFireSystem`
- `std::chrono::system_clock::time_point _basicMonsterSpawnTime`
- `std::chrono::system_clock::time_point _shooterEnemySpawnTime`
- `std::chrono::system_clock::time_point _spawnTimeMonsterThree`
- `GameParameters \_gameParameters`

## 6.42.1 Constructor & Destructor Documentation

### 6.42.1.1 Level()

```
template<typename T >
r\_type::Level< T >::Level ( ) [default]
```

### 6.42.1.2 ~Level()

```
template<typename T >
r\_type::Level< T >::~~Level ( ) [default]
```

## 6.42.2 Member Function Documentation

### 6.42.2.1 AnimationUpdate()

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

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

This function performs the following steps:

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

## Parameters

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

## 6.42.2.2 CollisionUpdate()

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

Updates the collision status of entities in the game.

This function checks for collisions between entities and handles the consequences of those collisions, such as updating health, removing entities, and adding new entities. It also handles entities that go off-screen.

## Parameters

<i>server</i>	Pointer to the server instance.
<i>componentManager</i>	Reference to the component manager.
<i>entityManager</i>	Reference to the entity manager.
<i>newClock</i>	The current time point for the update.

## 6.42.2.3 FireUpdate()

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

Updates the firing mechanism of entities in the game.

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

## Parameters

<i>server</i>	Pointer to the server instance.
<i>componentManager</i>	Reference to the <a href="#">ComponentManager</a> handling components.
<i>entityManager</i>	Reference to the <a href="#">EntityManager</a> handling entities.
<i>newClock</i>	The current time point used for timing events.

#### 6.42.2.4 LevelOne()

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

Handles the spawning of entities for [Level One](#).

This function is responsible for spawning basic monsters and shooter enemies at specific intervals defined by the game parameters. It checks the elapsed time since the last spawn of each entity type and spawns new entities if the required time has passed.

##### Parameters

<i>server</i>	Pointer to the server instance.
<i>componentManager</i>	Reference to the <a href="#">ComponentManager</a> instance.
<i>entityManager</i>	Reference to the <a href="#">EntityManager</a> instance.
<i>newClock</i>	The current time point used for timing calculations.

#### 6.42.2.5 MoveUpdate()

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

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

This function performs the following steps:

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

##### Parameters

<i>server</i>	Pointer to the server instance.
<i>componentManager</i>	Reference to the <a href="#">ComponentManager</a> .
<i>entityManager</i>	Reference to the <a href="#">EntityManager</a> .
<i>newClock</i>	The current time point.

### 6.42.2.6 `SetGameParameters()`

```
template<typename T >
void r_type::Level< T >::SetGameParameters (
    GameParameters gameParameters ) [inline]
```

Sets the game difficulty based on the provided game parameters.

This function sets the game difficulty based on the provided game parameters.

#### Parameters

<code>gameParameters</code>	The game parameters to set the difficulty.
-----------------------------	--

### 6.42.2.7 `SetSystem()`

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

Initializes and sets up various systems for the level.

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

#### Parameters

<code>componentManager</code>	Reference to the <a href="#">ComponentManager</a> used to manage components.
<code>entityManager</code>	Reference to the <a href="#">EntityManager</a> used to manage entities.

### 6.42.2.8 `SpawnEntity()`

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

Spawns a specified number of enemy entities in the game.

This function creates and spawns a specified number of enemy entities of a given type at random positions within the game world. The enemy entities are then broadcasted to all connected clients.

## Template Parameters

<i>T</i>	The type of the server.
----------	-------------------------

## Parameters

<i>server</i>	A pointer to the server instance.
<i>entityManager</i>	Reference to the <a href="#">EntityManager</a> responsible for managing entities.
<i>componentManager</i>	Reference to the <a href="#">ComponentManager</a> responsible for managing components.
<i>nbrOfEnemy</i>	The number of enemy entities to spawn.
<i>enemyType</i>	The type of enemy to spawn (e.g., BasicMonster, ShooterEnemy).

## 6.42.2.9 Update()

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

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

This function performs several tasks to update the game state:

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

## Parameters

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

## 6.42.3 Member Data Documentation

### 6.42.3.1 \_animationSystem

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

### 6.42.3.2 \_autoFireSystem

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

### 6.42.3.3 \_basicMonsterSpawnTime

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

#### Initial value:

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

### 6.42.3.4 \_collisionSystem

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

### 6.42.3.5 \_gameParameters

```
template<typename T >
GameParameters r_type::Level< T >::_gameParameters [protected]
```

### 6.42.3.6 \_moveSystem

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

### 6.42.3.7 `_shooterEnemySpawnTime`

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

#### Initial value:

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

### 6.42.3.8 `_spawnTimeMonsterThree`

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

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

- [/home/runner/work/R-Type/R-Type/Server/Interface/Include/level.hpp](#)

## 6.43 MovementComponent Struct Reference

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

### Public Attributes

- [MovementType](#) `movementType`
- `uint32_t` [index](#)

### 6.43.1 Member Data Documentation

#### 6.43.1.1 `index`

```
uint32_t MovementComponent::index
```

#### 6.43.1.2 `movementType`

```
MovementType MovementComponent::movementType
```

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

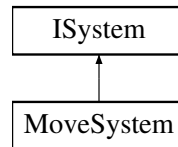
- [/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/movement\\_component.hpp](#)



## 6.44 MoveSystem Class Reference

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

Inheritance diagram for MoveSystem:



### Public Member Functions

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

### Private Attributes

- `ComponentManager` &\_componentManager
- `EntityManager` &\_entityManager

### 6.44.1 Constructor & Destructor Documentation

#### 6.44.1.1 MoveSystem()

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

### 6.44.2 Member Function Documentation

#### 6.44.2.1 moveEntities()

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

### 6.44.3 Member Data Documentation

#### 6.44.3.1 `_componentManager`

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

#### 6.44.3.2 `_entityManager`

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

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

- [/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Systems/move\\_system.hpp](#)
- [/home/runner/work/R-Type/R-Type/ECS/Src/Systems/move\\_system.cpp](#)

### 6.45 OffsetComponent Struct Reference

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

#### Public Attributes

- float [offset](#)

#### 6.45.1 Member Data Documentation

##### 6.45.1.1 `offset`

```
float OffsetComponent::offset
```

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

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

### 6.46 OnClickComponent Struct Reference

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

#### Public Member Functions

- [OnClickComponent](#) (std::function< [IScenes](#) \*([AScenes](#) \*)> onClickfunction)

## Public Attributes

- bool `isClicked` = false
- `std::function< IScenes *(AScenes *)>` `onClick`

### 6.46.1 Constructor & Destructor Documentation

#### 6.46.1.1 OnClickComponent()

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

### 6.46.2 Member Data Documentation

#### 6.46.2.1 isClicked

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

#### 6.46.2.2 onClick

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

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

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

## 6.47 PlayerComponent Struct Reference

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

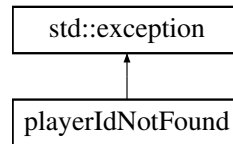
The documentation for this struct was generated from the following file:

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

## 6.48 playerIdNotFound Class Reference

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

Inheritance diagram for playerIdNotFound:



### Private Member Functions

- const char \* [what](#) () const noexcept override

### 6.48.1 Member Function Documentation

#### 6.48.1.1 what()

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

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

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

## 6.49 PlayerMissileComponent Struct Reference

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

### Public Attributes

- uint32\_t [playerId](#)

### 6.49.1 Member Data Documentation

#### 6.49.1.1 playerId

```
uint32_t PlayerMissileComponent::playerId
```

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](#)

## 6.50 PositionComponent Struct Reference

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

### Public Member Functions

- [PositionComponent](#) (float \_x, float \_y)

### Public Attributes

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

### 6.50.1 Constructor & Destructor Documentation

#### 6.50.1.1 PositionComponent()

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

### 6.50.2 Member Data Documentation

#### 6.50.2.1 x

```
float PositionComponent::x
```

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

## 6.51 PowerUpComponent Struct Reference

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

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

- [/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/power\\_up\\_component.hpp](/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/power_up_component.hpp)

## 6.52 RectangleShapeComponent Struct Reference

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

### Public Member Functions

- [RectangleShapeComponent](#) (sf::RectangleShape &[rectangleShape](#))

### Public Attributes

- sf::RectangleShape [rectangleShape](#)

### 6.52.1 Constructor & Destructor Documentation

#### 6.52.1.1 RectangleShapeComponent()

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

### 6.52.2 Member Data Documentation

### 6.52.2.1 rectangleShape

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

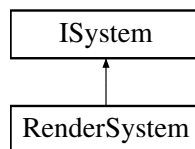
The documentation for this struct was generated from the following file:

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

## 6.53 RenderSystem Class Reference

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

Inheritance diagram for RenderSystem:



### Public Member Functions

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

### Private Attributes

- sf::RenderWindow & [\\_window](#)
- [ComponentManager](#) & [\\_componentManager](#)
- sf::Font [\\_font](#)

### 6.53.1 Constructor & Destructor Documentation

#### 6.53.1.1 RenderSystem()

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

### 6.53.2 Member Function Documentation

### 6.53.2.1 render()

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

## 6.53.3 Member Data Documentation

### 6.53.3.1 `_componentManager`

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

### 6.53.3.2 `_font`

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

### 6.53.3.3 `_window`

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

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

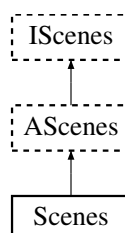
- [/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Systems/render\\_system.hpp](#)
- [/home/runner/work/R-Type/R-Type/ECS/Src/Systems/render\\_system.cpp](#)

## 6.54 Scenes Class Reference

Represents a class that manages different scenes in a game.

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

Inheritance diagram for Scenes:





## Public Member Functions

- [Scenes](#) (std::string ip, int port)  
*Construct a new [Scenes](#) object.*
- [~Scenes](#) ()=default  
*Destroy the [Scenes](#) object.*
- void [mainMenu](#) ()  
*displays the main menu, creates all the necessary entities*
- void [gameLoop](#) ()  
*displays the main game loop, creates all the necessary entities*
- void [HandleMessage](#) (r\_type::net::Message< TypeMessage > &msg, [ComponentManager](#) &componentManager, [TextureManager](#) &textureManager, [FontManager](#) &fontManager, std::shared\_ptr< [AudioSystem](#) > &audioSystem)
- void [StopGameLoop](#) (std::shared\_ptr< [AudioSystem](#) > &audioSystem)
- void [settingsMenu](#) ()  
*displays the settings menu, creates all the necessary entities*
- void [inGameMenu](#) ()  
*displays the in game menu, creates all the necessary entities*
- void [difficultyChoices](#) ()  
*displays the difficulty choices, creates all the necessary entities*
- void [render](#) ()  
*display what must be displayed (main menu, game loop, settings menu, in game menu), creates all the components needed and manages them*
- bool [shouldQuit](#) ()  
*check if game should stop running*
- sf::RenderWindow \* [getRenderWindow](#) ()  
*Get the RenderWindow object.*
- void [run](#) ()

## Public Attributes

- sf::RenderWindow [\\_window](#)
- r\_type::net::Client [\\_networkClient](#)

## Additional Inherited Members

### 6.54.1 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.

### 6.54.2 Constructor & Destructor Documentation

#### 6.54.2.1 Scenes()

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

Construct a new [Scenes](#) object.

#### Parameters

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

#### 6.54.2.2 ~Scenes()

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

Destroy the [Scenes](#) object.

### 6.54.3 Member Function Documentation

#### 6.54.3.1 difficultyChoices()

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

displays the difficulty choices, creates all the necessary entities

Implements [IScenes](#).

#### 6.54.3.2 gameLoop()

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

displays the main game loop, creates all the necessary entities

Implements [IScenes](#).

#### 6.54.3.3 getRenderWindow()

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

Get the RenderWindow object.

#### Returns

sf::RenderWindow\*

Implements [IScenes](#).

#### 6.54.3.4 HandleMessage()

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

#### 6.54.3.5 inGameMenu()

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

displays the in game menu, creates all the necessary entities

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

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

The game loop performs the following steps:

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

#### Note

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

#### See also

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

Displays the in-game menu.

Implements [IScenes](#).

#### 6.54.3.6 mainMenu()

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

displays the main menu, creates all the necessary entities

Displays the main menu scene.

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

##### Returns

void

Implements [IScenes](#).

#### 6.54.3.7 render()

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

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

Renders the current scene based on the value of currentScene.

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

##### Note

The currentScene variable must be set before calling this function.

Implements [IScenes](#).

#### 6.54.3.8 run()

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

### 6.54.3.9 settingsMenu()

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

displays the settings menu, creates all the necessary entities

Displays the settings menu.

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

Implements [IScenes](#).

### 6.54.3.10 shouldQuit()

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

check if game should stop running

#### Returns

true

false

Implements [IScenes](#).

### 6.54.3.11 StopGameLoop()

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

## 6.54.4 Member Data Documentation

### 6.54.4.1 \_networkClient

```
r\_type::net::Client Scenes::_networkClient
```

#### 6.54.4.2 `_window`

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

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

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

### 6.55 ScoreComponent Struct Reference

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

#### Public Attributes

- `int score`

#### 6.55.1 Member Data Documentation

##### 6.55.1.1 `score`

```
int ScoreComponent::score
```

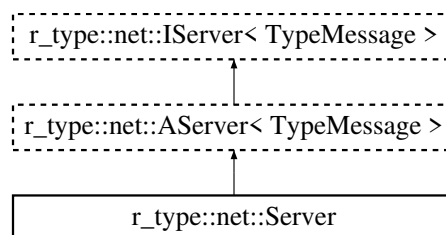
The documentation for this struct was generated from the following file:

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

### 6.56 `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

### 6.56.1 Constructor & Destructor Documentation

#### 6.56.1.1 Server()

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

#### 6.56.1.2 ~Server()

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

### 6.56.2 Member Function Documentation

#### 6.56.2.1 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

**6.56.2.2 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>	
---------------	--

**6.56.2.3 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](#)

**6.57 ShaderComponent Struct Reference**

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



## Public Member Functions

- [ShaderComponent](#) (std::string path)

## Public Attributes

- std::shared\_ptr< sf::Shader > [shader](#)

## 6.57.1 Constructor & Destructor Documentation

### 6.57.1.1 ShaderComponent()

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

## 6.57.2 Member Data Documentation

### 6.57.2.1 shader

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

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

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

## 6.58 ShootComponent Struct Reference

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

## Public Member Functions

- [ShootComponent](#) (std::chrono::milliseconds cooldown)

## Public Attributes

- std::chrono::system\_clock::time\_point [nextShootTime](#)
- std::chrono::milliseconds [cooldownTime](#)
- bool [canShoot](#)

## 6.58.1 Constructor & Destructor Documentation

### 6.58.1.1 ShootComponent()

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

## 6.58.2 Member Data Documentation

### 6.58.2.1 canShoot

```
bool ShootComponent::canShoot
```

### 6.58.2.2 cooldownTime

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

### 6.58.2.3 nextShootTime

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

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

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

## 6.59 SpriteComponent Struct Reference

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

### Public Member Functions

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

## Public Attributes

- `sf::Sprite` [sprite](#)
- `AScenes::SpriteType` [type](#)
- `int` [hitboxX](#)
- `int` [hitboxY](#)

## 6.59.1 Constructor & Destructor Documentation

### 6.59.1.1 SpriteComponent()

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

## 6.59.2 Member Data Documentation

### 6.59.2.1 hitboxX

```
int SpriteComponent::hitboxX
```

### 6.59.2.2 hitboxY

```
int SpriteComponent::hitboxY
```

### 6.59.2.3 sprite

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

#### 6.59.2.4 type

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

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

- `/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/sprite\_component.hpp`

## 6.60 SpriteDataComponent Struct Reference

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

### Public Attributes

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

### 6.60.1 Member Data Documentation

#### 6.60.1.1 scale

`vf2d` `SpriteDataComponent::scale`

#### 6.60.1.2 spritePath

`SpritePath` `SpriteDataComponent::spritePath`

#### 6.60.1.3 type

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

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

- `/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/sprite\_data\_component.hpp`

## 6.61 TextComponent Struct Reference

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

### Public Member Functions

- [TextComponent](#) (sf::Font &font, const std::string &string, float posX, float posY, int size=30)

### Public Attributes

- sf::Text [text](#)

### 6.61.1 Constructor & Destructor Documentation

#### 6.61.1.1 TextComponent()

```
TextComponent::TextComponent (
    sf::Font & font,
    const std::string & string,
    float posX,
    float posY,
    int size = 30 ) [inline]
```

### 6.61.2 Member Data Documentation

#### 6.61.2.1 text

```
sf::Text TextComponent::text
```

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

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

## 6.62 TextDataComponent Struct Reference

```
#include <text_data_component.hpp>
```

## Public Attributes

- [FontPath](#) `fontPath`
- `uint32_t` `charSize` = 0
- `uint32_t` `categoryIds` [5] = {0}
- [GameText](#) `categoryTexts` [5]
- `uint32_t` `categorySize` = 0

## 6.62.1 Member Data Documentation

### 6.62.1.1 categoryIds

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

### 6.62.1.2 categorySize

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

### 6.62.1.3 categoryTexts

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

### 6.62.1.4 charSize

```
uint32_t TextDataComponent::charSize = 0
```

### 6.62.1.5 fontPath

```
FontPath TextDataComponent::fontPath
```

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

- `/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/text\_data\_component.hpp`

## 6.63 TextureManager Class Reference

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

### Public Member Functions

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

### Private Attributes

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

### 6.63.1 Member Function Documentation

#### 6.63.1.1 `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.

#### 6.63.1.2 `releaseTexture()`

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

## 6.63.2 Member Data Documentation

### 6.63.2.1 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`

## 6.64 UIEntityInformation Struct Reference

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

### Public Attributes

- `uint32_t` [uniqueID](#) = 0
- `uint32_t` [lives](#) = 0
- `uint32_t` [score](#) = 0
- [SpriteDataComponent](#) [spriteData](#)
- [TextDataComponent](#) [textData](#)

### 6.64.1 Member Data Documentation

#### 6.64.1.1 lives

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

#### 6.64.1.2 score

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



### 6.64.1.3 spriteData

`SpriteDataComponent` `UIEntityInformation::spriteData`

### 6.64.1.4 textData

`TextDataComponent` `UIEntityInformation::textData`

### 6.64.1.5 uniqueID

`uint32_t` `UIEntityInformation::uniqueID = 0`

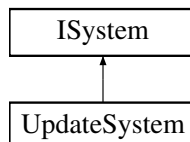
The documentation for this struct was generated from the following file:

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

## 6.65 UpdateSystem Class Reference

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

Inheritance diagram for UpdateSystem:



### Public Member Functions

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

### Private Attributes

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

### 6.65.1 Constructor & Destructor Documentation

### 6.65.1.1 UpdateSystem()

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

## 6.65.2 Member Function Documentation

### 6.65.2.1 updateSpritePositions()

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

## 6.65.3 Member Data Documentation

### 6.65.3.1 \_componentManager

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

### 6.65.3.2 \_entityManager

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

### 6.65.3.3 \_window

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

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

- [/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Systems/update\\_system.hpp](#)
- [/home/runner/work/R-Type/R-Type/ECS/Src/Systems/update\\_system.cpp](#)

## 6.66 VelocityComponent Struct Reference

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

## Public Attributes

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

### 6.66.1 Member Data Documentation

#### 6.66.1.1 [x](#)

```
float VelocityComponent::x
```

#### 6.66.1.2 [y](#)

```
float VelocityComponent::y
```

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

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

## 6.67 vf2d Struct Reference

Represents a 2D vector with x and y coordinates.

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

## Public Attributes

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

### 6.67.1 Detailed Description

Represents a 2D vector with x and y coordinates.

### 6.67.2 Member Data Documentation

### 6.67.2.1 x

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

### 6.67.2.2 y

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

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

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

## 6.68 WeaponComponent Struct Reference

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

### Public Member Functions

- [WeaponComponent](#) (uint32\_t \_playerId, uint32\_t \_level)

### Public Attributes

- uint32\_t [playerId](#)
- uint32\_t [level](#)

## 6.68.1 Constructor & Destructor Documentation

### 6.68.1.1 WeaponComponent()

```
WeaponComponent::WeaponComponent (
    uint32_t _playerId,
    uint32_t _level ) [inline]
```

## 6.68.2 Member Data Documentation

### 6.68.2.1 level

```
uint32_t WeaponComponent::level
```

### 6.68.2.2 playerId

```
uint32_t WeaponComponent::playerId
```

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 <SFML/Graphics.hpp>
#include <entity_struct.hpp>
#include <font_manager.hpp>
#include <texture_manager.hpp>
#include <unordered_map>
```

## Classes

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

## Namespaces

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

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

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

## Classes

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

## Namespaces

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

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

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

## Classes

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

## Namespaces

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

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

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

### Classes

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

### Functions

- std::string [keyToString](#) (sf::Keyboard::Key key)

#### 7.5.1 Function Documentation

##### 7.5.1.1 keyToString()

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

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

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

### Functions

- std::string [keyToString](#) (sf::Keyboard::Key key)

#### 7.6.1 Function Documentation

#### 7.6.1.1 keyToString()

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

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

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

### Functions

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

*The entry point of the program.*

### 7.7.1 Function Documentation

#### 7.7.1.1 isValidIPv4()

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

#### 7.7.1.2 isValidPort()

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

#### 7.7.1.3 main()

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

The entry point of the program.

This function initializes the Rtype object and runs the game.

#### Returns

0 indicating successful program execution.  
int



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

```
#include <Net/server.hpp>
#include <iostream>
#include <errno.h>
#include <signal.h>
#include <stdio.h>
```

### Functions

- void [signal\\_handler](#) (int signal)
- static bool [isValidPort](#) (const std::string &portStr)
- int [main](#) (int const argc, char const \*const \*const argv)

### Variables

- static bool [loopRunning](#) = true

### 7.8.1 Function Documentation

#### 7.8.1.1 isValidPort()

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

#### 7.8.1.2 main()

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

#### 7.8.1.3 signal\_handler()

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

## 7.8.2 Variable Documentation

### 7.8.2.1 loopRunning

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

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

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

## Functions

- void [reloadFilter](#) (sf::RectangleShape &rectangle, [AScenes::DaltonismMode](#) mode)
- void [handleEvents](#) (sf::Event event, [ComponentManager](#) &componentManager, sf::RenderWindow \*\_window, std::vector< std::shared\_ptr< [Entity](#) >> buttons, [Scenes](#) \*scenes)  
*Handles events for the scene, including window close and mouse button press events.*
- void [createDaltonismChoiceButtons](#) (std::vector< std::shared\_ptr< [Entity](#) >> &buttons, [ComponentManager](#) &componentManager, [EntityManager](#) &entityManager, [TextureManager](#) &textureManager, [FontManager](#) fontManager, [EntityFactory](#) &entityFactory)
- sf::Keyboard::Key [waitForKey](#) (sf::RenderWindow \*\_window)
- void [createKeyBindingButtons](#) (std::vector< std::shared\_ptr< [Entity](#) >> &buttons, [ComponentManager](#) &componentManager, [EntityManager](#) &entityManager, [TextureManager](#) &textureManager, [FontManager](#) fontManager, [EntityFactory](#) &entityFactory, std::map< [Scenes::Actions](#), sf::Keyboard::Key > &keyBinds)

### 7.9.1 Function Documentation

### 7.9.1.1 createDaltonismChoiceButtons()

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

### 7.9.1.2 createKeyBindingButtons()

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

### 7.9.1.3 handleEvents()

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

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

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

#### Parameters

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

#### 7.9.1.4 reloadFilter()

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

#### 7.9.1.5 waitForKey()

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

### 7.10 /home/runner/work/R-Type/R-Type/ECS/Interface/Include/a\_↵ scenes.hpp File Reference

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

#### Classes

- class [AScenes](#)

### 7.11 /home/runner/work/R-Type/R-Type/ECS/Interface/Include/audio\_↵ manager.hpp File Reference

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

#### Classes

- class [AudioManager](#)

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

#### Classes

- struct [AllyComponent](#)

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

### Classes

- struct [AllyMissileComponent](#)

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

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

### Classes

- struct [AnimationComponent](#)

### Functions

- bool [operator!=](#) ([AnimationComponent](#) animation, [AnimationComponent](#) other)  
*Inequality operator for [AnimationComponent](#).*

#### 7.14.1 Function Documentation

##### 7.14.1.1 [operator!=\(\)](#)

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

Inequality operator for [AnimationComponent](#).

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

#### Parameters

<i>animation</i>	The first <a href="#">AnimationComponent</a> to compare.
<i>other</i>	The second <a href="#">AnimationComponent</a> to compare.

**Returns**

true if the [AnimationComponent](#) objects are not equal, false otherwise.

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

**Classes**

- struct [BackgroundComponent](#)

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

**Classes**

- struct [BasicMonsterComponent](#)

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

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

**Classes**

- struct [BindComponent](#)

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

**Classes**

- struct [BossComponent](#)

## 7.19 /home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/component\_manager.hpp File Reference

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

### Classes

- class [ComponentManager](#)

*Manages the components of entities in an ECS system.*

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

```
#include "ally_component.hpp"
#include "ally_missile_component.hpp"
#include "animation_component.hpp"
#include "background_component.hpp"
#include "basic_monster_component.hpp"
#include "bind_component.hpp"
#include "enemy_component.hpp"
#include "enemy_missile_component.hpp"
#include "force_missile_component.hpp"
#include "front_component.hpp"
#include "health_component.hpp"
#include "hitbox_component.hpp"
#include "input_component.hpp"
#include "movement_component.hpp"
#include "offset_component.hpp"
#include "on_click_component.hpp"
#include "player_component.hpp"
#include "player_missile_component.hpp"
#include "position_component.hpp"
#include "power_up_component.hpp"
#include "rectangleShapeComponent.hpp"
#include "score_component.hpp"
#include "shoot_component.hpp"
#include "sprite_component.hpp"
#include "sprite_data_component.hpp"
#include "text_component.hpp"
#include "text_data_component.hpp"
#include "velocity_component.hpp"
#include "weapon_component.hpp"
```

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

### Classes

- struct [EnemyComponent](#)

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

### Classes

- struct [EnemyMissileComponent](#)

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

```
#include <stdint>
```

### Classes

- struct [ForceMissileComponent](#)

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

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

### Classes

- struct [FrontComponent](#)

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

### Classes

- struct [HealthComponent](#)



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

### Classes

- struct [HitboxComponent](#)

## 7.27 /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](#) }

#### 7.27.1 Enumeration Type Documentation

##### 7.27.1.1 InputType

```
enum InputType [strong]
```

##### Enumerator

UP	
DOWN	
LEFT	
RIGHT	
SHOOT	
QUIT	
NONE	

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

```
#include <iostream>
```

## Classes

- struct [labelComponent](#)

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

```
#include <stdint>
```

## Classes

- struct [MovementComponent](#)

## Enumerations

- enum class [MovementType](#) { [WIGGLE](#) , [DIAGONAL](#) , [CIRCLE](#) }

### 7.29.1 Enumeration Type Documentation

#### 7.29.1.1 MovementType

```
enum MovementType [strong]
```

##### Enumerator

WIGGLE	
DIAGONAL	
CIRCLE	

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

## Classes

- struct [OffsetComponent](#)

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

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

### Classes

- struct [OnClickComponent](#)

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

### Classes

- struct [PlayerComponent](#)

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

```
#include <cstdint>
```

### Classes

- struct [PlayerMissileComponent](#)

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

### Classes

- struct [PositionComponent](#)

## 7.35 /home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/power\_up\_component.hpp File Reference

### Classes

- struct [PowerUpComponent](#)

### 7.36 [/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/rectangleShapeComponent.hpp](#) File Reference

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

#### Classes

- struct [RectangleShapeComponent](#)

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

#### Classes

- struct [ScoreComponent](#)

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

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

#### Classes

- struct [ShaderComponent](#)

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

```
#include <chrono>
```

#### Classes

- struct [ShootComponent](#)

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

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

### Classes

- struct [SpriteComponent](#)

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

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

### Classes

- struct [SpriteDataComponent](#)

### Functions

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

#### 7.41.1 Function Documentation

##### 7.41.1.1 `operator<<()`

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

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

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

### Classes

- struct [TextComponent](#)

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

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

### Classes

- struct [TextDataComponent](#)

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

### Classes

- struct [VelocityComponent](#)

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

```
#include <cstdint>
```

### Classes

- struct [WeaponComponent](#)

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

```
#include <cstdint>
```

## Enumerations

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

### 7.46.1 Enumeration Type Documentation

#### 7.46.1.1 CreatableClientObject

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

Enumerator

<a href="#">PLAYERMISSILE</a>	
<a href="#">NONE</a>	

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

### Classes

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

## 7.48 /home/runner/work/R-Type/R-Type/ECS/Interface/Include/Entities/entity\_factory.hpp File Reference

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

### Classes

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

## 7.49 /home/runner/work/R-Type/R-Type/ECS/Interface/Include/Entities/entity\_manager.hpp File Reference

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

### Classes

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

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

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

### Classes

- class [IEntityFactory](#)  
*The interface for an entity factory.*

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

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

### Classes

- struct [EntityInformation](#)  
*Represents information about an entity.*
- struct [UIEntityInformation](#)



## 7.52 /home/runner/work/R-Type/R-Type/ECS/Interface/Include/error\_↵ handling.hpp File Reference

```
#include <exception>
```

### Classes

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

## 7.53 /home/runner/work/R-Type/R-Type/ECS/Interface/Include/font\_↵ manager.hpp File Reference

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

### Classes

- class [FontManager](#)

## 7.54 /home/runner/work/R-Type/R-Type/ECS/Interface/Include/font\_↵ path.hpp File Reference

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

### Enumerations

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

### Functions

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

## 7.54.1 Enumeration Type Documentation

### 7.54.1.1 FontPath

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

Enumerator

MAIN	
NONE	

## 7.54.2 Function Documentation

### 7.54.2.1 FontFactory()

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

### 7.54.2.2 operator<<()

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

## 7.55 /home/runner/work/R-Type/R-Type/ECS/Interface/Include/game\_↵ text.hpp File Reference

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

### Enumerations

- enum class GameText : uint32\_t { Lives , Score , NONE }

### Functions

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

## 7.55.1 Enumeration Type Documentation

### 7.55.1.1 GameText

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

#### Enumerator

Lives	
Score	
NONE	

## 7.55.2 Function Documentation

### 7.55.2.1 GameTextFactory()

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

### 7.55.2.2 operator<<()

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

## 7.56 /home/runner/work/R-Type/R-Type/ECS/Interface/Include/hitbox\_tmp.hpp File Reference

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

## Functions

- int [CheckEntityPosition](#) (uint32\_t entityId, [ComponentManager](#) componentManager, [EntityManager](#) entityManager)
- int [CheckEntityMovement](#) ([EntityInformation](#) desc, [ComponentManager](#) componentManager, [EntityManager](#) entityManager)

## 7.56.1 Function Documentation

### 7.56.1.1 CheckEntityMovement()

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

### 7.56.1.2 CheckEntityPosition()

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

## 7.57 /home/runner/work/R-Type/R-Type/ECS/Interface/Include/i\_↵ scenes.hpp File Reference

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

### Classes

- class [IScenes](#)  
*Interface for managing different scenes in a game.*

## 7.58 /home/runner/work/R-Type/R-Type/ECS/Interface/↵ Include/macros.hpp File Reference

### Classes

- struct [vf2d](#)  
*Represents a 2D vector with x and y coordinates.*

### Macros

- #define [SCREEN\\_WIDTH](#) 1920
- #define [SCREEN\\_HEIGHT](#) 1080

## 7.58.1 Macro Definition Documentation

### 7.58.1.1 SCREEN\_HEIGHT

```
#define SCREEN_HEIGHT 1080
```

### 7.58.1.2 SCREEN\_WIDTH

```
#define SCREEN_WIDTH 1920
```

## 7.59 /home/runner/work/R-Type/R-Type/ECS/Interface/Include/sound\_path.hpp File Reference

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

### Enumerations

- enum class [ActionType](#) : uint32\_t {  
[Win](#) , [Shot](#) , [Boss](#) , [PowerUp](#) ,  
[GameOver](#) , [BossDeath](#) , [Explosion](#) , [Background](#) ,  
[NONE](#) }

### Functions

- std::string [SoundFactory](#) ([ActionType](#) action)

## 7.59.1 Enumeration Type Documentation

### 7.59.1.1 ActionType

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

#### Enumerator

Win	
Shot	
Boss	
PowerUp	
GameOver	
BossDeath	
Explosion	

## 7.59.2 Function Documentation

### 7.59.2.1 SoundFactory()

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

## 7.60 /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](#) , [Missile](#) , [Weapon](#) ,  
    [BlueLaserCrystal](#) , [Background1](#) , [Background2](#) , [Background3](#) ,  
    [Explosion](#) , [PowerUp](#) , [Boss](#) , [BossBullet](#) ,  
    [Bar](#) , [NONE](#) }

### Functions

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

### 7.60.1 Enumeration Type Documentation

## Enumerator

---

### 7.60.1.1 SpritePath

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

#### Enumerator

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

## 7.60.2 Function Documentation

### 7.60.2.1 operator<<()

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

### 7.60.2.2 SpriteFactory()

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

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

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

### Classes

- class [AnimationSystem](#)

### Enumerations

- enum class [AnimationShip](#) : uint32\_t {  
    SHIP\_DOWN , SHIP\_FLIP\_DOWN , SHIP\_STRAIT , SHIP\_FLIP\_UP ,  
    SHIP\_UP }
- enum class [AnimationBasicMonster](#) : uint32\_t {  
    BASIC\_MONSTER\_DEFAULT , BASIC\_MONSTER\_1 , BASIC\_MONSTER\_2 , BASIC\_MONSTER\_3 ,  
    BASIC\_MONSTER\_4 , BASIC\_MONSTER\_5 , BASIC\_MONSTER\_6 , BASIC\_MONSTER\_7 }
- enum class [AnimationWeapon1](#) : uint32\_t {  
    WEAPON\_1\_DEFAULT , WEAPON\_1\_1 , WEAPON\_1\_2 , WEAPON\_1\_3 ,  
    WEAPON\_1\_4 , WEAPON\_1\_5 }

### Functions

- bool [operator!=](#) ([AnimationComponent](#) animation, [AnimationComponent](#) other)  
    *get if two animations are different.*
- [vf2d animationShipFactory](#) ([AnimationShip](#) animation)  
    *Factory function to create a ship animation.*

#### 7.61.1 Enumeration Type Documentation

##### 7.61.1.1 AnimationBasicMonster

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

##### Enumerator

BASIC_MONSTER_DEFAULT	
BASIC_MONSTER_1	
BASIC_MONSTER_2	
BASIC_MONSTER_3	
BASIC_MONSTER_4	
BASIC_MONSTER_5	
BASIC_MONSTER_6	
BASIC_MONSTER_7	



### 7.61.1.2 AnimationShip

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

#### Enumerator

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

### 7.61.1.3 AnimationWeapon1

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

#### Enumerator

WEAPON_1_DEFAULT	
WEAPON_1_1	
WEAPON_1_2	
WEAPON_1_3	
WEAPON_1_4	
WEAPON_1_5	

## 7.61.2 Function Documentation

### 7.61.2.1 animationShipFactory()

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

Factory function to create a ship animation.

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

#### Parameters

<i>animation</i>	The AnimationShip object containing the animation details.
------------------	--

**Returns**

[vf2d](#) The generated animation for the ship.

Factory function to create a ship animation.

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

**Parameters**

<i>animation</i>	The animation state of the ship, represented by the AnimationShip enumeration.
------------------	--

**Returns**

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

**7.61.2.2 operator"!=()**

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

get if two animations are different.

**Parameters**

<i>animation</i>	The first animation.
<i>other</i>	The second animation.

**Returns**

bool true if the animations are different, false otherwise.

get if two animations are different.

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

**Parameters**

<i>animation</i>	The first <a href="#">AnimationComponent</a> to compare.
<i>other</i>	The second <a href="#">AnimationComponent</a> to compare.

#### Returns

true if the [AnimationComponent](#) objects are not equal, false otherwise.

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

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

#### Classes

- class [AudioSystem](#)

## 7.63 /home/runner/work/R-Type/R-Type/ECS/Interface/Include/↵ Systems/auto\_fire\_system.hpp File Reference

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

#### Classes

- class [AutoFireSystem](#)

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

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

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

#### Classes

- class [CollisionSystem](#)

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

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

### Classes

- class [ISystem](#)

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

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

### Classes

- class [MoveSystem](#)

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

```
#include "Systems/i_system.hpp"  
#include <error_handling.hpp>
```

### Classes

- class [RenderSystem](#)

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

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

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

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

### Classes

- class [UpdateSystem](#)

## 7.71 /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](#)

## 7.72 /home/runner/work/R-Type/R-Type/ECS/Src/a\_scenes.cpp File Reference

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

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

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

## Functions

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

### 7.73.1 Function Documentation

#### 7.73.1.1 `operator<<()` [1/3]

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

#### 7.73.1.2 `operator<<()` [2/3]

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

#### 7.73.1.3 `operator<<()` [3/3]

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

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

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

## Functions

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

### 7.74.1 Function Documentation

#### 7.74.1.1 FontFactory()

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

### 7.75 /home/runner/work/R-Type/R-Type/ECS/Src/game\_text.cpp File Reference

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

#### Functions

- std::string [GameTextFactory](#) ([GameText](#) text)

#### 7.75.1 Function Documentation

##### 7.75.1.1 GameTextFactory()

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

### 7.76 /home/runner/work/R-Type/R-Type/ECS/Src/hitbox\_tmp.cpp File Reference

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

#### Functions

- static int [CheckCollisionLogic](#) (float descLeft, float descRight, float descTop, float descBottom, [ComponentManager](#) componentManager, [EntityManager](#) entityManager, int entityId)
- int [CheckEntityPosition](#) (uint32\_t entityId, [ComponentManager](#) componentManager, [EntityManager](#) entityManager↔)
- int [CheckEntityMovement](#) ([EntityInformation](#) desc, [ComponentManager](#) componentManager, [EntityManager](#) entityManager)

#### 7.76.1 Function Documentation

### 7.76.1.1 CheckCollisionLogic()

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

### 7.76.1.2 CheckEntityMovement()

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

### 7.76.1.3 CheckEntityPosition()

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

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

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

### Functions

- std::string SoundFactory (ActionType action)

### 7.77.1 Function Documentation

#### 7.77.1.1 SoundFactory()

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



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

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

### Functions

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

#### 7.78.1 Function Documentation

##### 7.78.1.1 [SpriteFactory\(\)](#)

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

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

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

### Functions

- `vf2d` [animationShipFactory](#) ([AnimationShip](#) animation)  
*Generates a vector representing the animation state of a ship.*
- `vf2d` [animationBasicMonsterFactory](#) ([AnimationBasicMonster](#) animation)
- `vf2d` [animationWeapon1Factory](#) ([AnimationWeapon1](#) animation)
- `bool` [operator!=](#) ([AnimationComponent](#) animation, [AnimationComponent](#) other)  
*Inequality operator for [AnimationComponent](#).*

#### 7.79.1 Function Documentation

##### 7.79.1.1 [animationBasicMonsterFactory\(\)](#)

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

### 7.79.1.2 animationShipFactory()

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

Generates a vector representing the animation state of a ship.

Factory function to create a ship animation.

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

#### Parameters

<i>animation</i>	The animation state of the ship, represented by the AnimationShip enumeration.
------------------	--

#### Returns

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

### 7.79.1.3 animationWeapon1Factory()

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

### 7.79.1.4 operator"!="()

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

Inequality operator for [AnimationComponent](#).

get if two animations are different.

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

#### Parameters

<i>animation</i>	The first <a href="#">AnimationComponent</a> to compare.
<i>other</i>	The second <a href="#">AnimationComponent</a> to compare.

#### Returns

true if the [AnimationComponent](#) objects are not equal, false otherwise.

### 7.80 /home/runner/work/R-Type/R-Type/ECS/Src/Systems/audio\_system.cpp File Reference

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

### 7.81 /home/runner/work/R-Type/R-Type/ECS/Src/Systems/auto\_fire\_system.cpp File Reference

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

### 7.82 /home/runner/work/R-Type/R-Type/ECS/Src/Systems/collision\_system.cpp File Reference

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

### 7.83 /home/runner/work/R-Type/R-Type/ECS/Src/Systems/move\_system.cpp File Reference

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

### 7.84 /home/runner/work/R-Type/R-Type/ECS/Src/Systems/render\_system.cpp File Reference

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

### 7.85 /home/runner/work/R-Type/R-Type/ECS/Src/Systems/update\_system.cpp File Reference

```
#include "Systems/update_system.hpp"
```

## 7.86 /home/runner/work/R-Type/R-Type/Server/Interface/↵ Include/level.hpp File Reference

```
#include <Components/component_manager.hpp>
#include <Components/components.hpp>
#include <cmath>
#include <game_struct.h>
#include <i_level.hpp>
```

### Classes

- class [r\\_type::Level< T >](#)

### Namespaces

- [r\\_type](#)
- [r\\_type::net](#)

## 7.87 /home/runner/work/R-Type/R-Type/Server/Interface/Include/Net/a\_↵ server.hpp File Reference

```
#include <Components/component_manager.hpp>
#include <Components/components.hpp>
#include <Entities/entity_factory.hpp>
#include <Entities/entity_manager.hpp>
#include <Net/i_server.hpp>
#include <Systems/systems.hpp>
#include <cmath>
#include <entity_struct.hpp>
#include <error_handling.hpp>
#include <game_struct.h>
#include <level.hpp>
#include <macros.hpp>
#include <unordered_map>
```

### Classes

- class [r\\_type::net::AServer< T >](#)  
*AServer class template for managing server operations.*

### Namespaces

- [r\\_type](#)
- [r\\_type::net](#)

## 7.88 /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](#)

## 7.89 /home/runner/work/R-Type/R-Type/Server/Interface/Include/r\_type-server.hpp File Reference

## 7.90 /home/runner/work/R-Type/R-Type/Server/Src/r\_type-server.cpp File Reference

## 7.91 /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, 129  
/home/runner/work/R-Type/R-Type/Client/Interface/Include/Net/client.hpp, 130  
/home/runner/work/R-Type/R-Type/Client/Interface/Include/Net/i\_client.hpp, 130  
/home/runner/work/R-Type/R-Type/Client/Interface/Include/mainmenu.hpp, 129  
/home/runner/work/R-Type/R-Type/Client/Interface/Include/scenes.hpp, 131  
/home/runner/work/R-Type/R-Type/Client/Src/keyToString.cpp, 131  
/home/runner/work/R-Type/R-Type/Client/Src/main.cpp, 132  
/home/runner/work/R-Type/R-Type/Client/Src/scenes.cpp, 134  
/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/all\_component.hpp, 136  
/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/all\_missile\_component.hpp, 137  
/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/animation\_component.hpp, 137  
/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/background\_component.hpp, 138  
/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/basic\_monster\_component.hpp, 138  
/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/bind\_component.hpp, 138  
/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/boss\_component.hpp, 138  
/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/component\_manager.hpp, 139  
/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/components.hpp, 139  
/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/enemy\_component.hpp, 140  
/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/enemy\_missile\_component.hpp, 140  
/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/force\_missile\_component.hpp, 140  
/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/front\_component.hpp, 140  
/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/health\_component.hpp, 140  
/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/hibitbox\_component.hpp, 141  
/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/input\_component.hpp, 141  
/home/runner/work/R-Type/R-Type/ECS/Interface/Include/Components/label\_component.hpp, 141





- RenderSystem, 108
- UpdateSystem, 126
- \_currentDaltonismMode
  - AScenes, 27
- \_currentGameMode
  - AScenes, 27
- \_currentMusicFilePath
  - AudioSystem, 48
- \_currentScene
  - AScenes, 27
- \_deqConnections
  - r\_type::net::AServer< T >, 44
- \_displayDaltonismChoice
  - AScenes, 27
- \_displayGameModeChoice
  - AScenes, 27
- \_displayKeyBindsChoice
  - AScenes, 28
- \_entityFactory
  - r\_type::net::AServer< T >, 44
- \_entityManager
  - AnimationSystem, 19
  - AutoFireSystem, 50
  - CollisionSystem, 55
  - MoveSystem, 102
  - r\_type::net::AServer< T >, 44
  - UpdateSystem, 126
- \_font
  - RenderSystem, 108
- \_gameParameters
  - r\_type::Level< T >, 99
- \_id
  - Entity, 61
- \_ip
  - AScenes, 28
- \_level
  - r\_type::net::AServer< T >, 44
- \_moveSystem
  - r\_type::Level< T >, 99
- \_nIDCounter
  - r\_type::net::AServer< T >, 45
- \_nbrOfPlayers
  - r\_type::net::AServer< T >, 44
- \_networkClient
  - Scenes, 113
- \_playerConnected
  - r\_type::net::AServer< T >, 45
- \_port
  - AScenes, 28
  - r\_type::net::AServer< T >, 45
- \_previousScene
  - AScenes, 28
- \_qMessagesIn
  - r\_type::net::AServer< T >, 45
- \_shooterEnemySpawnTime
  - r\_type::Level< T >, 99
- \_soundEffect
  - AudioSystem, 48
- \_spawnTimeMonsterThree
  - r\_type::Level< T >, 100
- \_tempBuffer
  - r\_type::net::AServer< T >, 45
- \_threadContext
  - r\_type::net::AServer< T >, 45
- \_window
  - RenderSystem, 108
  - Scenes, 113
  - UpdateSystem, 126
- ~AClient
  - r\_type::net::AClient< T >, 12
- ~AScenes
  - AScenes, 24
- ~AServer
  - r\_type::net::AServer< T >, 32
- ~IClient
  - r\_type::net::IClient< T >, 79
- ~IEntityFactory
  - IEntityFactory, 82
- ~IScenes
  - IScenes, 89
- ~ISystem
  - ISystem, 92
- ~Level
  - r\_type::Level< T >, 94
- ~Scenes
  - Scenes, 110
- ~Server
  - r\_type::net::Server, 115
- AbstractScenes, 11
- AClient
  - r\_type::net::AClient< T >, 12
- Actions
  - AScenes, 21
- ActionType
  - sound\_path.hpp, 153
- addComponent
  - ComponentManager, 56
- addEntity
  - r\_type::net::Client, 52
- ALLY
  - AScenes, 24
- AllyComponent, 16
- AllyMissileComponent, 16
- animateBasicMonster
  - AnimationSystem, 18
- animateEntity
  - r\_type::net::Client, 52
- animatePlayer
  - AnimationSystem, 18
- animateWeapon
  - AnimationSystem, 18
- animation\_component.hpp
  - operator!=, 137
- animation\_system.cpp
  - animationBasicMonsterFactory, 165
  - animationShipFactory, 165

- animationWeapon1Factory, 166
- operator!=, 166
- animation\_system.hpp
  - AnimationBasicMonster, 156
  - AnimationShip, 157
  - animationShipFactory, 157
  - AnimationWeapon1, 157
  - BASIC\_MONSTER\_1, 156
  - BASIC\_MONSTER\_2, 156
  - BASIC\_MONSTER\_3, 156
  - BASIC\_MONSTER\_4, 156
  - BASIC\_MONSTER\_5, 156
  - BASIC\_MONSTER\_6, 156
  - BASIC\_MONSTER\_7, 156
  - BASIC\_MONSTER\_DEFAULT, 156
  - operator!=, 158
  - SHIP\_DOWN, 157
  - SHIP\_FLIP\_DOWN, 157
  - SHIP\_FLIP\_UP, 157
  - SHIP\_STRAIT, 157
  - SHIP\_UP, 157
  - WEAPON\_1\_1, 157
  - WEAPON\_1\_2, 157
  - WEAPON\_1\_3, 157
  - WEAPON\_1\_4, 157
  - WEAPON\_1\_5, 157
  - WEAPON\_1\_DEFAULT, 157
- AnimationBasicMonster
  - animation\_system.hpp, 156
- animationBasicMonsterFactory
  - animation\_system.cpp, 165
- AnimationComponent, 16
  - AnimationComponent, 17
  - dimension, 17
  - offset, 17
- animationComponent
  - EntityInformation, 69
- AnimationEntities
  - AnimationSystem, 19
- AnimationShip
  - animation\_system.hpp, 157
- animationShipFactory
  - animation\_system.cpp, 165
  - animation\_system.hpp, 157
- AnimationSystem, 17
  - \_componentManager, 19
  - \_entityManager, 19
  - animateBasicMonster, 18
  - animatePlayer, 18
  - animateWeapon, 18
  - AnimationEntities, 19
  - AnimationSystem, 18
- AnimationUpdate
  - r\_type::Level< T >, 94
- AnimationWeapon1
  - animation\_system.hpp, 157
- animationWeapon1Factory
  - animation\_system.cpp, 166
- AScenes, 20
  - \_currentDaltonismMode, 27
  - \_currentGameMode, 27
  - \_currentScene, 27
  - \_displayDaltonismChoice, 27
  - \_displayGameModeChoice, 27
  - \_displayKeyBindsChoice, 28
  - \_ip, 28
  - \_port, 28
  - \_previousScene, 28
  - ~AScenes, 24
  - Actions, 21
  - ALLY, 24
  - AScenes, 24
  - BACKGROUND, 24
  - buttons, 28
  - DaltonismMode, 22
  - DEUTERANOPIA, 22
  - DOWN, 22
  - EASY, 23
  - ENEMY, 24
  - EXIT, 23
  - FILTER, 24
  - filter, 28
  - FIRE, 22
  - GAME\_LOOP, 23
  - GameMode, 22
  - getDaltonism, 24
  - getDisplayDaltonismChoice, 24
  - getDisplayGameModeChoice, 25
  - getDisplayKeyBindsChoice, 25
  - getIp, 25
  - getPort, 25
  - getPreviousScene, 25
  - HARD, 23
  - IN\_GAME\_MENU, 23
  - keyBinds, 28
  - LEFT, 22
  - MAIN\_MENU, 23
  - MEDIUM, 23
  - NORMAL, 22
  - OTHER, 24
  - PAUSE, 22
  - PLAYER, 24
  - POWER\_UP, 24
  - PROTANOPIA, 22
  - QUIT, 22
  - RIGHT, 22
  - Scene, 23
  - setDaltonism, 25
  - setDisplayDaltonismChoice, 26
  - setDisplayGameModeChoice, 26
  - setDisplayKeyBindsChoice, 26
  - setGameMode, 26
  - setIp, 26
  - setPort, 26
  - setScene, 27
  - SETTINGS\_MENU, 23

- SpriteType, 23
- TRITANOPIA, 22
- UI, 24
- UP, 22
- WEAPON, 24
- AServer
  - r\_type::net::AServer< T >, 32
- AudioManager, 46
  - getSoundBuffer, 46
  - soundBuffers, 46
- AudioSystem, 47
  - \_audioManager, 48
  - \_backgroundMusic, 48
  - \_currentMusicFilePath, 48
  - \_soundEffect, 48
  - AudioSystem, 47
  - playBackgroundMusic, 48
  - playSoundEffect, 48
  - stopBackgroundMusic, 48
- AutoFireSystem, 49
  - \_componentManager, 50
  - \_entityManager, 50
  - AutoFireSystem, 49
  - handleAutoFire, 49
- BACKGROUND
  - AScenes, 24
- Background
  - sound\_path.hpp, 153
- Background1
  - sprite\_path.hpp, 155
- Background2
  - sprite\_path.hpp, 155
- Background3
  - sprite\_path.hpp, 155
- BackgroundComponent, 50
- Bar
  - sprite\_path.hpp, 155
- BASIC\_MONSTER\_1
  - animation\_system.hpp, 156
- BASIC\_MONSTER\_2
  - animation\_system.hpp, 156
- BASIC\_MONSTER\_3
  - animation\_system.hpp, 156
- BASIC\_MONSTER\_4
  - animation\_system.hpp, 156
- BASIC\_MONSTER\_5
  - animation\_system.hpp, 156
- BASIC\_MONSTER\_6
  - animation\_system.hpp, 156
- BASIC\_MONSTER\_7
  - animation\_system.hpp, 156
- BASIC\_MONSTER\_DEFAULT
  - animation\_system.hpp, 156
- BasicMonster
  - IEntityFactory, 82
- BasicMonsterComponent, 50
- bind
  - BindComponent, 51
- BindComponent, 50
  - bind, 51
  - BindComponent, 51
  - isHovered, 51
- BlueLaserCrystal
  - sprite\_path.hpp, 155
- Boss
  - IEntityFactory, 82
  - sound\_path.hpp, 153
  - sprite\_path.hpp, 155
- BossBullet
  - sprite\_path.hpp, 155
- BossComponent, 51
- BossDeath
  - sound\_path.hpp, 153
- buttons
  - AScenes, 28
- canShoot
  - ShootComponent, 118
- categoryIds
  - TextDataComponent, 122
- categorySize
  - TextDataComponent, 122
- categoryTexts
  - TextDataComponent, 122
- charSize
  - TextDataComponent, 122
- checkCollision
  - CollisionSystem, 55
- CheckCollisionLogic
  - hitbox\_tmp.cpp, 163
- CheckEntityMovement
  - hitbox\_tmp.cpp, 164
  - hitbox\_tmp.hpp, 152
- CheckEntityPosition
  - hitbox\_tmp.cpp, 164
  - hitbox\_tmp.hpp, 152
- checkOffScreen
  - CollisionSystem, 55
- CIRCLE
  - movement\_component.hpp, 142
- CollisionSystem, 54
  - \_componentManager, 55
  - \_entityManager, 55
  - checkCollision, 55
  - checkOffScreen, 55
  - CollisionSystem, 54
- CollisionUpdate
  - r\_type::Level< T >, 95
- ComponentManager, 56
  - addComponent, 56
  - components, 58
  - getComponent, 57
  - getComponentMap, 57
  - removeEntityFromAllComponents, 58
  - removeEntityFromComponent, 58
- componentNotFound, 58
- what, 59

- components
  - ComponentManager, 58
- Connect
  - r\_type::net::AClient< T >, 13
  - r\_type::net::IClient< T >, 79
- cooldownTime
  - ShootComponent, 118
- creatable\_client\_object.hpp
  - CreatableClientObject, 147
  - NONE, 147
  - PLAYERMISSILE, 147
- CreatableClientObject, 59
  - creatable\_client\_object.hpp, 147
- createBackground
  - EntityFactory, 62
  - IEntityFactory, 83
- createBasicMonster
  - EntityFactory, 63
  - IEntityFactory, 83
- createButton
  - EntityFactory, 63
  - IEntityFactory, 84
- createDaltonismChoiceButtons
  - scenes.cpp, 134
- createEnemyMissile
  - EntityFactory, 64
  - IEntityFactory, 84
- createEntity
  - EntityManager, 70
- createFilter
  - EntityFactory, 65
- createForceMissile
  - EntityFactory, 65
  - IEntityFactory, 85
- createForceWeapon
  - EntityFactory, 65
  - IEntityFactory, 85
- createInfoBar
  - EntityFactory, 66
  - IEntityFactory, 85
- createKeyBindingButtons
  - scenes.cpp, 135
- createPlayer
  - EntityFactory, 66
  - IEntityFactory, 86
- createPlayerMissile
  - EntityFactory, 67
  - IEntityFactory, 86
- createPowerUpBlueLaserCrystal
  - EntityFactory, 67
  - IEntityFactory, 87
- createShooterEnemy
  - EntityFactory, 67
  - IEntityFactory, 87
- createSmallButton
  - EntityFactory, 68
  - IEntityFactory, 87
- DaltonismMode
  - AScenes, 22
- DEUTERANOPIA
  - AScenes, 22
- DIAGONAL
  - movement\_component.hpp, 142
- difficultyChoices
  - IScenes, 89
  - Scenes, 110
- dimension
  - AnimationComponent, 17
- Disconnect
  - r\_type::net::AClient< T >, 13
  - r\_type::net::IClient< T >, 80
- DOWN
  - AScenes, 22
  - input\_component.hpp, 141
- EASY
  - AScenes, 23
- ENEMY
  - AScenes, 24
- Enemy1
  - sprite\_path.hpp, 155
- Enemy2
  - sprite\_path.hpp, 155
- Enemy3
  - sprite\_path.hpp, 155
- Enemy4
  - sprite\_path.hpp, 155
- Enemy5
  - sprite\_path.hpp, 155
- Enemy6
  - sprite\_path.hpp, 155
- EnemyComponent, 59
- EnemyMissileComponent, 60
- EnemyType
  - IEntityFactory, 82
- entities
  - EntityManager, 71
- Entity, 60
  - \_id, 61
  - Entity, 60
  - getId, 61
- entity\_factory.cpp
  - operator<<, 162
- EntityFactory, 61
  - createBackground, 62
  - createBasicMonster, 63
  - createButton, 63
  - createEnemyMissile, 64
  - createFilter, 65
  - createForceMissile, 65
  - createForceWeapon, 65
  - createInfoBar, 66
  - createPlayer, 66
  - createPlayerMissile, 67
  - createPowerUpBlueLaserCrystal, 67
  - createShooterEnemy, 67
  - createSmallButton, 68

- EntityInformation, 69
  - animationComponent, 69
  - ratio, 69
  - spriteData, 69
  - uniqueID, 69
  - vPos, 69
- EntityManager, 70
  - createEntity, 70
  - entities, 71
  - entityNb, 72
  - getAllEntities, 70
  - getEntity, 71
  - removeEntity, 71
- entityNb
  - EntityManager, 72
- entityNotFound, 72
  - what, 72
- EXIT
  - AScenes, 23
- Explosion
  - sound\_path.hpp, 153
  - sprite\_path.hpp, 155
- failedToLoadFont, 73
  - what, 73
- failedToLoadSound, 73
  - what, 74
- failedToLoadTexture, 74
  - what, 74
- FILTER
  - AScenes, 24
- filter
  - AScenes, 28
- FIRE
  - AScenes, 22
- FireUpdate
  - r\_type::Level< T >, 95
- font\_path.cpp
  - FontFactory, 162
- font\_path.hpp
  - FontFactory, 150
  - FontPath, 150
  - MAIN, 150
  - NONE, 150
  - operator<<, 150
- FontFactory
  - font\_path.cpp, 162
  - font\_path.hpp, 150
- FontManager, 75
  - fonts, 75
  - getFont, 75
  - releaseFont, 75
- FontPath
  - font\_path.hpp, 150
- fontPath
  - TextDataComponent, 122
- fonts
  - FontManager, 75
- forceld
  - ForceMissileComponent, 76
- ForceMissileComponent, 76
  - forceld, 76
- FormatEntityInformation
  - r\_type::net::AServer< T >, 33
- FrontComponent, 76
  - FrontComponent, 77
  - targetId, 77
- GAME\_LOOP
  - AScenes, 23
- game\_text.cpp
  - GameTextFactory, 163
- game\_text.hpp
  - GameText, 151
  - GameTextFactory, 151
  - Lives, 151
  - NONE, 151
  - operator<<, 151
  - Score, 151
- gameLoop
  - IScenes, 89
  - Scenes, 110
- GameMode
  - AScenes, 22
- GameOver
  - sound\_path.hpp, 153
- GameText
  - game\_text.hpp, 151
- GameTextFactory
  - game\_text.cpp, 163
  - game\_text.hpp, 151
- getAllEntities
  - EntityManager, 70
- getClientById
  - r\_type::net::AServer< T >, 33
- getClientInfoBarId
  - r\_type::net::AServer< T >, 33
- getClientPlayerId
  - r\_type::net::AServer< T >, 33
- GetClock
  - r\_type::net::AServer< T >, 34
- getComponent
  - ComponentManager, 57
- GetComponentManager
  - r\_type::net::AServer< T >, 34
- GetComponentMap
  - ComponentManager, 57
- getConnection
  - r\_type::net::AClient< T >, 13
- getDaltonism
  - AScenes, 24
- getDisplayDaltonismChoice
  - AScenes, 24
- getDisplayGameModeChoice
  - AScenes, 25
- getDisplayKeyBindsChoice
  - AScenes, 25
- getEntity

- EntityManager, 71
- GetEntityFactory
  - r\_type::net::AServer< T >, 34
- GetEntityManager
  - r\_type::net::AServer< T >, 35
- getFont
  - FontManager, 75
- getId
  - Entity, 61
- getIp
  - AScenes, 25
- GetPlayerClientId
  - r\_type::net::AServer< T >, 35
- getPlayerId
  - r\_type::net::AClient< T >, 13
- getPort
  - AScenes, 25
- getPreviousScene
  - AScenes, 25
- getRenderWindow
  - IScenes, 90
  - Scenes, 110
- getSoundBuffer
  - AudioManager, 46
- getTexture
  - TextureManager, 123
- getWindowSize
  - r\_type::net::AClient< T >, 14
- h
  - HitboxComponent, 78
- handleAutoFire
  - AutoFireSystem, 49
- handleEvents
  - scenes.cpp, 135
- HandleMessage
  - Scenes, 110
- HARD
  - AScenes, 23
- health
  - HealthComponent, 77
- HealthComponent, 77
  - health, 77
  - max\_health, 77
- hitbox\_tmp.cpp
  - CheckCollisionLogic, 163
  - CheckEntityMovement, 164
  - CheckEntityPosition, 164
- hitbox\_tmp.hpp
  - CheckEntityMovement, 152
  - CheckEntityPosition, 152
- HitboxComponent, 78
  - h, 78
  - w, 78
- hitboxX
  - SpriteComponent, 119
- hitboxY
  - SpriteComponent, 119
- IClient
  - r\_type::net::IClient< T >, 79
- IEntityFactory, 81
  - ~IEntityFactory, 82
  - BasicMonster, 82
  - Boss, 82
  - createBackground, 83
  - createBasicMonster, 83
  - createButton, 84
  - createEnemyMissile, 84
  - createForceMissile, 85
  - createForceWeapon, 85
  - createInfoBar, 85
  - createPlayer, 86
  - createPlayerMissile, 86
  - createPowerUpBlueLaserCrystal, 87
  - createShooterEnemy, 87
  - createSmallButton, 87
  - EnemyType, 82
  - ShooterEnemy, 82
- IN\_GAME\_MENU
  - AScenes, 23
- Incoming
  - r\_type::net::AClient< T >, 14
  - r\_type::net::IClient< T >, 80
- index
  - MovementComponent, 100
- inGameMenu
  - IScenes, 90
  - Scenes, 111
- InitiateBackground
  - r\_type::net::AServer< T >, 35
- InitiateEnemyMissile
  - r\_type::net::AServer< T >, 35
- InitiatePlayer
  - r\_type::net::AServer< T >, 36
- InitiatePlayerMissile
  - r\_type::net::AServer< T >, 36
- InitiateWeaponForce
  - r\_type::net::AServer< T >, 36
- InitInfoBar
  - r\_type::net::AServer< T >, 37
- initInfoBar
  - r\_type::net::Client, 53
- input
  - InputComponent, 88
- input\_component.hpp
  - DOWN, 141
  - InputType, 141
  - LEFT, 141
  - NONE, 141
  - QUIT, 141
  - RIGHT, 141
  - SHOOT, 141
  - UP, 141
- InputComponent, 88
  - input, 88
- InputType

- input\_component.hpp, 141
- IScenes, 88
  - ~IScenes, 89
  - difficultyChoices, 89
  - gameLoop, 89
  - getRenderWindow, 90
  - inGameMenu, 90
  - mainMenu, 90
  - render, 90
  - settingsMenu, 90
  - shouldQuit, 91
- isClicked
  - OnClickComponent, 103
- IsConnected
  - r\_type::net::AClient< T >, 14
  - r\_type::net::IClient< T >, 80
- isHovered
  - BindComponent, 51
- isValidIPv4
  - main.cpp, 132
- isValidPort
  - main.cpp, 132, 133
- ISystem, 91
  - ~ISystem, 92
  - ISystem, 91
- keyBinds
  - AScenes, 28
- keyToString
  - keyToString.cpp, 131
  - scenes.hpp, 131
- keyToString.cpp
  - keyToString, 131
- labelComponent, 92
  - name, 92
  - x, 92
  - y, 92
- LEFT
  - AScenes, 22
  - input\_component.hpp, 141
- Level
  - r\_type::Level< T >, 94
- level
  - WeaponComponent, 128
- LevelOne
  - r\_type::Level< T >, 96
- Lives
  - game\_text.hpp, 151
- lives
  - UIEntityInformation, 124
- loopRunning
  - main.cpp, 134
- m\_connection
  - r\_type::net::AClient< T >, 15
- m\_context
  - r\_type::net::AClient< T >, 15
- m\_qMessagesIn
  - r\_type::net::AClient< T >, 15
- macros.hpp
  - SCREEN\_HEIGHT, 153
  - SCREEN\_WIDTH, 153
- MAIN
  - font\_path.hpp, 150
- main
  - main.cpp, 132, 133
- main.cpp
  - isValidIPv4, 132
  - isValidPort, 132, 133
  - loopRunning, 134
  - main, 132, 133
  - signal\_handler, 133
- MAIN\_MENU
  - AScenes, 23
- MainMenu
  - mainmenu.hpp, 129
- mainMenu
  - IScenes, 90
  - Scenes, 111
- mainmenu.hpp
  - MainMenu, 129
- max\_health
  - HealthComponent, 77
- MEDIUM
  - AScenes, 23
- MessageAll
  - r\_type::net::Client, 53
- MessageAllClients
  - r\_type::net::AServer< T >, 37
- MessageClient
  - r\_type::net::AServer< T >, 37
- Missile
  - sprite\_path.hpp, 155
- moveEntities
  - MoveSystem, 101
- moveEntity
  - r\_type::net::Client, 53
- movement\_component.hpp
  - CIRCLE, 142
  - DIAGONAL, 142
  - MovementType, 142
  - WIGGLE, 142
- MovementComponent, 100
  - index, 100
  - movementType, 100
- MovementType
  - movement\_component.hpp, 142
- movementType
  - MovementComponent, 100
- MoveSystem, 101
  - \_componentManager, 101
  - \_entityManager, 102
  - moveEntities, 101
  - MoveSystem, 101
- MoveUpdate
  - r\_type::Level< T >, 96

- name
  - labelComponent, 92
- nextShootTime
  - ShootComponent, 118
- NONE
  - creatable\_client\_object.hpp, 147
  - font\_path.hpp, 150
  - game\_text.hpp, 151
  - input\_component.hpp, 141
  - sound\_path.hpp, 153
  - sprite\_path.hpp, 155
- NORMAL
  - AScenes, 22
- offset
  - AnimationComponent, 17
  - OffsetComponent, 102
- OffsetComponent, 102
  - offset, 102
- onClick
  - OnClickComponent, 103
- OnClickComponent, 102
  - isClicked, 103
  - onClick, 103
  - OnClickComponent, 103
- OnClientConnect
  - r\_type::net::AServer< T >, 38
  - r\_type::net::Server, 115
- OnClientDisconnect
  - r\_type::net::AServer< T >, 38
  - r\_type::net::Server, 116
- OnClientValidated
  - r\_type::net::AServer< T >, 38
- OnMessage
  - r\_type::net::AServer< T >, 39
  - r\_type::net::Server, 116
- operator!=
  - animation\_component.hpp, 137
  - animation\_system.cpp, 166
  - animation\_system.hpp, 158
- operator<<
  - entity\_factory.cpp, 162
  - font\_path.hpp, 150
  - game\_text.hpp, 151
  - sprite\_data\_component.hpp, 145
  - sprite\_path.hpp, 155
- OTHER
  - AScenes, 24
- PAUSE
  - AScenes, 22
- PingServer
  - r\_type::net::Client, 53
- playBackgroundMusic
  - AudioSystem, 48
- PLAYER
  - AScenes, 24
- PlayerComponent, 103
- playerId
  - PlayerMissileComponent, 104
  - r\_type::net::AClient< T >, 15
  - WeaponComponent, 128
- playerIdNotFound, 104
  - what, 104
- PLAYERMISSILE
  - creatable\_client\_object.hpp, 147
- PlayerMissileComponent, 104
  - playerId, 104
- playSoundEffect
  - AudioSystem, 48
- PositionComponent, 105
  - PositionComponent, 105
  - x, 105
  - y, 105
- POWER\_UP
  - AScenes, 24
- PowerUp
  - sound\_path.hpp, 153
  - sprite\_path.hpp, 155
- PowerUpComponent, 106
- PROTANOPIA
  - AScenes, 22
- QUIT
  - AScenes, 22
  - input\_component.hpp, 141
- r\_type, 9
- r\_type::Level< T >, 93
  - \_animationSystem, 98
  - \_autoFireSystem, 99
  - \_basicMonsterSpawnTime, 99
  - \_collisionSystem, 99
  - \_gameParameters, 99
  - \_moveSystem, 99
  - \_shooterEnemySpawnTime, 99
  - \_spawnTimeMonsterThree, 100
  - ~Level, 94
  - AnimationUpdate, 94
  - CollisionUpdate, 95
  - FireUpdate, 95
  - Level, 94
  - LevelOne, 96
  - MoveUpdate, 96
  - SetGameParameters, 97
  - SetSystem, 97
  - SpawnEntity, 97
  - Update, 98
- r\_type::net, 9
- r\_type::net::AClient< T >, 11
  - ~AClient, 12
  - AClient, 12
  - Connect, 13
  - Disconnect, 13
  - getConnection, 13
  - getPlayerId, 13
  - getWindowSize, 14
  - Incoming, 14



- IsConnected, 14
- m\_connection, 15
- m\_context, 15
- m\_qMessagesIn, 15
- playerId, 15
- Send, 14
- setPlayerId, 15
- setWindowSize, 15
- thrContext, 16
- windowSize, 16
- r\_type::net::AServer< T >, 29
  - \_asioContext, 42
  - \_asioSocket, 42
  - \_background, 42
  - \_clientEndpoint, 43
  - \_clientInfoBarID, 43
  - \_clientPlayerID, 43
  - \_clock, 43
  - \_componentManager, 43
  - \_deqConnections, 44
  - \_entityFactory, 44
  - \_entityManager, 44
  - \_level, 44
  - \_nIDCounter, 45
  - \_nbrOfPlayers, 44
  - \_playerConnected, 45
  - \_port, 45
  - \_qMessagesIn, 45
  - \_tempBuffer, 45
  - \_threadContext, 45
  - ~AServer, 32
  - AServer, 32
  - FormatEntityInformation, 33
  - getClientById, 33
  - getClientInfoBarId, 33
  - getClientPlayerId, 33
  - GetClock, 34
  - GetComponentManager, 34
  - GetEntityFactory, 34
  - GetEntityManager, 35
  - GetPlayerClientId, 35
  - InitiateBackground, 35
  - InitiateEnemyMissile, 35
  - InitiatePlayer, 36
  - InitiatePlayerMissile, 36
  - InitiateWeaponForce, 36
  - InitInfoBar, 37
  - MessageAllClients, 37
  - MessageClient, 37
  - OnClientConnect, 38
  - OnClientDisconnect, 38
  - OnClientValidated, 38
  - OnMessage, 39
  - RemoveEntity, 39
  - RemoveInfoBar, 39
  - RemovePlayer, 39
  - SetClock, 40
  - Start, 40
  - Stop, 40
  - Update, 40
  - UpdateInfoBar, 41
  - UpdatePlayerPosition, 41
  - WaitForClientMessage, 42
- r\_type::net::IClient< T >, 78
  - ~IClient, 79
  - Connect, 79
  - Disconnect, 80
  - IClient, 79
  - Incoming, 80
  - IsConnected, 80
  - Send, 80
- r\_type::net::Server, 114
  - ~Server, 115
  - OnClientConnect, 115
  - OnClientDisconnect, 116
  - OnMessage, 116
  - Server, 115
- ratio
  - EntityInformation, 69
- rectangleShape
  - RectangleShapeComponent, 106
- RectangleShapeComponent, 106
  - rectangleShape, 106
  - RectangleShapeComponent, 106
- releaseFont
  - FontManager, 75
- releaseTexture
  - TextureManager, 123
- reloadFilter
  - scenes.cpp, 135
- RemoveEntity
  - r\_type::net::AServer< T >, 39
- removeEntity
  - EntityManager, 71
  - r\_type::net::Client, 53
- removeEntityFromAllComponents
  - ComponentManager, 58
- removeEntityFromComponent
  - ComponentManager, 58
- RemoveInfoBar
  - r\_type::net::AServer< T >, 39
- RemovePlayer
  - r\_type::net::AServer< T >, 39
- render
  - IScenes, 90
  - RenderSystem, 107
  - Scenes, 112

- RenderSystem, 107
  - \_componentManager, 108
  - \_font, 108
  - \_window, 108
  - render, 107
  - RenderSystem, 107
- RIGHT
  - AScenes, 22
  - input\_component.hpp, 141
- run
  - Scenes, 112
- scale
  - SpriteDataComponent, 120
- Scene
  - AScenes, 23
- Scenes, 108
  - \_networkClient, 113
  - \_window, 113
  - ~Scenes, 110
  - difficultyChoices, 110
  - gameLoop, 110
  - getRenderWindow, 110
  - HandleMessage, 110
  - inGameMenu, 111
  - mainMenu, 111
  - render, 112
  - run, 112
  - Scenes, 109
  - settingsMenu, 112
  - shouldQuit, 113
  - StopGameLoop, 113
- scenes.cpp
  - createDaltonismChoiceButtons, 134
  - createKeyBindingButtons, 135
  - handleEvents, 135
  - reloadFilter, 135
  - waitForKey, 136
- scenes.hpp
  - keyToString, 131
- Score
  - game\_text.hpp, 151
- score
  - ScoreComponent, 114
  - UIEntityInformation, 124
- ScoreComponent, 114
  - score, 114
- SCREEN\_HEIGHT
  - macros.hpp, 153
- SCREEN\_WIDTH
  - macros.hpp, 153
- Send
  - r\_type::net::AClient< T >, 14
  - r\_type::net::IClient< T >, 80
- Server
  - r\_type::net::Server, 115
- SetClock
  - r\_type::net::AServer< T >, 40
- setDaltonism
  - AScenes, 25
- setDisplayDaltonismChoice
  - AScenes, 26
- setDisplayGameModeChoice
  - AScenes, 26
- setDisplayKeyBindsChoice
  - AScenes, 26
- setGameMode
  - AScenes, 26
- SetGameParameters
  - r\_type::Level< T >, 97
- setIp
  - AScenes, 26
- setPlayerId
  - r\_type::net::AClient< T >, 15
- setPort
  - AScenes, 26
- setScene
  - AScenes, 27
- SetSystem
  - r\_type::Level< T >, 97
- SETTINGS\_MENU
  - AScenes, 23
- settingsMenu
  - IScenes, 90
  - Scenes, 112
- setWindowSize
  - r\_type::net::AClient< T >, 15
- shader
  - ShaderComponent, 117
- ShaderComponent, 116
  - shader, 117
  - ShaderComponent, 117
- Ship1
  - sprite\_path.hpp, 155
- Ship2
  - sprite\_path.hpp, 155
- Ship3
  - sprite\_path.hpp, 155
- Ship4
  - sprite\_path.hpp, 155
- SHIP\_DOWN
  - animation\_system.hpp, 157
- SHIP\_FLIP\_DOWN
  - animation\_system.hpp, 157
- SHIP\_FLIP\_UP
  - animation\_system.hpp, 157
- SHIP\_STRAIT
  - animation\_system.hpp, 157
- SHIP\_UP
  - animation\_system.hpp, 157
- SHOOT
  - input\_component.hpp, 141
- ShootComponent, 117
  - canShoot, 118
  - cooldownTime, 118
  - nextShootTime, 118
  - ShootComponent, 118

- ShooterEnemy
  - IFactory, 82
- Shot
  - sound\_path.hpp, 153
- shouldQuit
  - IScenes, 91
  - Scenes, 113
- signal\_handler
  - main.cpp, 133
- sound\_path.cpp
  - SoundFactory, 164
- sound\_path.hpp
  - ActionType, 153
  - Background, 153
  - Boss, 153
  - BossDeath, 153
  - Explosion, 153
  - GameOver, 153
  - NONE, 153
  - PowerUp, 153
  - Shot, 153
  - SoundFactory, 154
  - Win, 153
- soundBuffers
  - AudioManager, 46
- SoundFactory
  - sound\_path.cpp, 164
  - sound\_path.hpp, 154
- SpawnEntity
  - r\_type::Level< T >, 97
- sprite
  - SpriteComponent, 119
- sprite\_data\_component.hpp
  - operator<<, 145
- sprite\_path.cpp
  - SpriteFactory, 165
- sprite\_path.hpp
  - Background1, 155
  - Background2, 155
  - Background3, 155
  - Bar, 155
  - BlueLaserCrystal, 155
  - Boss, 155
  - BossBullet, 155
  - Enemy1, 155
  - Enemy2, 155
  - Enemy3, 155
  - Enemy4, 155
  - Enemy5, 155
  - Enemy6, 155
  - Explosion, 155
  - Missile, 155
  - NONE, 155
  - operator<<, 155
  - PowerUp, 155
  - Ship1, 155
  - Ship2, 155
  - Ship3, 155
  - Ship4, 155
  - SpriteFactory, 155
  - SpritePath, 154
  - Weapon, 155
- SpriteComponent, 118
  - hitboxX, 119
  - hitboxY, 119
  - sprite, 119
  - SpriteComponent, 119
  - type, 119
- spriteData
  - EntityInformation, 69
  - UIEntityInformation, 124
- SpriteDataComponent, 120
  - scale, 120
  - spritePath, 120
  - type, 120
- SpriteFactory
  - sprite\_path.cpp, 165
  - sprite\_path.hpp, 155
- SpritePath
  - sprite\_path.hpp, 154
- spritePath
  - SpriteDataComponent, 120
- SpriteType
  - AScenes, 23
- Start
  - r\_type::net::AServer< T >, 40
- Stop
  - r\_type::net::AServer< T >, 40
- stopBackgroundMusic
  - AudioSystem, 48
- StopGameLoop
  - Scenes, 113
- targetId
  - FrontComponent, 77
- text
  - TextComponent, 121
- TextComponent, 121
  - text, 121
  - TextComponent, 121
- textData
  - UIEntityInformation, 125
- TextDataComponent, 121
  - categoryIds, 122
  - categorySize, 122
  - categoryTexts, 122
  - charSize, 122
  - fontPath, 122
- TextureManager, 123
  - getTexture, 123
  - releaseTexture, 123
  - textures, 124
- textures
  - TextureManager, 124
- thrContext
  - r\_type::net::AClient< T >, 16
- TRITANOPIA

- AScenes, [22](#)
- type
  - SpriteComponent, [119](#)
  - SpriteDataComponent, [120](#)
- UI
  - AScenes, [24](#)
- UIEntityInformation, [124](#)
  - lives, [124](#)
  - score, [124](#)
  - spriteData, [124](#)
  - textData, [125](#)
  - uniqueID, [125](#)
- uniqueID
  - EntityInformation, [69](#)
  - UIEntityInformation, [125](#)
- UP
  - AScenes, [22](#)
  - input\_component.hpp, [141](#)
- Update
  - r\_type::Level< T >, [98](#)
  - r\_type::net::AServer< T >, [40](#)
- UpdateInfoBar
  - r\_type::net::AServer< T >, [41](#)
- updateInfoBar
  - r\_type::net::Client, [54](#)
- UpdatePlayerPosition
  - r\_type::net::AServer< T >, [41](#)
- updateSpritePositions
  - UpdateSystem, [126](#)
- UpdateSystem, [125](#)
  - \_componentManager, [126](#)
  - \_entityManager, [126](#)
  - \_window, [126](#)
  - updateSpritePositions, [126](#)
  - UpdateSystem, [125](#)
- VelocityComponent, [126](#)
  - x, [127](#)
  - y, [127](#)
- vf2d, [127](#)
  - x, [127](#)
  - y, [128](#)
- vPos
  - EntityInformation, [69](#)
- w
  - HitboxComponent, [78](#)
- WaitForClientMessage
  - r\_type::net::AServer< T >, [42](#)
- waitForKey
  - scenes.cpp, [136](#)
- WEAPON
  - AScenes, [24](#)
- Weapon
  - sprite\_path.hpp, [155](#)
- WEAPON\_1\_1
  - animation\_system.hpp, [157](#)
- WEAPON\_1\_2
  - animation\_system.hpp, [157](#)
- WEAPON\_1\_3
  - animation\_system.hpp, [157](#)
- WEAPON\_1\_4
  - animation\_system.hpp, [157](#)
- WEAPON\_1\_5
  - animation\_system.hpp, [157](#)
- WEAPON\_1\_DEFAULT
  - animation\_system.hpp, [157](#)
- WeaponComponent, [128](#)
  - level, [128](#)
  - playerId, [128](#)
  - WeaponComponent, [128](#)
- what
  - componentNotFound, [59](#)
  - entityNotFound, [72](#)
  - failedToLoadFont, [73](#)
  - failedToLoadSound, [74](#)
  - failedToLoadTexture, [74](#)
  - playerIdNotFound, [104](#)
- WIGGLE
  - movement\_component.hpp, [142](#)
- Win
  - sound\_path.hpp, [153](#)
- windowSize
  - r\_type::net::AClient< T >, [16](#)
- x
  - labelComponent, [92](#)
  - PositionComponent, [105](#)
  - VelocityComponent, [127](#)
  - vf2d, [127](#)
- y
  - labelComponent, [92](#)
  - PositionComponent, [105](#)
  - VelocityComponent, [127](#)
  - vf2d, [128](#)