

RTYPE

Generated by Doxygen 1.9.1



<b>1 Hierarchical Index</b>	<b>1</b>
1.1 Class Hierarchy	1
<b>2 Class Index</b>	<b>5</b>
2.1 Class List	5
<b>3 Class Documentation</b>	<b>9</b>
3.1 <code>rtype::AssetManager</code> Class Reference	9
3.1.1 Detailed Description	9
3.1.2 Member Function Documentation	9
3.1.2.1 <code>getInstance()</code>	10
3.1.2.2 <code>getSound()</code>	10
3.1.2.3 <code>getSoundBuffer()</code>	10
3.1.2.4 <code>getTexture()</code>	11
3.1.2.5 <code>playSound()</code>	11
3.2 <code>rtype::ecs::components::AsteroidId</code> Struct Reference	12
3.3 <code>engine::ecs::BaseComponentContainer</code> Class Reference	13
3.3.1 Detailed Description	13
3.3.2 Member Function Documentation	13
3.3.2.1 <code>reserve()</code>	13
3.3.2.2 <code>tryRemove()</code>	14
3.4 <code>rtype::ecs::components::BossAlien</code> Struct Reference	14
3.4.1 Detailed Description	15
3.5 <code>rtype::system::BossAlienShootSystem</code> Class Reference	15
3.5.1 Detailed Description	16
3.5.2 Constructor & Destructor Documentation	16
3.5.2.1 <code>BossAlienShootSystem()</code> [1/2]	16
3.5.2.2 <code>BossAlienShootSystem()</code> [2/2]	17
3.5.3 Member Function Documentation	17
3.5.3.1 <code>update()</code> [1/2]	17
3.5.3.2 <code>update()</code> [2/2]	17
3.6 <code>rtype::ecs::components::BossHive</code> Struct Reference	18
3.6.1 Detailed Description	19
3.7 <code>rtype::system::BossHiveShootSystem</code> Class Reference	19
3.7.1 Detailed Description	20
3.7.2 Constructor & Destructor Documentation	20
3.7.2.1 <code>BossHiveShootSystem()</code> [1/2]	20
3.7.2.2 <code>BossHiveShootSystem()</code> [2/2]	20
3.7.3 Member Function Documentation	21
3.7.3.1 <code>update()</code> [1/2]	21
3.7.3.2 <code>update()</code> [2/2]	21
3.8 <code>rtype::system::CleanEntitySystem</code> Class Reference	21
3.8.1 Detailed Description	22

3.8.2 Constructor & Destructor Documentation	23
3.8.2.1 CleanEntitySystem()	23
3.8.3 Member Function Documentation	23
3.8.3.1 update()	23
3.9 rtype::ClientGame Class Reference	24
3.9.1 Constructor & Destructor Documentation	25
3.9.1.1 ClientGame()	25
3.9.1.2 ~ClientGame()	25
3.9.2 Member Function Documentation	25
3.9.2.1 getFps()	25
3.9.2.2 getIp()	26
3.9.2.3 getPlayerID()	26
3.9.2.4 isOver()	26
3.10 client::clientInstance Class Reference	26
3.10.1 Detailed Description	27
3.10.2 Constructor & Destructor Documentation	27
3.10.2.1 clientInstance()	27
3.11 rtype::system::CollisionSystem Class Reference	27
3.11.1 Detailed Description	28
3.11.2 Constructor & Destructor Documentation	28
3.11.2.1 CollisionSystem()	28
3.11.3 Member Function Documentation	29
3.11.3.1 update()	29
3.12 engine::ecs::Component< T, Type > Class Template Reference	29
3.12.1 Detailed Description	29
3.13 engine::ecs::ComponentContainer< T, ComponentCount, SystemCount > Class Template Reference	30
3.13.1 Detailed Description	31
3.13.2 Member Function Documentation	31
3.13.2.1 add()	31
3.13.2.2 get() [1/2]	32
3.13.2.3 get() [2/2]	32
3.13.2.4 getOwner()	32
3.13.2.5 remove()	33
3.13.2.6 reserve()	33
3.13.2.7 tryRemove()	33
3.14 rtype::event::DeathEvent Class Reference	34
3.15 rtype::debug::Debugger Class Reference	35
3.16 rtype::ecs::components::DebugTag Struct Reference	35
3.16.1 Detailed Description	36
3.17 rtype::ecs::components::Enemy Struct Reference	36
3.17.1 Detailed Description	37
3.18 rtype::ecs::components::EnemyBeetle Struct Reference	38

3.19 <code>rtype::ecs::components::EnemyCrabId</code> Struct Reference . . . . .	39
3.20 <code>rtype::system::EnemyShootSystem</code> Class Reference . . . . .	40
3.20.1 Detailed Description . . . . .	41
3.20.2 Constructor & Destructor Documentation . . . . .	41
3.20.2.1 <code>EnemyShootSystem()</code> [1/2] . . . . .	41
3.20.2.2 <code>EnemyShootSystem()</code> [2/2] . . . . .	41
3.20.3 Member Function Documentation . . . . .	42
3.20.3.1 <code>update()</code> [1/2] . . . . .	42
3.20.3.2 <code>update()</code> [2/2] . . . . .	42
3.21 <code>rtype::ecs::components::Ennemild</code> Struct Reference . . . . .	43
3.22 <code>engine::ecs::EntityContainer&lt; ComponentCount, SystemCount &gt;</code> Class Template Reference . . . . .	44
3.22.1 Detailed Description . . . . .	44
3.22.2 Member Function Documentation . . . . .	44
3.22.2.1 <code>create()</code> . . . . .	45
3.22.2.2 <code>getBitset()</code> . . . . .	45
3.22.2.3 <code>getEntityToBitset()</code> . . . . .	45
3.22.2.4 <code>getEntityToComponent()</code> . . . . .	45
3.22.2.5 <code>getEntityToManagedEntity()</code> . . . . .	46
3.22.2.6 <code>remove()</code> . . . . .	46
3.22.2.7 <code>reserve()</code> . . . . .	46
3.23 <code>engine::ecs::EntityManager&lt; ComponentCount, SystemCount &gt;</code> Class Template Reference . . . . .	47
3.23.1 Detailed Description . . . . .	48
3.23.2 Member Function Documentation . . . . .	48
3.23.2.1 <code>addComponent()</code> . . . . .	48
3.23.2.2 <code>createEntity()</code> . . . . .	49
3.23.2.3 <code>createSystem()</code> . . . . .	49
3.23.2.4 <code>dispatchEvent()</code> . . . . .	49
3.23.2.5 <code>drawSystems()</code> . . . . .	50
3.23.2.6 <code>getComponent()</code> [1/2] . . . . .	50
3.23.2.7 <code>getComponent()</code> [2/2] . . . . .	50
3.23.2.8 <code>getComponents()</code> [1/2] . . . . .	51
3.23.2.9 <code>getComponents()</code> [2/2] . . . . .	51
3.23.2.10 <code>getOwner()</code> . . . . .	53
3.23.2.11 <code>hasComponent()</code> . . . . .	53
3.23.2.12 <code>hasComponents()</code> . . . . .	54
3.23.2.13 <code>registerComponent()</code> . . . . .	54
3.23.2.14 <code>removeComponent()</code> . . . . .	55
3.23.2.15 <code>removeEntity()</code> . . . . .	55
3.23.2.16 <code>reserve()</code> . . . . .	55
3.23.2.17 <code>updateSystems()</code> . . . . .	56
3.24 <code>rtype::EntityTemplate</code> Class Reference . . . . .	56
3.24.1 Detailed Description . . . . .	58

3.24.2 Constructor & Destructor Documentation	58
3.24.2.1 EntityTemplate() [1/2]	58
3.24.2.2 EntityTemplate() [2/2]	58
3.24.3 Member Function Documentation	58
3.24.3.1 createAsteroid() [1/2]	58
3.24.3.2 createAsteroid() [2/2]	60
3.24.3.3 createBossAlien() [1/2]	60
3.24.3.4 createBossAlien() [2/2]	61
3.24.3.5 createBossAlienBullet() [1/2]	61
3.24.3.6 createBossAlienBullet() [2/2]	61
3.24.3.7 createBossHive() [1/2]	62
3.24.3.8 createBossHive() [2/2]	62
3.24.3.9 createBossHiveBullet() [1/2]	63
3.24.3.10 createBossHiveBullet() [2/2]	63
3.24.3.11 createBottomBoundary() [1/2]	64
3.24.3.12 createBottomBoundary() [2/2]	64
3.24.3.13 createEnemyBeetle() [1/2]	64
3.24.3.14 createEnemyBeetle() [2/2]	66
3.24.3.15 createEnemyBullet() [1/2]	66
3.24.3.16 createEnemyBullet() [2/2]	67
3.24.3.17 createEnemyCrab() [1/2]	67
3.24.3.18 createEnemyCrab() [2/2]	67
3.24.3.19 createHealthBonus() [1/2]	68
3.24.3.20 createHealthBonus() [2/2]	68
3.24.3.21 createParallax() [1/2]	69
3.24.3.22 createParallax() [2/2]	69
3.24.3.23 createPlayer() [1/2]	70
3.24.3.24 createPlayer() [2/2]	70
3.24.3.25 createPlayerBullet() [1/2]	70
3.24.3.26 createPlayerBullet() [2/2]	71
3.24.3.27 createSpecialPlayerBullet() [1/2]	71
3.24.3.28 createSpecialPlayerBullet() [2/2]	72
3.24.3.29 createSpecialShootBonus() [1/2]	72
3.24.3.30 createSpecialShootBonus() [2/2]	72
3.24.3.31 createTopBoundary() [1/2]	73
3.24.3.32 createTopBoundary() [2/2]	73
3.25 engine::Event Class Reference	74
3.25.1 Detailed Description	74
3.26 rtype::Game Class Reference	74
3.26.1 Detailed Description	75
3.26.2 Constructor & Destructor Documentation	76
3.26.2.1 Game()	76

3.26.3 Member Function Documentation	77
3.26.3.1 getEntityManager()	77
3.26.3.2 getFps()	77
3.26.3.3 handleEvent()	77
3.26.3.4 isOver()	78
3.27 rtype::ecs::components::Health Struct Reference	78
3.27.1 Detailed Description	79
3.28 rtype::ecs::components::HealthBonusId Struct Reference	80
3.29 rtype::system::HealthSystem Class Reference	81
3.29.1 Detailed Description	82
3.29.2 Constructor & Destructor Documentation	82
3.29.2.1 HealthSystem() [1/2]	82
3.29.2.2 HealthSystem() [2/2]	82
3.29.3 Member Function Documentation	83
3.29.3.1 update() [1/2]	83
3.29.3.2 update() [2/2]	83
3.30 engine::ecs::components::Hitbox Struct Reference	83
3.30.1 Detailed Description	85
3.31 rtype::event::InputEvent Class Reference	85
3.32 rtype::LevelManager Class Reference	86
3.32.1 Detailed Description	86
3.32.2 Constructor & Destructor Documentation	86
3.32.2.1 LevelManager() [1/2]	86
3.32.2.2 LevelManager() [2/2]	87
3.32.3 Member Function Documentation	87
3.32.3.1 createBossLevel() [1/2]	87
3.32.3.2 createBossLevel() [2/2]	87
3.32.3.3 createLevel() [1/2]	88
3.32.3.4 createLevel() [2/2]	88
3.32.3.5 getLevel() [1/2]	88
3.32.3.6 getLevel() [2/2]	88
3.33 rtype::system::LevelSystem Class Reference	89
3.33.1 Detailed Description	90
3.33.2 Constructor & Destructor Documentation	90
3.33.2.1 LevelSystem()	90
3.33.3 Member Function Documentation	90
3.33.3.1 update()	90
3.34 rtype::ecs::components::LifeTime Struct Reference	91
3.34.1 Detailed Description	92
3.35 rtype::ecs::components::MoveComponent Struct Reference	92
3.35.1 Detailed Description	93
3.36 rtype::system::MovementSystem Class Reference	93

3.36.1 Detailed Description	94
3.36.2 Constructor & Destructor Documentation	94
3.36.2.1 MovementSystem()	94
3.36.3 Member Function Documentation	94
3.36.3.1 handleEvent()	94
3.37 rtype::NetworkEvent Struct Reference	95
3.37.1 Detailed Description	95
3.37.2 Member Data Documentation	95
3.37.2.1 id	95
3.37.2.2 key	95
3.37.2.3 objectType	95
3.37.2.4 pos	96
3.37.2.5 type	96
3.38 rtype::ecs::components::Parallax Struct Reference	96
3.38.1 Detailed Description	97
3.39 rtype::system::ParallaxSystem Class Reference	97
3.39.1 Detailed Description	98
3.39.2 Constructor & Destructor Documentation	98
3.39.2.1 ParallaxSystem()	98
3.39.3 Member Function Documentation	99
3.39.3.1 update()	99
3.40 rtype::ecs::components::ParralaxId Struct Reference	99
3.41 rtype::ecs::components::ParticleSpawner Struct Reference	100
3.41.1 Detailed Description	102
3.42 rtype::ecs::components::PlayerId Struct Reference	102
3.42.1 Detailed Description	103
3.43 rtype::ecs::components::PlayerShoot Struct Reference	103
3.43.1 Detailed Description	104
3.44 rtype::system::PlayerShootSystem Class Reference	104
3.44.1 Detailed Description	105
3.44.2 Constructor & Destructor Documentation	105
3.44.2.1 PlayerShootSystem() [1/2]	105
3.44.2.2 PlayerShootSystem() [2/2]	106
3.44.3 Member Function Documentation	106
3.44.3.1 handleEvent() [1/2]	106
3.44.3.2 handleEvent() [2/2]	106
3.45 rtype::QueueManager Class Reference	107
3.46 rtype::ScoreManager Class Reference	107
3.46.1 Detailed Description	108
3.46.2 Constructor & Destructor Documentation	108
3.46.2.1 ScoreManager() [1/2]	108
3.46.2.2 ScoreManager() [2/2]	108



3.46.3 Member Function Documentation	108
3.46.3.1 displayScore() [1/2]	108
3.46.3.2 displayScore() [2/2]	109
3.46.3.3 getScore() [1/2]	109
3.46.3.4 getScore() [2/2]	109
3.46.3.5 updateScore() [1/2]	109
3.46.3.6 updateScore() [2/2]	110
3.47 rtype::system::ScoreSystem Class Reference	110
3.47.1 Detailed Description	111
3.47.2 Constructor & Destructor Documentation	111
3.47.2.1 ScoreSystem()	111
3.47.3 Member Function Documentation	112
3.47.3.1 draw()	112
3.47.3.2 update()	112
3.48 rtype::ServerGame Class Reference	112
3.48.1 Detailed Description	113
3.48.2 Constructor & Destructor Documentation	114
3.48.2.1 ServerGame()	114
3.48.3 Member Function Documentation	114
3.48.3.1 getFps()	114
3.48.3.2 handleEvent()	114
3.48.3.3 isOver()	115
3.49 server::serverInstance Class Reference	115
3.49.1 Detailed Description	115
3.50 rtype::ecs::components::ShootBonusId Struct Reference	116
3.51 rtype::system::SpawnBonusSystem Class Reference	117
3.51.1 Detailed Description	118
3.51.2 Constructor & Destructor Documentation	118
3.51.2.1 SpawnBonusSystem() [1/2]	118
3.51.2.2 SpawnBonusSystem() [2/2]	118
3.51.3 Member Function Documentation	119
3.51.3.1 update() [1/2]	119
3.51.3.2 update() [2/2]	119
3.52 rtype::ecs::components::Sprite Struct Reference	119
3.52.1 Detailed Description	121
3.53 rtype::ecs::components::SpriteAnimation Struct Reference	121
3.53.1 Detailed Description	122
3.54 rtype::system::SpriteAnimationSystem Class Reference	122
3.54.1 Detailed Description	123
3.54.2 Constructor & Destructor Documentation	123
3.54.2.1 SpriteAnimationSystem()	123
3.54.3 Member Function Documentation	124

3.54.3.1 update()	124
3.55 rtype::system::SpriteSystem Class Reference	124
3.55.1 Detailed Description	125
3.55.2 Constructor & Destructor Documentation	125
3.55.2.1 SpriteSystem()	125
3.55.3 Member Function Documentation	126
3.55.3.1 draw()	126
3.56 rtype::ecs::components::State Struct Reference	126
3.56.1 Detailed Description	127
3.57 engine::ecs::System< ComponentCount, SystemCount > Class Template Reference	127
3.57.1 Detailed Description	128
3.57.2 Member Function Documentation	128
3.57.2.1 draw()	128
3.57.2.2 getManagedEntities()	129
3.57.2.3 handleEvent()	129
3.57.2.4 onManagedEntityAdded()	129
3.57.2.5 onManagedEntityRemoved()	129
3.57.2.6 setRequirements()	130
3.57.2.7 update()	130
3.58 engine::ecs::components::Transform Struct Reference	131
3.58.1 Detailed Description	132
3.59 rtype::system::TransformSystem Class Reference	132
3.59.1 Detailed Description	133
3.59.2 Constructor & Destructor Documentation	133
3.59.2.1 TransformSystem()	133
3.59.3 Member Function Documentation	133
3.59.3.1 update()	133
3.60 engine::ecs::components::Velocity Struct Reference	134
3.60.1 Detailed Description	135

## Chapter 1

# Hierarchical Index

### 1.1 Class Hierarchy

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

rtype::AssetManager . . . . .	9
engine::ecs::BaseComponentContainer . . . . .	13
engine::ecs::ComponentContainer< T, ComponentCount, SystemCount > . . . . .	30
client::clientInstance . . . . .	26
engine::ecs::Component< T, Type > . . . . .	29
engine::ecs::Component< AsteroidId, ComponentType::AsteroidId > . . . . .	29
rtype::ecs::components::AsteroidId . . . . .	12
rtype::ecs::components::AsteroidId . . . . .	12
engine::ecs::Component< BossAlien, ComponentType::BossAlien > . . . . .	29
rtype::ecs::components::BossAlien . . . . .	14
rtype::ecs::components::BossAlien . . . . .	14
engine::ecs::Component< BossHive, ComponentType::BossHive > . . . . .	29
rtype::ecs::components::BossHive . . . . .	18
rtype::ecs::components::BossHive . . . . .	18
engine::ecs::Component< DebugTag, ComponentType::DebugTag > . . . . .	29
rtype::ecs::components::DebugTag . . . . .	35
rtype::ecs::components::DebugTag . . . . .	35
engine::ecs::Component< Enemy, ComponentType::Enemy > . . . . .	29
rtype::ecs::components::Enemy . . . . .	36
rtype::ecs::components::Enemy . . . . .	36
engine::ecs::Component< EnemyBeetleId, ComponentType::EnemyBeetleId > . . . . .	29
rtype::ecs::components::EnemyBeetleId . . . . .	38
rtype::ecs::components::EnemyBeetleId . . . . .	38
engine::ecs::Component< EnemyCrabId, ComponentType::EnemyCrabId > . . . . .	29
rtype::ecs::components::EnemyCrabId . . . . .	39
rtype::ecs::components::EnemyCrabId . . . . .	39
engine::ecs::Component< Ennemild, ComponentType::Ennemild > . . . . .	29
rtype::ecs::components::Ennemild . . . . .	43
rtype::ecs::components::Ennemild . . . . .	43
engine::ecs::Component< Health, ComponentType::Health > . . . . .	29
rtype::ecs::components::Health . . . . .	78
rtype::ecs::components::Health . . . . .	78

engine::ecs::Component< HealthBonusId, ComponentType::HealthBonusId > . . . . .	29
rtype::ecs::components::HealthBonusId . . . . .	80
rtype::ecs::components::HealthBonusId . . . . .	80
engine::ecs::Component< Hitbox, ComponentType::Hitbox > . . . . .	29
engine::ecs::components::Hitbox . . . . .	83
engine::ecs::Component< LifeTime, ComponentType::LifeTime > . . . . .	29
rtype::ecs::components::LifeTime . . . . .	91
engine::ecs::Component< MoveComponent, ComponentType::MoveComponent > . . . . .	29
rtype::ecs::components::MoveComponent . . . . .	92
rtype::ecs::components::MoveComponent . . . . .	92
engine::ecs::Component< Parallax, ComponentType::Parallax > . . . . .	29
rtype::ecs::components::Parallax . . . . .	96
rtype::ecs::components::Parallax . . . . .	96
engine::ecs::Component< ParralaxId, ComponentType::ParralaxId > . . . . .	29
rtype::ecs::components::ParralaxId . . . . .	99
rtype::ecs::components::ParralaxId . . . . .	99
engine::ecs::Component< ParticleSpawner, ComponentType::ParticleSpawner > . . . . .	29
rtype::ecs::components::ParticleSpawner . . . . .	100
engine::ecs::Component< PlayerId, ComponentType::PlayerId > . . . . .	29
rtype::ecs::components::PlayerId . . . . .	102
rtype::ecs::components::PlayerId . . . . .	102
engine::ecs::Component< PlayerShoot, ComponentType::PlayerShoot > . . . . .	29
rtype::ecs::components::PlayerShoot . . . . .	103
rtype::ecs::components::PlayerShoot . . . . .	103
engine::ecs::Component< ShootBonusId, ComponentType::ShootBonusId > . . . . .	29
rtype::ecs::components::ShootBonusId . . . . .	116
rtype::ecs::components::ShootBonusId . . . . .	116
engine::ecs::Component< Sprite, ComponentType::Sprite > . . . . .	29
rtype::ecs::components::Sprite . . . . .	119
rtype::ecs::components::Sprite . . . . .	119
engine::ecs::Component< SpriteAnimation, ComponentType::SpriteAnimation > . . . . .	29
rtype::ecs::components::SpriteAnimation . . . . .	121
rtype::ecs::components::SpriteAnimation . . . . .	121
engine::ecs::Component< State, ComponentType::State > . . . . .	29
rtype::ecs::components::State . . . . .	126
rtype::ecs::components::State . . . . .	126
engine::ecs::Component< Transform, ComponentType::Transform > . . . . .	29
engine::ecs::components::Transform . . . . .	131
engine::ecs::Component< Velocity, ComponentType::Velocity > . . . . .	29
engine::ecs::components::Velocity . . . . .	134
rtype::debug::Debugger . . . . .	35
engine::ecs::EntityContainer< ComponentCount, SystemCount > . . . . .	44
engine::ecs::EntityManager< ComponentCount, SystemCount > . . . . .	47
engine::ecs::EntityManager< rtype::config::ComponentCount, rtype::config::SystemCount > . . . . .	47
rtype::EntityTemplate . . . . .	56
engine::Event . . . . .	74
rtype::event::DeathEvent . . . . .	34
rtype::event::InputEvent . . . . .	85
rtype::Game . . . . .	74
rtype::ClientGame . . . . .	24
rtype::ServerGame . . . . .	112
rtype::LevelManager . . . . .	86
rtype::NetworkEvent . . . . .	95

rtype::QueueManager . . . . .	107
rtype::ScoreManager . . . . .	107
server::serverInstance . . . . .	115
engine::ecs::System< ComponentCount, SystemCount > . . . . .	127
engine::ecs::System< rtype::config::ComponentCount, rtype::config::SystemCount > . . . . .	127
rtype::system::BossAlienShootSystem . . . . .	15
rtype::system::BossAlienShootSystem . . . . .	15
rtype::system::BossHiveShootSystem . . . . .	19
rtype::system::BossHiveShootSystem . . . . .	19
rtype::system::CleanEntitySystem . . . . .	21
rtype::system::CollisionSystem . . . . .	27
rtype::system::EnemyShootSystem . . . . .	40
rtype::system::EnemyShootSystem . . . . .	40
rtype::system::HealthSystem . . . . .	81
rtype::system::HealthSystem . . . . .	81
rtype::system::LevelSystem . . . . .	89
rtype::system::MovementSystem . . . . .	93
rtype::system::ParallaxSystem . . . . .	97
rtype::system::PlayerShootSystem . . . . .	104
rtype::system::PlayerShootSystem . . . . .	104
rtype::system::ScoreSystem . . . . .	110
rtype::system::SpawnBonusSystem . . . . .	117
rtype::system::SpawnBonusSystem . . . . .	117
rtype::system::SpriteAnimationSystem . . . . .	122
rtype::system::SpriteSystem . . . . .	124
rtype::system::TransformSystem . . . . .	132



## Chapter 2

# Class Index

### 2.1 Class List

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

<code>rtype::AssetManager</code>	
<code>AssetManager</code> class which allows you to load the assets only once	9
<code>rtype::ecs::components::AsteroidId</code>	12
<code>engine::ecs::BaseComponentContainer</code>	
<code>BaseComponentContainer</code> Interface used to store components	13
<code>rtype::ecs::components::BossAlien</code>	
<code>BossAlien</code> struct which contains the boss alien of the entity	14
<code>rtype::system::BossAlienShootSystem</code>	
<code>EnemyShoot</code> system class used to let the enemy shoot	15
<code>rtype::ecs::components::BossHive</code>	
<code>BossHive</code> struct which contains the boss hive of the entity	18
<code>rtype::system::BossHiveShootSystem</code>	
<code>EnemyShoot</code> system class used to let the enemy shoot	19
<code>rtype::system::CleanEntitySystem</code>	
<code>Clean</code> entity system class to remove entities	21
<code>rtype::ClientGame</code>	24
<code>client::clientInstance</code>	
Instance of the client containing a game	26
<code>rtype::system::CollisionSystem</code>	
<code>Collision</code> system class to check collision between entities	27
<code>engine::ecs::Component&lt; T, Type &gt;</code>	
<code>Component</code> class which is the base class for all components	29
<code>engine::ecs::ComponentContainer&lt; T, ComponentCount, SystemCount &gt;</code>	
<code>ComponentContainer</code> class used to store components	30
<code>rtype::event::DeathEvent</code>	34
<code>rtype::debug::Debugger</code>	35
<code>rtype::ecs::components::DebugTag</code>	
<code>DebugTag</code> struct which contains the debug tag of the entity	35
<code>rtype::ecs::components::Enemy</code>	
<code>Enemy</code> struct which contains the enemy of the entity	36
<code>rtype::ecs::components::EnemyBeetleId</code>	38
<code>rtype::ecs::components::EnemyCrabId</code>	39
<code>rtype::system::EnemyShootSystem</code>	
<code>EnemyShoot</code> system class used to let the enemy shoot	40
<code>rtype::ecs::components::Ennemild</code>	43

<a href="#">engine::ecs::EntityContainer&lt; ComponentCount, SystemCount &gt;</a>	
EntityContainer class used to store entities	44
<a href="#">engine::ecs::EntityManager&lt; ComponentCount, SystemCount &gt;</a>	
Entity manager class. It is used to create entities, add components to them and create systems with the entity linked	47
<a href="#">rtype::EntityTemplate</a>	
EntityTemplate class which allows you to create entities	56
<a href="#">engine::Event</a>	
Event class which is the base class for all events	74
<a href="#">rtype::Game</a>	
Game's base, has to be completed in a child class	74
<a href="#">rtype::ecs::components::Health</a>	
Health struct which contains the health of the entity	78
<a href="#">rtype::ecs::components::HealthBonusId</a>	80
<a href="#">rtype::system::HealthSystem</a>	
Health system class to check if an entity is dead or not	81
<a href="#">engine::ecs::components::Hitbox</a>	
Hitbox component	83
<a href="#">rtype::event::InputEvent</a>	85
<a href="#">rtype::LevelManager</a>	
Level manager class to manage the creation of levels	86
<a href="#">rtype::system::LevelSystem</a>	
LevelSystem class which is used to have the operation of stages	89
<a href="#">rtype::ecs::components::LifeTime</a>	
LifeTime struct which contains the life time info of the entity	91
<a href="#">rtype::ecs::components::MoveComponent</a>	
MoveComponent struct which contains the movement of the entity	92
<a href="#">rtype::system::MovementSystem</a>	
Movement system class used to move the entities	93
<a href="#">rtype::NetworkEvent</a>	
Package structure	95
<a href="#">rtype::ecs::components::Parallax</a>	
Parallax struct which contains the parallax of the entity	96
<a href="#">rtype::system::ParallaxSystem</a>	
ParallaxSystem used to spin the parallax	97
<a href="#">rtype::ecs::components::ParralaxId</a>	99
<a href="#">rtype::ecs::components::ParticleSpawner</a>	
ParticleSpawner struct which contains the particle spawner info of the entity	100
<a href="#">rtype::ecs::components::PlayerId</a>	
PlayerId struct which contains the player id	102
<a href="#">rtype::ecs::components::PlayerShoot</a>	
PlayerShoot struct which contains the player shoot	103
<a href="#">rtype::system::PlayerShootSystem</a>	
PlayerShoot system class used to let the player shoot	104
<a href="#">rtype::QueueManager</a>	107
<a href="#">rtype::ScoreManager</a>	
Score manager class to manage the score	107
<a href="#">rtype::system::ScoreSystem</a>	
ScoreSystem class to have the score the players	110
<a href="#">rtype::ServerGame</a>	
Instance of the server version of the game (without graphics but having all powers)	112
<a href="#">server::serverInstance</a>	
Instance of server handling a game	115
<a href="#">rtype::ecs::components::ShootBonusId</a>	116
<a href="#">rtype::system::SpawnBonusSystem</a>	
Clean entity system class to remove entities	117
<a href="#">rtype::ecs::components::Sprite</a>	
Sprite component	119



<a href="#">rtype::ecs::components::SpriteAnimation</a>	
<a href="#">SpriteAnimation</a> component . . . . .	121
<a href="#">rtype::system::SpriteAnimationSystem</a>	
<a href="#">SpriteAnimation</a> system class used to animate sprites . . . . .	122
<a href="#">rtype::system::SpriteSystem</a>	
<a href="#">SpriteSystem</a> class used to draw the sprites of the game . . . . .	124
<a href="#">rtype::ecs::components::State</a>	
<a href="#">State</a> struct which contains the state of the entity . . . . .	126
<a href="#">engine::ecs::System&lt; ComponentCount, SystemCount &gt;</a>	
<a href="#">BaseSystem</a> Interface used to store systems . . . . .	127
<a href="#">engine::ecs::components::Transform</a>	
<a href="#">Transform</a> component . . . . .	131
<a href="#">rtype::system::TransformSystem</a>	
<a href="#">TransformSystem</a> class used to update the position of the entities . . . . .	132
<a href="#">engine::ecs::components::Velocity</a>	
<a href="#">Velocity</a> component . . . . .	134



## Chapter 3

# Class Documentation

### 3.1 rtype::AssetManager Class Reference

[AssetManager](#) class which allows you to load the assets only once.

```
#include <AssetManager.hpp>
```

#### Public Member Functions

- sf::Texture & [getTexture](#) (const std::string &textureName)  
*Get the Texture object.*
- sf::SoundBuffer & [getSoundBuffer](#) (const std::string &soundName)  
*Get the Sound Buffer object.*
- sf::Sound & [getSound](#) (const std::string &soundName)  
*Get the Sound object.*
- void [playSound](#) (const std::string &soundName)  
*Play the sound.*

#### Static Public Member Functions

- static [AssetManager](#) & [getInstance](#) ()  
*Get the Instance object.*

#### 3.1.1 Detailed Description

[AssetManager](#) class which allows you to load the assets only once.

#### 3.1.2 Member Function Documentation

### 3.1.2.1 getInstance()

```
static AssetManager& rtype::AssetManager::getInstance ( ) [inline], [static]
```

Get the Instance object.

Returns

[AssetManager&](#)

### 3.1.2.2 getSound()

```
sf::Sound& rtype::AssetManager::getSound (
    const std::string & soundName ) [inline]
```

Get the Sound object.

Parameters

<i>soundName</i>	
------------------	--

Returns

sf::Sound&

### 3.1.2.3 getSoundBuffer()

```
sf::SoundBuffer& rtype::AssetManager::getSoundBuffer (
    const std::string & soundName ) [inline]
```

Get the Sound Buffer object.

Parameters

<i>soundName</i>	
------------------	--

Returns

sf::SoundBuffer&

### 3.1.2.4 getTexture()

```
sf::Texture& rtype::AssetManager::getTexture (
    const std::string & textureName ) [inline]
```

Get the Texture object.

#### Parameters

<i>textureName</i>	
--------------------	--

#### Returns

sf::Texture&

### 3.1.2.5 playSound()

```
void rtype::AssetManager::playSound (
    const std::string & soundName ) [inline]
```

Play the sound.

#### Parameters

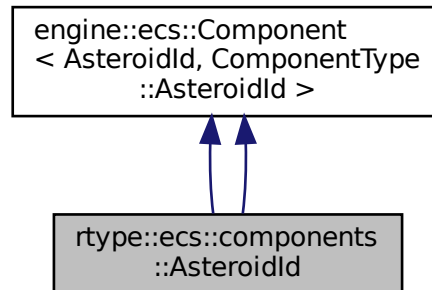
<i>soundName</i>	
------------------	--

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

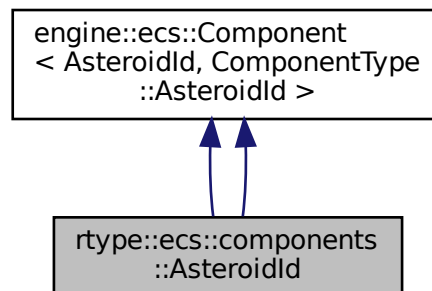
- rtype-client/include/AssetManager.hpp

## 3.2 rtype::ecs::components::AsteroidId Struct Reference

Inheritance diagram for rtype::ecs::components::AsteroidId:



Collaboration diagram for rtype::ecs::components::AsteroidId:



### Public Member Functions

- **AsteroidId** (int asteroidId)
- **AsteroidId** (int asteroidId)

### Public Attributes

- int id

## Additional Inherited Members

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

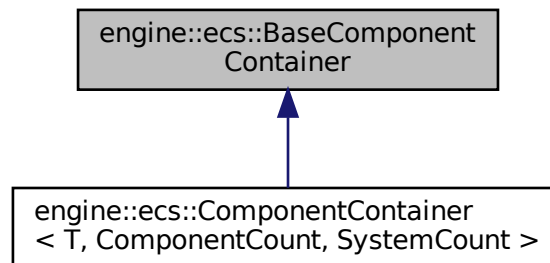
- rtype-client/include/Components.hpp

## 3.3 engine::ecs::BaseComponentContainer Class Reference

[BaseComponentContainer](#) Interface used to store components.

```
#include <ComponentContainer.hpp>
```

Inheritance diagram for engine::ecs::BaseComponentContainer:



## Public Member Functions

- virtual void [reserve](#) (std::size\_t size)=0  
*Reserve the size of the container.*
- virtual bool [tryRemove](#) (Entity entity)=0  
*Try to remove an entity from the container.*

### 3.3.1 Detailed Description

[BaseComponentContainer](#) Interface used to store components.

### 3.3.2 Member Function Documentation

#### 3.3.2.1 reserve()

```
virtual void engine::ecs::BaseComponentContainer::reserve (
    std::size_t size ) [pure virtual]
```

Reserve the size of the container.

## Parameters

<i>size</i>	
-------------	--

Implemented in [engine::ecs::ComponentContainer< T, ComponentCount, SystemCount >](#).

### 3.3.2.2 tryRemove()

```
virtual bool engine::ecs::BaseComponentContainer::tryRemove (
    Entity entity ) [pure virtual]
```

Try to remove an entity from the container.

## Parameters

<i>entity</i>	
---------------	--

## Returns

true  
false

Implemented in [engine::ecs::ComponentContainer< T, ComponentCount, SystemCount >](#).

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

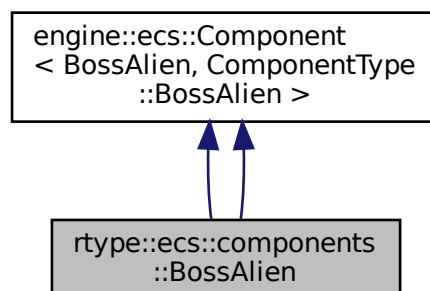
- engine/include/ecs/ComponentContainer.hpp

## 3.4 rtype::ecs::components::BossAlien Struct Reference

[BossAlien](#) struct which contains the boss alien of the entity.

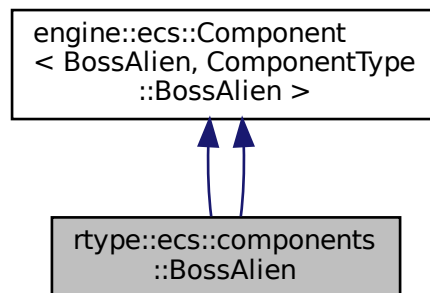
```
#include <Components.hpp>
```

Inheritance diagram for rtype::ecs::components::BossAlien:





Collaboration diagram for rtype::ecs::components::BossAlien:



## Additional Inherited Members

### 3.4.1 Detailed Description

[BossAlien](#) struct which contains the boss alien of the entity.

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

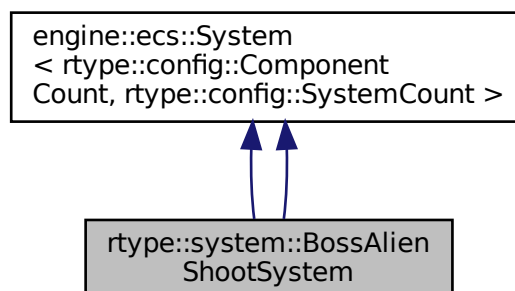
- `rtype-client/include/Components.hpp`

## 3.5 rtype::system::BossAlienShootSystem Class Reference

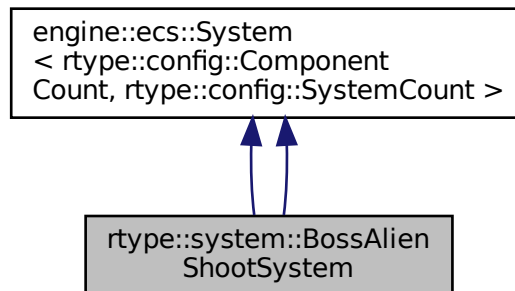
EnemyShoot system class used to let the enemy shoot.

```
#include <BossAlienShootSystem.hpp>
```

Inheritance diagram for rtype::system::BossAlienShootSystem:



Collaboration diagram for `rtype::system::BossAlienShootSystem`:



## Public Member Functions

- `BossAlienShootSystem` (`engine::ecs::EntityManager` < `rtype::config::ComponentCount`, `rtype::config::SystemCount` > &`entityManager`)  
Construct a new *BossAlien Shoot System* object.
- `void update` (`const float dt`)  
Update the *BossAlien Shoot* system class, to allow enemies to shoot.
- `BossAlienShootSystem` (`engine::ecs::EntityManager` < `rtype::config::ComponentCount`, `rtype::config::SystemCount` > &`entityManager`)  
Construct a new *BossAlien Shoot System* object.
- `void update` (`const float dt`)  
Update the *BossAlien Shoot* system class, to allow enemies to shoot.

## Additional Inherited Members

### 3.5.1 Detailed Description

EnemyShoot system class used to let the enemy shoot.

### 3.5.2 Constructor & Destructor Documentation

#### 3.5.2.1 BossAlienShootSystem() [1/2]

```

rtype::system::BossAlienShootSystem::BossAlienShootSystem (
    engine::ecs::EntityManager < rtype::config::ComponentCount, rtype::config::SystemCount > & entityManager ) [inline]
  
```

Construct a new *BossAlien Shoot System* object.

## Parameters

<i>entityManager</i>	
----------------------	--

**3.5.2.2 BossAlienShootSystem()** [2/2]

```
rtype::system::BossAlienShootSystem::BossAlienShootSystem (
    engine::ecs::EntityManager< rtype::config::ComponentCount, rtype::config::SystemCount > & entityManager ) [inline]
```

Construct a new BossAlien Shoot System object.

## Parameters

<i>entityManager</i>	
----------------------	--

**3.5.3 Member Function Documentation****3.5.3.1 update()** [1/2]

```
void rtype::system::BossAlienShootSystem::update (
    const float dt ) [inline], [virtual]
```

Update the BossAlien Shoot system class, to allow enemies to shoot.

## Parameters

<i>dt</i>	The Delta Time for the shoot of enemies
-----------	---

Reimplemented from [engine::ecs::System< rtype::config::ComponentCount, rtype::config::SystemCount >](#).

**3.5.3.2 update()** [2/2]

```
void rtype::system::BossAlienShootSystem::update (
    const float dt ) [inline], [virtual]
```

Update the BossAlien Shoot system class, to allow enemies to shoot.

## Parameters

<i>dt</i>	The Delta Time for the shoot of enemies
-----------	---

Reimplemented from [engine::ecs::System< rtype::config::ComponentCount, rtype::config::SystemCount >](#).

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

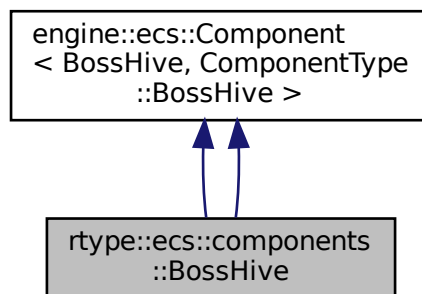
- [rtype-client/include/systems/BossAlienShootSystem.hpp](#)

### 3.6 rtype::ecs::components::BossHive Struct Reference

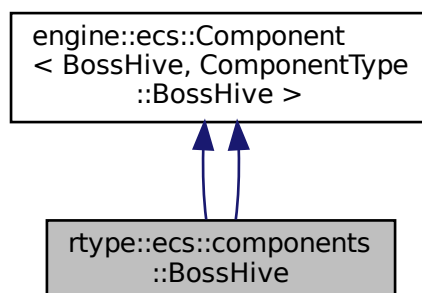
[BossHive](#) struct which contains the boss hive of the entity.

```
#include <Components.hpp>
```

Inheritance diagram for rtype::ecs::components::BossHive:



Collaboration diagram for rtype::ecs::components::BossHive:



## Additional Inherited Members

### 3.6.1 Detailed Description

[BossHive](#) struct which contains the boss hive of the entity.

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

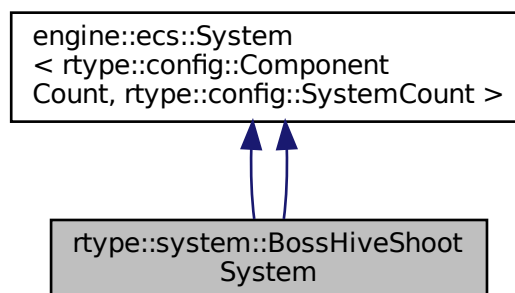
- rtype-client/include/Components.hpp

## 3.7 rtype::system::BossHiveShootSystem Class Reference

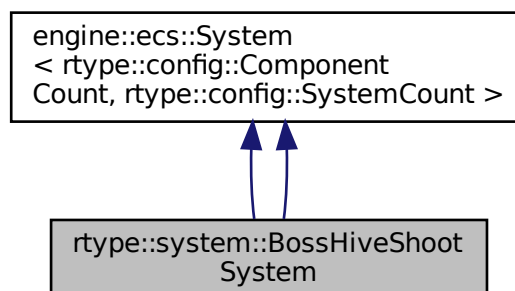
EnemyShoot system class used to let the enemy shoot.

```
#include <BossHiveShootSystem.hpp>
```

Inheritance diagram for rtype::system::BossHiveShootSystem:



Collaboration diagram for rtype::system::BossHiveShootSystem:



## Public Member Functions

- [BossHiveShootSystem](#) ([engine::ecs::EntityManager](#)< [rtype::config::ComponentCount](#), [rtype::config::↔SystemCount](#) > &[entityManager](#))  
Construct a new BossHive Shoot System object.
- void [update](#) (const float dt)  
Update the BossHive Shoot system class, to allow enemies to shoot.
- [BossHiveShootSystem](#) ([engine::ecs::EntityManager](#)< [rtype::config::ComponentCount](#), [rtype::config::↔SystemCount](#) > &[entityManager](#))  
Construct a new BossHive Shoot System object.
- void [update](#) (const float dt)  
Update the BossHive Shoot system class, to allow enemies to shoot.

## Additional Inherited Members

### 3.7.1 Detailed Description

EnemyShoot system class used to let the enemy shoot.

### 3.7.2 Constructor & Destructor Documentation

#### 3.7.2.1 BossHiveShootSystem() [1/2]

```
rtype::system::BossHiveShootSystem::BossHiveShootSystem (
    engine::ecs::EntityManager< rtype::config::ComponentCount, rtype::config::↔SystemCount > & entityManager ) [inline]
```

Construct a new BossHive Shoot System object.

#### Parameters

<a href="#">entityManager</a>	
-------------------------------	--

#### 3.7.2.2 BossHiveShootSystem() [2/2]

```
rtype::system::BossHiveShootSystem::BossHiveShootSystem (
    engine::ecs::EntityManager< rtype::config::ComponentCount, rtype::config::↔SystemCount > & entityManager ) [inline]
```

Construct a new BossHive Shoot System object.

## Parameters

<i>entityManager</i>	
----------------------	--

### 3.7.3 Member Function Documentation

#### 3.7.3.1 update() [1/2]

```
void rtype::system::BossHiveShootSystem::update (
    const float dt ) [inline], [virtual]
```

Update the BossHive Shoot system class, to allow enemies to shoot.

## Parameters

<i>dt</i>	The Delta Time for the shoot of enemies
-----------	---

Reimplemented from [engine::ecs::System< rtype::config::ComponentCount, rtype::config::SystemCount >](#).

#### 3.7.3.2 update() [2/2]

```
void rtype::system::BossHiveShootSystem::update (
    const float dt ) [inline], [virtual]
```

Update the BossHive Shoot system class, to allow enemies to shoot.

## Parameters

<i>dt</i>	The Delta Time for the shoot of enemies
-----------	---

Reimplemented from [engine::ecs::System< rtype::config::ComponentCount, rtype::config::SystemCount >](#).

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

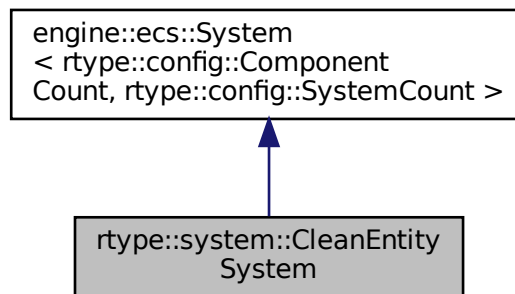
- rtype-client/include/systems/BossHiveShootSystem.hpp

## 3.8 rtype::system::CleanEntitySystem Class Reference

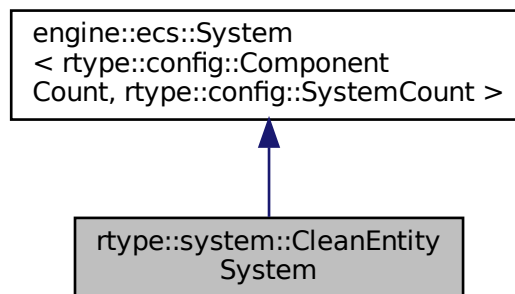
Clean entity system class to remove entities.

```
#include <CleanEntitySystem.hpp>
```

Inheritance diagram for `rtype::system::CleanEntitySystem`:



Collaboration diagram for `rtype::system::CleanEntitySystem`:



## Public Member Functions

- `CleanEntitySystem` (`engine::ecs::EntityManager` `< rtype::config::ComponentCount, rtype::config::SystemCount >` `&entityManager`)  
Construct a new Clean Entity System object.
- `void update` (`const float dt`)  
Update the Clean Entity System object.

## Additional Inherited Members

### 3.8.1 Detailed Description

Clean entity system class to remove entities.



## 3.8.2 Constructor & Destructor Documentation

### 3.8.2.1 CleanEntitySystem()

```
rtype::system::CleanEntitySystem::CleanEntitySystem (
    engine::ecs::EntityManager< rtype::config::ComponentCount, rtype::config::↵
SystemCount > & entityManager ) [inline]
```

Construct a new Clean Entity System object.

#### Parameters

<i>entityManager</i>	
----------------------	--

## 3.8.3 Member Function Documentation

### 3.8.3.1 update()

```
void rtype::system::CleanEntitySystem::update (
    const float dt ) [inline], [virtual]
```

Update the Clean Entity System object.

#### Parameters

<i>dt</i>	
-----------	--

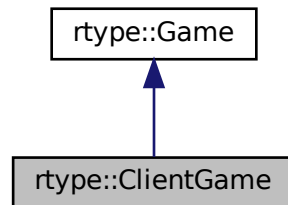
Reimplemented from [engine::ecs::System< rtype::config::ComponentCount, rtype::config::SystemCount >](#).

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

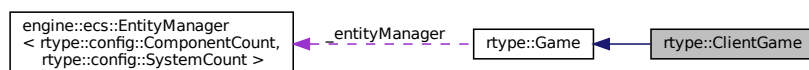
- game/include/systems/CleanEntitySystem.hpp

### 3.9 rtype::ClientGame Class Reference

Inheritance diagram for rtype::ClientGame:



Collaboration diagram for rtype::ClientGame:



#### Public Member Functions

- [ClientGame](#) ()  
Construct a new *ClientGame* object.
- [~ClientGame](#) ()
- float [getFps](#) () const  
Get the Fps of the game.
- void [handleEvent](#) ([NetworkEvent](#) package)  
Handle the event received from the server.
- void [gameEvent](#) () override  
*ClientGame* event.
- void [gameUpdate](#) () override  
*ClientGame* update.
- void [gameUi](#) ()  
*ClientGame* ui for the debug menu.
- void [gameDraw](#) ()  
*ClientGame* draw.
- void [gameLoop](#) () override  
*Game* loop to repeat at each frame.
- bool [isOver](#) () override  
Says whether the game is over or not.
- std::string [getIp](#) ()  
Gets the IP address of the server enter at the beginning.
- int [getPlayerID](#) ()  
Gets the id of the player.

## Additional Inherited Members

### 3.9.1 Constructor & Destructor Documentation

#### 3.9.1.1 ClientGame()

```
rtype::ClientGame::ClientGame ( )
```

Construct a new [ClientGame](#) object.

##### Parameters

<i>io_context</i>	
-------------------	--

#### 3.9.1.2 ~ClientGame()

```
rtype::ClientGame::~~ClientGame ( )
```

Destroy the game and add a dying event to the queue

### 3.9.2 Member Function Documentation

#### 3.9.2.1 getFps()

```
float rtype::ClientGame::getFps ( ) const [virtual]
```

Get the Fps of the game.

##### Returns

float

Implements [rtype::Game](#).

### 3.9.2.2 getIp()

```
std::string rtype::ClientGame::getIp ( )
```

Gets the IP address of the server enter at the beginning.

#### Returns

IP address of the running server

### 3.9.2.3 getPlayerID()

```
int rtype::ClientGame::getPlayerID ( )
```

Gets the id of the player.

#### Returns

An int containing the player ID

### 3.9.2.4 isOver()

```
bool rtype::ClientGame::isOver ( ) [override], [virtual]
```

Says whether the game is over or not.

#### Returns

The state of the game

Implements [rtype::Game](#).

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

- `rtype-client/include/ClientGame.hpp`

## 3.10 client::clientInstance Class Reference

Instance of the client containing a game.

```
#include <clientInstance.hpp>
```

## Public Member Functions

- [clientInstance](#) ()
- void [lifeCycle](#) ()

*Handle the life cycle of the client.*

### 3.10.1 Detailed Description

Instance of the client containing a game.

### 3.10.2 Constructor & Destructor Documentation

#### 3.10.2.1 clientInstance()

```
client::clientInstance::clientInstance ( )
```

Create a window which is waiting for the user to enter the server ip adress

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

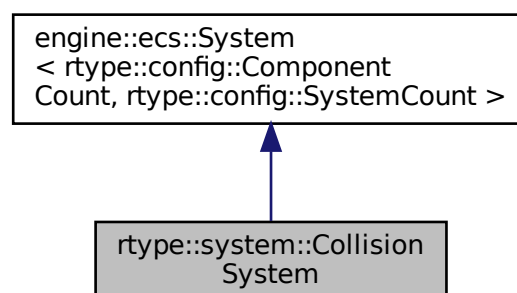
- rtype-client/include/clientInstance.hpp

## 3.11 rtype::system::CollisionSystem Class Reference

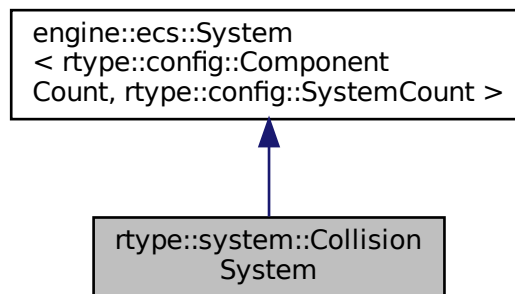
Collision system class to check collision between entities.

```
#include <CollisionSystem.hpp>
```

Inheritance diagram for rtype::system::CollisionSystem:



Collaboration diagram for `rtype::system::CollisionSystem`:



## Public Member Functions

- `CollisionSystem` (`engine::ecs::EntityManager` `< rtype::config::ComponentCount, rtype::config::SystemCount >` `&entityManager`)  
Construct a new *Collision System* object.
- void `update` (`const float dt`)  
Update the *Collision System* object.

## Additional Inherited Members

### 3.11.1 Detailed Description

Collision system class to check collision between entities.

### 3.11.2 Constructor & Destructor Documentation

#### 3.11.2.1 CollisionSystem()

```

rtype::system::CollisionSystem::CollisionSystem (
    engine::ecs::EntityManager< rtype::config::ComponentCount, rtype::config::SystemCount > & entityManager ) [inline]
  
```

Construct a new *Collision System* object.

#### Parameters

<code>entityManager</code>	
----------------------------	--

### 3.11.3 Member Function Documentation

#### 3.11.3.1 update()

```
void rtype::system::CollisionSystem::update (
    const float dt ) [inline], [virtual]
```

Update the Collision System object.

##### Parameters

<i>dt</i>	
-----------	--

Reimplemented from [engine::ecs::System< rtype::config::ComponentCount, rtype::config::SystemCount >](#).

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

- game/include/systems/CollisionSystem.hpp

## 3.12 engine::ecs::Component< T, Type > Class Template Reference

[Component](#) class which is the base class for all components.

```
#include <Component.hpp>
```

### Static Public Attributes

- static constexpr auto **type** = static\_cast<std::size\_t>(Type)

#### 3.12.1 Detailed Description

```
template<typename T, auto Type>
class engine::ecs::Component< T, Type >
```

[Component](#) class which is the base class for all components.

##### Template Parameters

<i>T</i>	
<i>Type</i>	

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

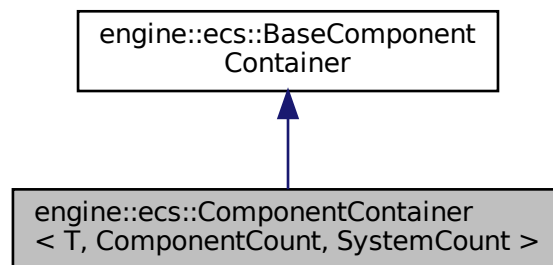
- engine/include/ecs/Component.hpp

### 3.13 engine::ecs::ComponentContainer< T, ComponentCount, SystemCount > Class Template Reference

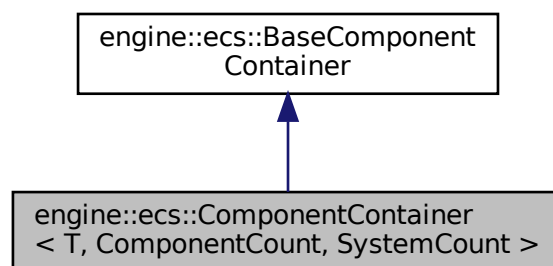
[ComponentContainer](#) class used to store components.

```
#include <ComponentContainer.hpp>
```

Inheritance diagram for engine::ecs::ComponentContainer< T, ComponentCount, SystemCount >:



Collaboration diagram for engine::ecs::ComponentContainer< T, ComponentCount, SystemCount >:



#### Public Member Functions

- **ComponentContainer** (std::vector< Index > &entityToComponent, std::vector< std::bitset< ComponentCount >> &entityToBitset)
- virtual void [reserve](#) (std::size\_t size) override  
*Reserve the size of the container.*
- T & [get](#) (Entity entity)  
*Get a component of an entity.*
- const T & [get](#) (Entity entity) const



*Get a component of an entity with const.*

- template<typename... Args>  
void [add](#) (Entity entity, Args &&...args)

*Add a component to an entity.*

- void [remove](#) (Entity entity)

*Remove a component from an entity.*

- virtual bool [tryRemove](#) (Entity entity) override

*Try to remove a component from an entity.*

- Entity [getOwner](#) (const T &component) const

*Get the Owner object of a component.*

### 3.13.1 Detailed Description

```
template<typename T, std::size_t ComponentCount, std::size_t SystemCount>
class engine::ecs::ComponentContainer< T, ComponentCount, SystemCount >
```

[ComponentContainer](#) class used to store components.

Template Parameters

<i>T</i>	
<i>ComponentCount</i>	
<i>SystemCount</i>	

### 3.13.2 Member Function Documentation

#### 3.13.2.1 add()

```
template<typename T , std::size_t ComponentCount, std::size_t SystemCount>
template<typename... Args>
void engine::ecs::ComponentContainer< T, ComponentCount, SystemCount >::add (
    Entity entity,
    Args &&... args ) [inline]
```

Add a component to an entity.

Template Parameters

<i>Args</i>	
-------------	--

Parameters

<i>entity</i>	
<i>args</i>	

### 3.13.2.2 `get()` [1/2]

```
template<typename T , std::size_t ComponentCount, std::size_t SystemCount>
T& engine::ecs::ComponentContainer< T, ComponentCount, SystemCount >::get (
    Entity entity ) [inline]
```

Get a component of an entity.

#### Parameters

<i>entity</i>	
---------------	--

#### Returns

T&

### 3.13.2.3 `get()` [2/2]

```
template<typename T , std::size_t ComponentCount, std::size_t SystemCount>
const T& engine::ecs::ComponentContainer< T, ComponentCount, SystemCount >::get (
    Entity entity ) const [inline]
```

Get a component of an entity with const.

#### Parameters

<i>entity</i>	
---------------	--

#### Returns

const T&

### 3.13.2.4 `getOwner()`

```
template<typename T , std::size_t ComponentCount, std::size_t SystemCount>
Entity engine::ecs::ComponentContainer< T, ComponentCount, SystemCount >::getOwner (
    const T & component ) const [inline]
```

Get the Owner object of a component.

## Parameters

<i>component</i>	
------------------	--

## Returns

Entity

## 3.13.2.5 remove()

```
template<typename T , std::size_t ComponentCount, std::size_t SystemCount>
void engine::ecs::ComponentContainer< T, ComponentCount, SystemCount >::remove (
    Entity entity ) [inline]
```

Remove a component from an entity.

## Parameters

<i>entity</i>	
---------------	--

## 3.13.2.6 reserve()

```
template<typename T , std::size_t ComponentCount, std::size_t SystemCount>
virtual void engine::ecs::ComponentContainer< T, ComponentCount, SystemCount >::reserve (
    std::size_t size ) [inline], [override], [virtual]
```

Reserve the size of the container.

## Parameters

<i>size</i>	
-------------	--

Implements [engine::ecs::BaseComponentContainer](#).

## 3.13.2.7 tryRemove()

```
template<typename T , std::size_t ComponentCount, std::size_t SystemCount>
virtual bool engine::ecs::ComponentContainer< T, ComponentCount, SystemCount >::tryRemove (
    Entity entity ) [inline], [override], [virtual]
```

Try to remove a component from an entity.

## Parameters

<i>entity</i>	
---------------	--

## Returns

true if the component was removed.  
false if the component was not removed.

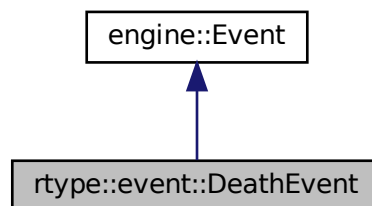
Implements [engine::ecs::BaseComponentContainer](#).

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

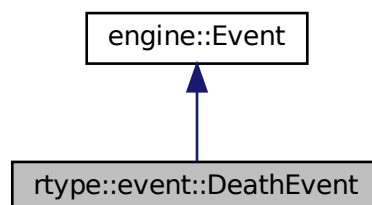
- engine/include/ecs/ComponentContainer.hpp

### 3.14 rtype::event::DeathEvent Class Reference

Inheritance diagram for rtype::event::DeathEvent:



Collaboration diagram for rtype::event::DeathEvent:



## Public Attributes

- int **entity**
- bool **isDead**

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

- game/include/events/DeathEvent.hpp

## 3.15 rtype::debug::Debugger Class Reference

### Public Member Functions

- bool **getShowHitbox** () const
- bool **hasComponents** (rtype::Game &game, int id)
- void **calculateNumberOfActiveEntities** (rtype::Game &game)
- void **update** (rtype::Game &game)
- void **draw** (rtype::Game &game)
- void **drawEntityWindow** (rtype::Game &game)

### Static Public Member Functions

- static Debugger & **getInstance** ()

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

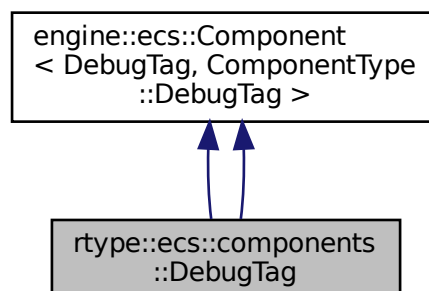
- rtype-client/include/debug/Debugger.hpp

## 3.16 rtype::ecs::components::DebugTag Struct Reference

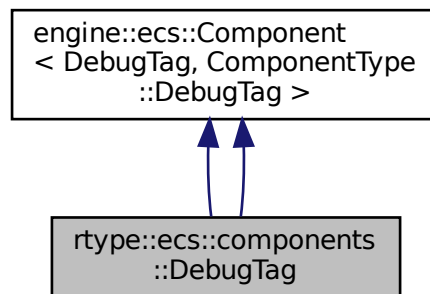
[DebugTag](#) struct which contains the debug tag of the entity.

```
#include <Components.hpp>
```

Inheritance diagram for rtype::ecs::components::DebugTag:



Collaboration diagram for `rtype::ecs::components::DebugTag`:



## Public Member Functions

- **DebugTag** (const std::string &Tag)
- **DebugTag** (const std::string &Tag)

## Public Attributes

- std::string **tag**

## Additional Inherited Members

### 3.16.1 Detailed Description

[DebugTag](#) struct which contains the debug tag of the entity.

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

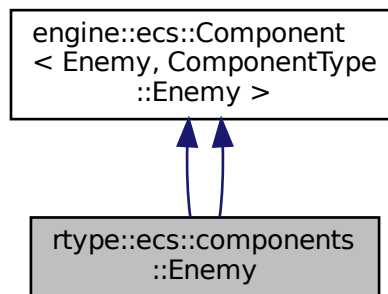
- `rtype-client/include/Components.hpp`

## 3.17 rtype::ecs::components::Enemy Struct Reference

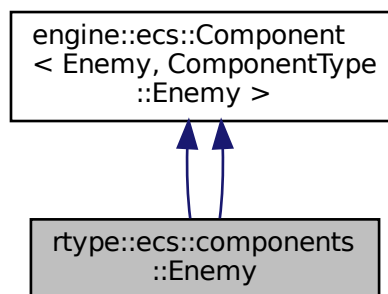
[Enemy](#) struct which contains the enemy of the entity.

```
#include <Components.hpp>
```

Inheritance diagram for rtype::ecs::components::Enemy:



Collaboration diagram for rtype::ecs::components::Enemy:



## Additional Inherited Members

### 3.17.1 Detailed Description

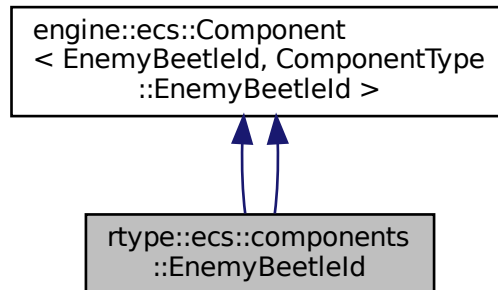
[Enemy](#) struct which contains the enemy of the entity.

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

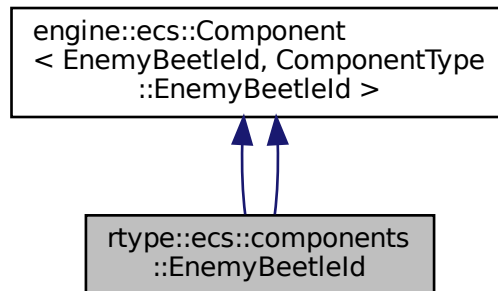
- `rtype-client/include/Components.hpp`

### 3.18 rtype::ecs::components::EnemyBeetleId Struct Reference

Inheritance diagram for rtype::ecs::components::EnemyBeetleId:



Collaboration diagram for rtype::ecs::components::EnemyBeetleId:



#### Public Member Functions

- **EnemyBeetleId** (int enemyBeetle)
- **EnemyBeetleId** (int enemyBeetle)

#### Public Attributes

- int **id**



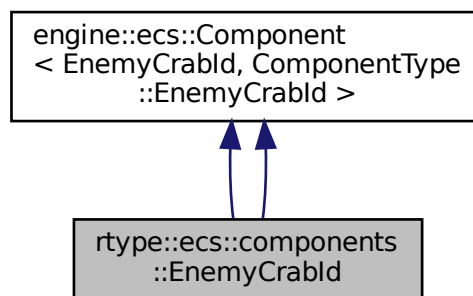
## Additional Inherited Members

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

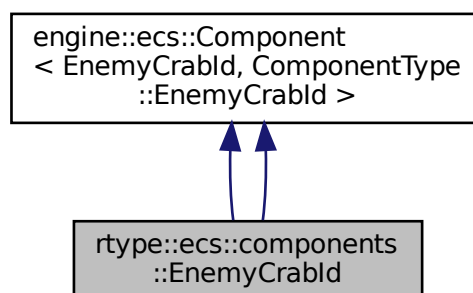
- rtype-client/include/Components.hpp

## 3.19 rtype::ecs::components::EnemyCrabId Struct Reference

Inheritance diagram for rtype::ecs::components::EnemyCrabId:



Collaboration diagram for rtype::ecs::components::EnemyCrabId:



## Public Member Functions

- **EnemyCrabId** (int enemyCrab)
- **EnemyCrabId** (int enemyCrab)

## Public Attributes

- `int id`

## Additional Inherited Members

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

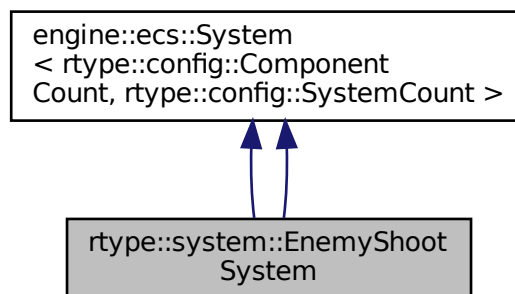
- `rtype-client/include/Components.hpp`

## 3.20 `rtype::system::EnemyShootSystem` Class Reference

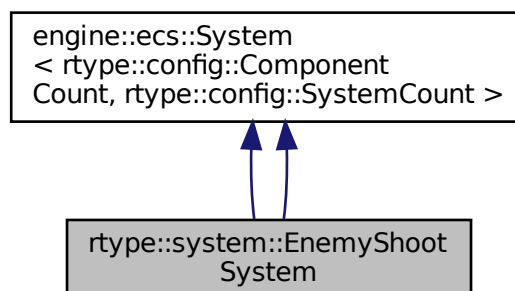
EnemyShoot system class used to let the enemy shoot.

```
#include <EnemyShootSystem.hpp>
```

Inheritance diagram for `rtype::system::EnemyShootSystem`:



Collaboration diagram for `rtype::system::EnemyShootSystem`:



## Public Member Functions

- [EnemyShootSystem](#) ([engine::ecs::EntityManager](#)< rtype::config::ComponentCount, rtype::config::System↔Count > &entityManager)  
Construct a new *Enemy Shoot System* object.
- void [update](#) (const float dt)  
Update the *Enemy Shoot* system class, to allow enemies to shoot.
- [EnemyShootSystem](#) ([engine::ecs::EntityManager](#)< rtype::config::ComponentCount, rtype::config::System↔Count > &entityManager)  
Construct a new *Enemy Shoot System* object.
- void [update](#) (const float dt)  
Update the *Enemy Shoot* system class, to allow enemies to shoot.

## Additional Inherited Members

### 3.20.1 Detailed Description

EnemyShoot system class used to let the enemy shoot.

### 3.20.2 Constructor & Destructor Documentation

#### 3.20.2.1 EnemyShootSystem() [1/2]

```
rtype::system::EnemyShootSystem::EnemyShootSystem (
    engine::ecs::EntityManager< rtype::config::ComponentCount, rtype::config::↔
SystemCount > & entityManager ) [inline]
```

Construct a new *Enemy Shoot System* object.

Parameters

<i>entityManager</i>	
----------------------	--

#### 3.20.2.2 EnemyShootSystem() [2/2]

```
rtype::system::EnemyShootSystem::EnemyShootSystem (
    engine::ecs::EntityManager< rtype::config::ComponentCount, rtype::config::↔
SystemCount > & entityManager ) [inline]
```

Construct a new *Enemy Shoot System* object.

## Parameters

<i>entityManager</i>	
----------------------	--

### 3.20.3 Member Function Documentation

#### 3.20.3.1 update() [1/2]

```
void rtype::system::EnemyShootSystem::update (
    const float dt ) [inline], [virtual]
```

Update the Enemy Shoot system class, to allow enemies to shoot.

## Parameters

<i>dt</i>	The Delta Time for the shoot of enemies
-----------	---

Reimplemented from [engine::ecs::System< rtype::config::ComponentCount, rtype::config::SystemCount >](#).

#### 3.20.3.2 update() [2/2]

```
void rtype::system::EnemyShootSystem::update (
    const float dt ) [inline], [virtual]
```

Update the Enemy Shoot system class, to allow enemies to shoot.

## Parameters

<i>dt</i>	The Delta Time for the shoot of enemies
-----------	---

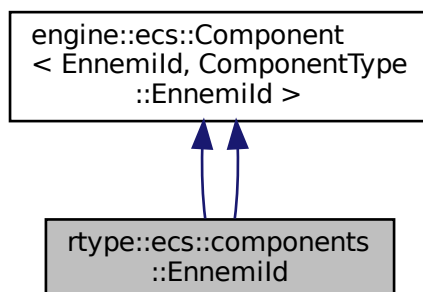
Reimplemented from [engine::ecs::System< rtype::config::ComponentCount, rtype::config::SystemCount >](#).

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

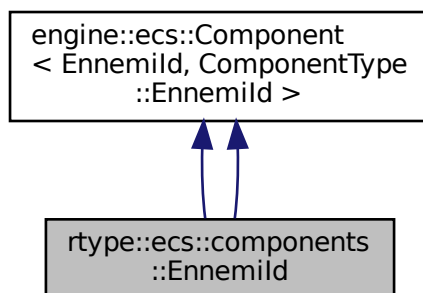
- [rtype-client/include/systems/EnemyShootSystem.hpp](#)

## 3.21 rtype::ecs::components::Ennemild Struct Reference

Inheritance diagram for rtype::ecs::components::Ennemild:



Collaboration diagram for rtype::ecs::components::Ennemild:



### Public Member Functions

- **Ennemild** (int ennemild)
- **Ennemild** (int ennemild)

### Public Attributes

- int **id**

## Additional Inherited Members

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

- `rtype-client/include/Components.hpp`

## 3.22 `engine::ecs::EntityContainer< ComponentCount, SystemCount >` Class Template Reference

`EntityContainer` class used to store entities.

```
#include <EntityContainer.hpp>
```

### Public Member Functions

- void `reserve` (std::size\_t size)  
*Construct a new `EntityContainer` object.*
- std::vector< std::bitset< ComponentCount > > & `getEntityToBitset` ()  
*Get the Entity To Bitset object.*
- const std::bitset< ComponentCount > & `getBitset` (Entity entity) const  
*Get the Entity To Bitset object.*
- std::vector< Index > & `getEntityToComponent` (std::size\_t type)  
*Get the Entity To `Component` object.*
- std::vector< Index > & `getEntityToManagedEntity` (std::size\_t type)  
*Get the Entity To Managed Entity object.*
- Entity `create` ()  
*Create a new Entity object.*
- void `remove` (Entity entity)  
*Remove an Entity object.*

### 3.22.1 Detailed Description

```
template<std::size_t ComponentCount, std::size_t SystemCount>
class engine::ecs::EntityContainer< ComponentCount, SystemCount >
```

`EntityContainer` class used to store entities.

#### Template Parameters

<i>ComponentCount</i>	
<i>SystemCount</i>	

### 3.22.2 Member Function Documentation

### 3.22.2.1 create()

```
template<std::size_t ComponentCount, std::size_t SystemCount>
Entity engine::ecs::EntityContainer< ComponentCount, SystemCount >::create ( ) [inline]
```

Create a new Entity object.

#### Returns

Entity

### 3.22.2.2 getBitset()

```
template<std::size_t ComponentCount, std::size_t SystemCount>
const std::bitset<ComponentCount>& engine::ecs::EntityContainer< ComponentCount, SystemCount
>::getBitset (
    Entity entity ) const [inline]
```

Get the Entity To Bitset object.

#### Parameters

<i>entity</i>	
---------------	--

#### Returns

const std::bitset<ComponentCount>&

### 3.22.2.3 getEntityToBitset()

```
template<std::size_t ComponentCount, std::size_t SystemCount>
std::vector<std::bitset<ComponentCount> >& engine::ecs::EntityContainer< ComponentCount,
SystemCount >::getEntityToBitset ( ) [inline]
```

Get the Entity To Bitset object.

#### Returns

std::vector<std::bitset<ComponentCount>>&

### 3.22.2.4 getEntityToComponent()

```
template<std::size_t ComponentCount, std::size_t SystemCount>
std::vector<Index>& engine::ecs::EntityContainer< ComponentCount, SystemCount >::getEntity↔
ToComponent (
    std::size_t type ) [inline]
```

Get the Entity To Component object.

**Parameters**

<i>type</i>	
-------------	--

**Returns**

`std::vector<Index>&`

**3.22.2.5 getEntityToManagedEntity()**

```
template<std::size_t ComponentCount, std::size_t SystemCount>
std::vector<Index>& engine::ecs::EntityContainer< ComponentCount, SystemCount >::getEntityToManagedEntity (
    std::size_t type ) [inline]
```

Get the Entity To Managed Entity object.

**Parameters**

<i>type</i>	
-------------	--

**Returns**

`std::vector<Index>&`

**3.22.2.6 remove()**

```
template<std::size_t ComponentCount, std::size_t SystemCount>
void engine::ecs::EntityContainer< ComponentCount, SystemCount >::remove (
    Entity entity ) [inline]
```

Remove an Entity object.

**Parameters**

<i>entity</i>	
---------------	--

**3.22.2.7 reserve()**

```
template<std::size_t ComponentCount, std::size_t SystemCount>
void engine::ecs::EntityContainer< ComponentCount, SystemCount >::reserve (
    std::size_t size ) [inline]
```



Construct a new [EntityContainer](#) object.

#### Parameters

<i>size</i>	
-------------	--

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

- engine/include/ecs/EntityContainer.hpp

## 3.23 engine::ecs::EntityManager< ComponentCount, SystemCount > Class Template Reference

Entity manager class. It is used to create entities, add components to them and create systems with the entity linked.

```
#include <EntityManager.hpp>
```

### Public Member Functions

- template<typename T >  
void [registerComponent](#) ()  
*Construct a new Entity Manager object.*
- template<typename T , typename... Args>  
T \* [createSystem](#) (Args &&...args)  
*Create a System object.*
- void [reserve](#) (std::size\_t size)  
*Reserve the size of the containers.*
- Entity [createEntity](#) ()  
*Create a Entity object.*
- void [removeEntity](#) (Entity entity)  
*Remove a Entity object.*
- template<typename T >  
bool [hasComponent](#) (Entity entity) const  
*Add a Component to a Entity.*
- template<typename... Ts>  
bool [hasComponents](#) (Entity entity) const  
*Check if a Entity has a set of Components.*
- template<typename T >  
T & [getComponent](#) (Entity entity)  
*Get the Component object.*
- template<typename T >  
const T & [getComponent](#) (Entity entity) const  
*Get the Component object with const.*
- template<typename... Ts>  
std::tuple< Ts &... > [getComponents](#) (Entity entity)  
*Get the Components object.*
- template<typename... Ts>  
std::tuple< const Ts &... > [getComponents](#) (Entity entity) const

- Get the Components object.*

  - `template<typename T , typename... Args>`  
`void addComponent (Entity entity, Args &&...args)`

*Add a [Component](#) to a Entity.*
- `template<typename T >`  
`void removeComponent (Entity entity)`

*Remove a [Component](#) from a Entity.*
- `template<typename T >`  
`Entity getOwner (const T &component) const`

*Get the Owner object.*
- `void updateSystems (const float dt)`

*Get the [Component](#) Container object.*
- `void drawSystems (engine::graphical::GraphicalWindow &window)`

*Draw the Systems.*
- `void dispatchEvent (const Event &event)`

*Dispatch an [Event](#) to all Systems.*

### 3.23.1 Detailed Description

```
template<std::size_t ComponentCount, std::size_t SystemCount>
class engine::ecs::EntityManager< ComponentCount, SystemCount >
```

Entity manager class. It is used to create entities, add components to them and create systems with the entity linked.

Template Parameters

<i>ComponentCount</i>	
<i>SystemCount</i>	

### 3.23.2 Member Function Documentation

#### 3.23.2.1 `addComponent()`

```
template<std::size_t ComponentCount, std::size_t SystemCount>
template<typename T , typename... Args>
void engine::ecs::EntityManager< ComponentCount, SystemCount >::addComponent (
    Entity entity,
    Args &&... args ) [inline]
```

Add a [Component](#) to a Entity.

Template Parameters

<i>T</i>	
<i>Args</i>	

## Parameters

<i>entity</i>	
<i>args</i>	

**3.23.2.2 createEntity()**

```
template<std::size_t ComponentCount, std::size_t SystemCount>
Entity engine::ecs::EntityManager< ComponentCount, SystemCount >::createEntity ( ) [inline]
```

Create a Entity object.

## Returns

Entity

**3.23.2.3 createSystem()**

```
template<std::size_t ComponentCount, std::size_t SystemCount>
template<typename T , typename... Args>
T* engine::ecs::EntityManager< ComponentCount, SystemCount >::createSystem (
    Args &&... args ) [inline]
```

Create a [System](#) object.

## Template Parameters

<i>T</i>	
<i>Args</i>	

## Parameters

<i>args</i>	
-------------	--

## Returns

T\*

**3.23.2.4 dispatchEvent()**

```
template<std::size_t ComponentCount, std::size_t SystemCount>
void engine::ecs::EntityManager< ComponentCount, SystemCount >::dispatchEvent (
    const Event & event ) [inline]
```

Dispatch an [Event](#) to all Systems.

#### Parameters

<i>event</i>	
--------------	--

### 3.23.2.5 drawSystems()

```
template<std::size_t ComponentCount, std::size_t SystemCount>
void engine::ecs::EntityManager< ComponentCount, SystemCount >::drawSystems (
    engine::graphical::GraphicalWindow & window ) [inline]
```

Draw the Systems.

#### Parameters

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

### 3.23.2.6 getComponent() [1/2]

```
template<std::size_t ComponentCount, std::size_t SystemCount>
template<typename T >
T& engine::ecs::EntityManager< ComponentCount, SystemCount >::getComponent (
    Entity entity ) [inline]
```

Get the [Component](#) object.

#### Template Parameters

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

#### Parameters

<i>entity</i>	
---------------	--

#### Returns

T&

### 3.23.2.7 getComponent() [2/2]

```
template<std::size_t ComponentCount, std::size_t SystemCount>
template<typename T >
```

```
const T& engine::ecs::EntityManager< ComponentCount, SystemCount >::getComponent (
    Entity entity ) const [inline]
```

Get the [Component](#) object with const.

#### Template Parameters

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

#### Parameters

<i>entity</i>	
---------------	--

#### Returns

const T&

### 3.23.2.8 getComponents() [1/2]

```
template<std::size_t ComponentCount, std::size_t SystemCount>
template<typename... Ts>
std::tuple<Ts &...> engine::ecs::EntityManager< ComponentCount, SystemCount >::getComponents
(
    Entity entity ) [inline]
```

Get the Components object.

#### Template Parameters

<i>Ts</i>	
-----------	--

#### Parameters

<i>entity</i>	
---------------	--

#### Returns

std::tuple<Ts &...>

### 3.23.2.9 getComponents() [2/2]

```
template<std::size_t ComponentCount, std::size_t SystemCount>
template<typename... Ts>
```

```
std::tuple<const Ts &...> engine::ecs::EntityManager< ComponentCount, SystemCount >::get↵  
Components (↵  
    Entity entity ) const    [inline]
```

Get the Components object.

## Template Parameters

<i>Ts</i>	
-----------	--

## Parameters

<i>entity</i>	
---------------	--

## Returns

std::tuple<const Ts &...>

## 3.23.2.10 getOwner()

```
template<std::size_t ComponentCount, std::size_t SystemCount>
template<typename T >
Entity engine::ecs::EntityManager< ComponentCount, SystemCount >::getOwner (
    const T & component ) const [inline]
```

Get the Owner object.

## Template Parameters

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

## Parameters

<i>component</i>	
------------------	--

## Returns

Entity

## 3.23.2.11 hasComponent()

```
template<std::size_t ComponentCount, std::size_t SystemCount>
template<typename T >
bool engine::ecs::EntityManager< ComponentCount, SystemCount >::hasComponent (
    Entity entity ) const [inline]
```

Add a [Component](#) to a Entity.

**Template Parameters**

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

**Parameters**

<i>entity</i>	
---------------	--

**Returns**

true

false

**3.23.2.12 hasComponents()**

```
template<std::size_t ComponentCount, std::size_t SystemCount>
template<typename... Ts>
bool engine::ecs::EntityManager< ComponentCount, SystemCount >::hasComponents (
    Entity entity ) const [inline]
```

Check if a Entity has a set of Components.

**Template Parameters**

<i>Ts</i>	
-----------	--

**Parameters**

<i>entity</i>	
---------------	--

**Returns**

true

false

**3.23.2.13 registerComponent()**

```
template<std::size_t ComponentCount, std::size_t SystemCount>
template<typename T >
void engine::ecs::EntityManager< ComponentCount, SystemCount >::registerComponent ( ) [inline]
```

Construct a new Entity Manager object.



## Template Parameters

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

**3.23.2.14 removeComponent()**

```
template<std::size_t ComponentCount, std::size_t SystemCount>
template<typename T >
void engine::ecs::EntityManager< ComponentCount, SystemCount >::removeComponent (
    Entity entity ) [inline]
```

Remove a [Component](#) from a Entity.

## Template Parameters

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

## Parameters

<i>entity</i>	
---------------	--

**3.23.2.15 removeEntity()**

```
template<std::size_t ComponentCount, std::size_t SystemCount>
void engine::ecs::EntityManager< ComponentCount, SystemCount >::removeEntity (
    Entity entity ) [inline]
```

Remove a Entity object.

## Parameters

<i>entity</i>	
---------------	--

**3.23.2.16 reserve()**

```
template<std::size_t ComponentCount, std::size_t SystemCount>
void engine::ecs::EntityManager< ComponentCount, SystemCount >::reserve (
    std::size_t size ) [inline]
```

Reserve the size of the containers.

## Parameters

<i>size</i>	
-------------	--

**3.23.2.17 updateSystems()**

```
template<std::size_t ComponentCount, std::size_t SystemCount>
void engine::ecs::EntityManager< ComponentCount, SystemCount >::updateSystems (
    const float dt ) [inline]
```

Get the [Component](#) Container object.

## Parameters

<i>dt</i>	
-----------	--

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

- engine/include/ecs/EntityManager.hpp

**3.24 rtype::EntityTemplate Class Reference**

[EntityTemplate](#) class which allows you to create entities.

```
#include <EntityTemplate.hpp>
```

**Public Member Functions**

- [EntityTemplate](#) ([engine::ecs::EntityManager](#)< [rtype::config::ComponentCount](#), [rtype::config::SystemCount](#) > &entityManager)  
*Construct a new Entity Template object.*
- int [createPlayer](#) (int playerId, bool isOwner, std::pair< float, float > pos)  
*Create a Player object.*
- int [createPlayerBullet](#) (int x, int y)  
*Create a Player Bullet object.*
- int [createSpecialPlayerBullet](#) (int x, int y)  
*Create a Special Player Bullet object when a bonus is triggered.*
- int [createEnemyBullet](#) (int x, int y)  
*Create a Enemy Bullet object.*
- int [createBossHiveBullet](#) (int x, int y, float velX, float velY)  
*Create a Boss Hive Bullet object.*
- int [createBossAlienBullet](#) (int x, int y, float velX, float velY)  
*Create a Boss Alien Bullet object.*
- int [createBossAlien](#) (float x, float y, int enemyId)  
*Create a Boss Alien object.*

- int [createBossHive](#) (float x, float y, int enemyId)  
*Create a Boss Hive object.*
- int [createEnemyCrab](#) (float x, float y, int enemyId)  
*Create a Enemy Crab object.*
- int [createEnemyBeetle](#) (float x, float y, int enemyId)  
*Create a Enemy Beetle object.*
- int [createSpecialShootBonus](#) (float x, float y, int id)  
*Create a Bonus object.*
- int [createHealthBonus](#) (float x, float y, int id)  
*Create a Health Bonus object.*
- int [createParallax](#) (float x, float y, const std::string &textureName, float speed, int spriteLayer)  
*Create a Parallax object.*
- int [createTopBoundary](#) (float x, float y)  
*Create a top boundary object.*
- int [createBottomBoundary](#) (float x, float y)  
*Create a bottom boundary object.*
- int [createAsteroid](#) (float x, float y, int id)  
*Create a asteroid object.*
- [EntityTemplate](#) ([engine::ecs::EntityManager](#)< rtype::config::ComponentCount, rtype::config::SystemCount > &entityManager)  
*Construct a new Entity Template object.*
- int [createPlayer](#) (int playerId, bool isOwner, std::pair< float, float > pos)  
*Create a Player object.*
- int [createPlayerBullet](#) (int x, int y)  
*Create a Player Bullet object.*
- int [createSpecialPlayerBullet](#) (int x, int y)  
*Create a Special Player Bullet object when a bonus is triggered.*
- int [createEnemyBullet](#) (int x, int y)  
*Create a Enemy Bullet object.*
- int [createBossHiveBullet](#) (int x, int y, float velX, float velY)  
*Create a Boss Hive Bullet object.*
- int [createBossAlienBullet](#) (int x, int y, float velX, float velY)  
*Create a Boss Alien Bullet object.*
- int [createBossAlien](#) (float x, float y, int enemyId)  
*Create a Boss Alien object.*
- int [createBossHive](#) (float x, float y, int enemyId)  
*Create a Boss Hive object.*
- int [createEnemyCrab](#) (float x, float y, int enemyId)  
*Create a Enemy Crab object.*
- int [createEnemyBeetle](#) (float x, float y, int enemyId)  
*Create a Enemy Beetle object.*
- int [createSpecialShootBonus](#) (float x, float y, int id)  
*Create a Bonus object.*
- int [createHealthBonus](#) (float x, float y, int id)  
*Create a Health Bonus object.*
- int [createParallax](#) (float x, float y, const std::string &textureName, float speed, int spriteLayer)  
*Create a Parallax object.*
- int [createTopBoundary](#) (float x, float y)  
*Create a top boundary object.*
- int [createBottomBoundary](#) (float x, float y)  
*Create a bottom boundary object.*
- int [createAsteroid](#) (float x, float y, int id)  
*Create a asteroid object.*

### 3.24.1 Detailed Description

[EntityTemplate](#) class which allows you to create entities.

### 3.24.2 Constructor & Destructor Documentation

#### 3.24.2.1 EntityTemplate() [1/2]

```
rtype::EntityTemplate::EntityTemplate (
    engine::ecs::EntityManager< rtype::config::ComponentCount, rtype::config::↵
SystemCount > & entityManager )
```

Construct a new Entity Template object.

##### Parameters

<i>entityManager</i>	
<i>assetManager</i>	

#### 3.24.2.2 EntityTemplate() [2/2]

```
rtype::EntityTemplate::EntityTemplate (
    engine::ecs::EntityManager< rtype::config::ComponentCount, rtype::config::↵
SystemCount > & entityManager )
```

Construct a new Entity Template object.

##### Parameters

<i>entityManager</i>	
<i>assetManager</i>	

### 3.24.3 Member Function Documentation

#### 3.24.3.1 createAsteroid() [1/2]

```
int rtype::EntityTemplate::createAsteroid (
    float x,
```

```
float y,  
int id )
```

Create a asteroid object.

**Parameters**

<i>x</i>	position
<i>y</i>	position

**Returns**

int

**3.24.3.2 createAsteroid()** [2/2]

```
int rtype::EntityTemplate::createAsteroid (
    float x,
    float y,
    int id )
```

Create a asteroid object.

**Parameters**

<i>x</i>	position
<i>y</i>	position

**Returns**

int

**3.24.3.3 createBossAlien()** [1/2]

```
int rtype::EntityTemplate::createBossAlien (
    float x,
    float y,
    int enemyId )
```

Create a Boss Alien object.

**Parameters**

<i>x</i>	position
<i>y</i>	position

**Returns**

int

### 3.24.3.4 createBossAlien() [2/2]

```
int rtype::EntityTemplate::createBossAlien (
    float x,
    float y,
    int enemyId )
```

Create a Boss Alien object.

#### Parameters

<i>x</i>	position
<i>y</i>	position

#### Returns

int

### 3.24.3.5 createBossAlienBullet() [1/2]

```
int rtype::EntityTemplate::createBossAlienBullet (
    int x,
    int y,
    float velX,
    float velY )
```

Create a Boss Alien Bullet object.

#### Parameters

<i>x</i>	position
<i>y</i>	position
<i>velX</i>	velocity in X
<i>velY</i>	velocity in Y

#### Returns

int

### 3.24.3.6 createBossAlienBullet() [2/2]

```
int rtype::EntityTemplate::createBossAlienBullet (
    int x,
```

```
int y,  
float velX,  
float velY )
```

Create a Boss Alien Bullet object.

#### Parameters

<i>x</i>	position
<i>y</i>	position
<i>velX</i>	velocity in X
<i>velY</i>	velocity in Y

#### Returns

int

#### 3.24.3.7 createBossHive() [1/2]

```
int rtype::EntityTemplate::createBossHive (  
    float x,  
    float y,  
    int enemyId )
```

Create a Boss Hive object.

#### Parameters

<i>x</i>	position
<i>y</i>	position

#### Returns

int

#### 3.24.3.8 createBossHive() [2/2]

```
int rtype::EntityTemplate::createBossHive (  
    float x,  
    float y,  
    int enemyId )
```

Create a Boss Hive object.



## Parameters

<i>x</i>	position
<i>y</i>	position

## Returns

int

**3.24.3.9 createBossHiveBullet()** [1/2]

```
int rtype::EntityTemplate::createBossHiveBullet (
    int x,
    int y,
    float velX,
    float velY )
```

Create a Boss Hive Bullet object.

## Parameters

<i>x</i>	position
<i>y</i>	position

## Returns

int

**3.24.3.10 createBossHiveBullet()** [2/2]

```
int rtype::EntityTemplate::createBossHiveBullet (
    int x,
    int y,
    float velX,
    float velY )
```

Create a Boss Hive Bullet object.

## Parameters

<i>x</i>	position
<i>y</i>	position

**Returns**

int

**3.24.3.11 createBottomBoundary() [1/2]**

```
int rtype::EntityTemplate::createBottomBoundary (
    float x,
    float y )
```

Create a bottom boundary object.

**Parameters**

<i>x</i>	position
<i>y</i>	position

**Returns**

int

**3.24.3.12 createBottomBoundary() [2/2]**

```
int rtype::EntityTemplate::createBottomBoundary (
    float x,
    float y )
```

Create a bottom boundary object.

**Parameters**

<i>x</i>	position
<i>y</i>	position

**Returns**

int

**3.24.3.13 createEnemyBeetle() [1/2]**

```
int rtype::EntityTemplate::createEnemyBeetle (
    float x,
```

```
float y,  
int enemyId )
```

Create a Enemy Beetle object.

**Parameters**

<i>x</i>	position
<i>y</i>	position

**Returns**

int

**3.24.3.14 createEnemyBeetle() [2/2]**

```
int rtype::EntityTemplate::createEnemyBeetle (
    float x,
    float y,
    int enemyId )
```

Create a Enemy Beetle object.

**Parameters**

<i>x</i>	position
<i>y</i>	position

**Returns**

int

**3.24.3.15 createEnemyBullet() [1/2]**

```
int rtype::EntityTemplate::createEnemyBullet (
    int x,
    int y )
```

Create a Enemy Bullet object.

**Parameters**

<i>x</i>	position
<i>y</i>	position

**Returns**

int

### 3.24.3.16 createEnemyBullet() [2/2]

```
int rtype::EntityTemplate::createEnemyBullet (
    int x,
    int y )
```

Create a Enemy Bullet object.

#### Parameters

<i>x</i>	position
<i>y</i>	position

#### Returns

int

### 3.24.3.17 createEnemyCrab() [1/2]

```
int rtype::EntityTemplate::createEnemyCrab (
    float x,
    float y,
    int enemyId )
```

Create a Enemy Crab object.

#### Parameters

<i>x</i>	position
<i>y</i>	position

#### Returns

int

### 3.24.3.18 createEnemyCrab() [2/2]

```
int rtype::EntityTemplate::createEnemyCrab (
    float x,
    float y,
    int enemyId )
```

Create a Enemy Crab object.

**Parameters**

<i>x</i>	position
<i>y</i>	position

**Returns**

int

**3.24.3.19 createHealthBonus() [1/2]**

```
int rtype::EntityTemplate::createHealthBonus (
    float x,
    float y,
    int id )
```

Create a Health Bonus object.

**Parameters**

<i>x</i>	
<i>y</i>	

**Returns**

int

**3.24.3.20 createHealthBonus() [2/2]**

```
int rtype::EntityTemplate::createHealthBonus (
    float x,
    float y,
    int id )
```

Create a Health Bonus object.

**Parameters**

<i>x</i>	
<i>y</i>	

**Returns**

int

### 3.24.3.21 createParallax() [1/2]

```
int rtype::EntityTemplate::createParallax (
    float x,
    float y,
    const std::string & textureName,
    float speed,
    int spriteLayer )
```

Create a Parallax object.

#### Parameters

<i>x</i>	position
<i>y</i>	position
<i>textureName</i>	for each assets
<i>speed</i>	the speed of each objects of the parallax
<i>spriteLayer</i>	the layer of the sprite

#### Returns

int

### 3.24.3.22 createParallax() [2/2]

```
int rtype::EntityTemplate::createParallax (
    float x,
    float y,
    const std::string & textureName,
    float speed,
    int spriteLayer )
```

Create a Parallax object.

#### Parameters

<i>x</i>	position
<i>y</i>	position
<i>textureName</i>	for each assets
<i>speed</i>	the speed of each objects of the parallax
<i>spriteLayer</i>	the layer of the sprite

#### Returns

int

### 3.24.3.23 createPlayer() [1/2]

```
int rtype::EntityTemplate::createPlayer (
    int playerId,
    bool isOwner,
    std::pair< float, float > pos )
```

Create a Player object.

#### Parameters

<i>playerId</i>	for multiplayer
<i>isOwner</i>	

#### Returns

int

### 3.24.3.24 createPlayer() [2/2]

```
int rtype::EntityTemplate::createPlayer (
    int playerId,
    bool isOwner,
    std::pair< float, float > pos )
```

Create a Player object.

#### Parameters

<i>playerId</i>	for multiplayer
<i>isOwner</i>	

#### Returns

int

### 3.24.3.25 createPlayerBullet() [1/2]

```
int rtype::EntityTemplate::createPlayerBullet (
    int x,
    int y )
```

Create a Player Bullet object.



**Parameters**

<i>x</i>	position
<i>y</i>	position

**Returns**

int

**3.24.3.26 createPlayerBullet() [2/2]**

```
int rtype::EntityTemplate::createPlayerBullet (
    int x,
    int y )
```

Create a Player Bullet object.

**Parameters**

<i>x</i>	position
<i>y</i>	position

**Returns**

int

**3.24.3.27 createSpecialPlayerBullet() [1/2]**

```
int rtype::EntityTemplate::createSpecialPlayerBullet (
    int x,
    int y )
```

Create a Special Player Bullet object when a bonus is triggered.

**Parameters**

<i>x</i>	position
<i>y</i>	position

**Returns**

int

**3.24.3.28 createSpecialPlayerBullet()** [2/2]

```
int rtype::EntityTemplate::createSpecialPlayerBullet (
    int x,
    int y )
```

Create a Special Player Bullet object when a bonus is triggered.

**Parameters**

<i>x</i>	position
<i>y</i>	position

**Returns**

int

**3.24.3.29 createSpecialShootBonus()** [1/2]

```
int rtype::EntityTemplate::createSpecialShootBonus (
    float x,
    float y,
    int id )
```

Create a Bonus object.

**Parameters**

<i>x</i>	position
<i>y</i>	position

**Returns**

int

**3.24.3.30 createSpecialShootBonus()** [2/2]

```
int rtype::EntityTemplate::createSpecialShootBonus (
    float x,
    float y,
    int id )
```

Create a Bonus object.

**Parameters**

<i>x</i>	position
<i>y</i>	position

**Returns**

int

**3.24.3.31 createTopBoundary() [1/2]**

```
int rtype::EntityTemplate::createTopBoundary (
    float x,
    float y )
```

Create a top boundary object.

**Parameters**

<i>x</i>	position
<i>y</i>	position

**Returns**

int

**3.24.3.32 createTopBoundary() [2/2]**

```
int rtype::EntityTemplate::createTopBoundary (
    float x,
    float y )
```

Create a top boundary object.

**Parameters**

<i>x</i>	position
<i>y</i>	position

**Returns**

int

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

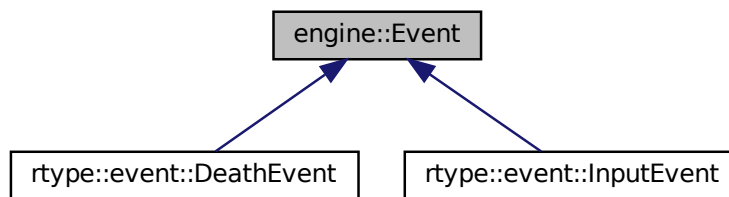
- `rtype-client/include/EntityTemplate.hpp`

## 3.25 engine::Event Class Reference

`Event` class which is the base class for all events.

```
#include <Event.hpp>
```

Inheritance diagram for `engine::Event`:



### 3.25.1 Detailed Description

`Event` class which is the base class for all events.

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

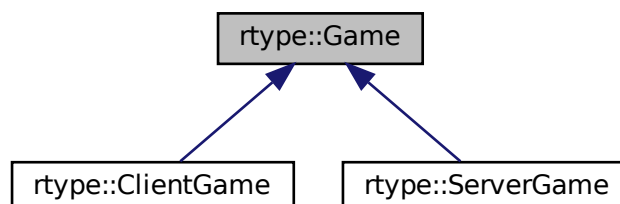
- `engine/include/ecs/Event.hpp`

## 3.26 rtype::Game Class Reference

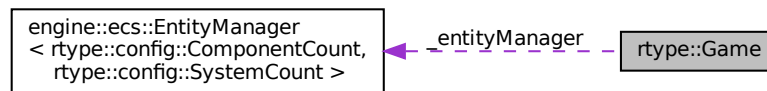
`Game`'s base, has to be completed in a child class.

```
#include <Game.hpp>
```

Inheritance diagram for `rtype::Game`:



Collaboration diagram for rtype::Game:



## Public Member Functions

- [Game](#) ()  
*Create base of the game.*
- virtual void [handleEvent](#) (rtype::NetworkEvent content)=0  
*Handle how to handle the received events.*
- [engine::ecs::EntityManager](#)< rtype::config::ComponentCount, rtype::config::SystemCount > & [getEntityManager](#) ()  
*Gets the entity manager of the game.*
- virtual void [gameLoop](#) ()=0  
*Game loop to repeat at every frame.*
- virtual bool [isOver](#) ()=0  
*Tells if the game is over.*
- virtual float [getFps](#) () const =0  
*Gets the fps of the game.*

## Protected Member Functions

- virtual void [gameUpdate](#) ()=0  
*Update the game.*
- virtual void [gameEvent](#) ()  
*Handle game events from the game (keyboard, mouse, ...)*

## Protected Attributes

- std::chrono::time\_point< std::chrono::steady\_clock > **\_clockStarting**
- double **\_dt**
- [engine::ecs::EntityManager](#)< rtype::config::ComponentCount, rtype::config::SystemCount > **\_entityManager**
- std::unordered\_map< int, engine::ecs::Entity > **\_players**

### 3.26.1 Detailed Description

[Game](#)'s base, has to be completed in a child class.

### 3.26.2 Constructor & Destructor Documentation

#### 3.26.2.1 Game()

`rtype::Game::Game ( )`

Create base of the game.

## Parameters

<i>queue</i>	Queue to fill if need to send package by network
--------------	--

### 3.26.3 Member Function Documentation

#### 3.26.3.1 getEntityManager()

```
engine::ecs::EntityManager<rtype::config::ComponentCount, rtype::config::SystemCount>& rtype↵  
::Game::getEntityManager ( )
```

Gets the entity manager of the game.

## Returns

The entity manager

#### 3.26.3.2 getFps()

```
virtual float rtype::Game::getFps ( ) const [pure virtual]
```

Gets the fps of the game.

## Returns

Float containing the fps

Implemented in [rtype::ServerGame](#), and [rtype::ClientGame](#).

#### 3.26.3.3 handleEvent()

```
virtual void rtype::Game::handleEvent (↵  
    rtype::NetworkEvent content ) [pure virtual]
```

Handle how to handle the received events.

## Parameters

<i>content</i>	Package to handle
----------------	-------------------

Implemented in [rtype::ServerGame](#), and [rtype::ClientGame](#).

### 3.26.3.4 isOver()

```
virtual bool rtype::Game::isOver ( ) [pure virtual]
```

Tells if the game is over.

#### Returns

Boolean saying if the game is over

Implemented in [rtype::ServerGame](#), and [rtype::ClientGame](#).

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

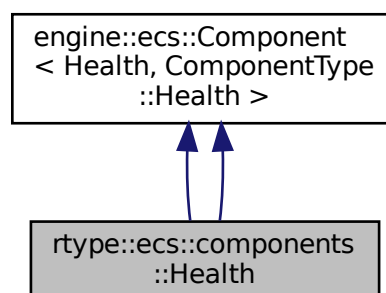
- game/include/Game.hpp

## 3.27 rtype::ecs::components::Health Struct Reference

[Health](#) struct which contains the health of the entity.

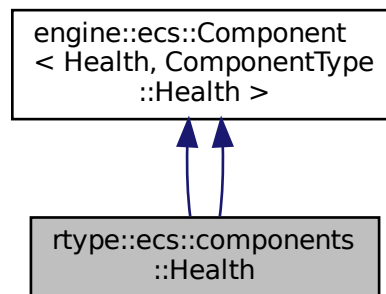
```
#include <Components.hpp>
```

Inheritance diagram for `rtype::ecs::components::Health`:





Collaboration diagram for rtype::ecs::components::Health:



### Public Member Functions

- **Health** (int Hp, const std::string &Tag)
- **Health** (int Hp, const std::string &Tag)

### Public Attributes

- int **hp**
- std::string **tag**

### Additional Inherited Members

#### 3.27.1 Detailed Description

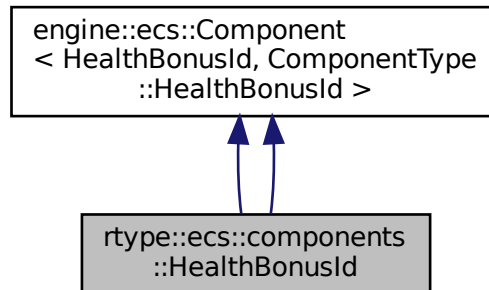
[Health](#) struct which contains the health of the entity.

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

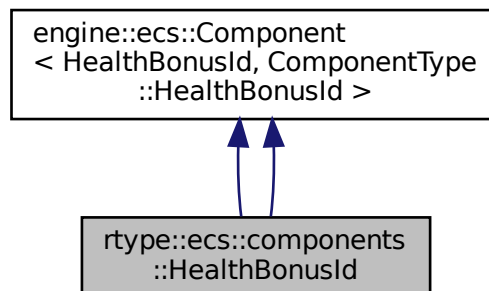
- `rtype-client/include/Components.hpp`

### 3.28 rtype::ecs::components::HealthBonusId Struct Reference

Inheritance diagram for rtype::ecs::components::HealthBonusId:



Collaboration diagram for rtype::ecs::components::HealthBonusId:



#### Public Member Functions

- **HealthBonusId** (int healthBonusId)
- **HealthBonusId** (int healthBonusId)

#### Public Attributes

- int **id**

## Additional Inherited Members

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

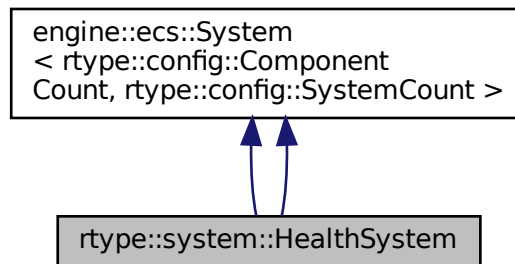
- rtype-client/include/Components.hpp

## 3.29 rtype::system::HealthSystem Class Reference

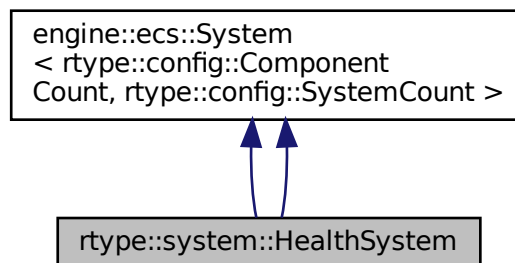
Health system class to check if an entity is dead or not.

```
#include <HealthSystem.hpp>
```

Inheritance diagram for rtype::system::HealthSystem:



Collaboration diagram for rtype::system::HealthSystem:



## Public Member Functions

- [HealthSystem](#) ([engine::ecs::EntityManager](#) < [rtype::config::ComponentCount](#), [rtype::config::SystemCount](#) > &[entityManager](#))  
*Construct a new Health System object.*
- void [update](#) (const float dt)  
*Update the Health System object.*
- [HealthSystem](#) ([engine::ecs::EntityManager](#) < [rtype::config::ComponentCount](#), [rtype::config::SystemCount](#) > &[entityManager](#))  
*Construct a new Health System object.*
- void [update](#) (const float dt)  
*Update the Health System object.*

## Additional Inherited Members

### 3.29.1 Detailed Description

Health system class to check if an entity is dead or not.

### 3.29.2 Constructor & Destructor Documentation

#### 3.29.2.1 HealthSystem() [1/2]

```
rtype::system::HealthSystem::HealthSystem (
    engine::ecs::EntityManager < rtype::config::ComponentCount, rtype::config::↵
    SystemCount > & entityManager ) [inline]
```

Construct a new Health System object.

#### Parameters

<a href="#">entityManager</a>	
-------------------------------	--

#### 3.29.2.2 HealthSystem() [2/2]

```
rtype::system::HealthSystem::HealthSystem (
    engine::ecs::EntityManager < rtype::config::ComponentCount, rtype::config::↵
    SystemCount > & entityManager ) [inline]
```

Construct a new Health System object.

## Parameters

<i>entityManager</i>	
----------------------	--

### 3.29.3 Member Function Documentation

#### 3.29.3.1 update() [1/2]

```
void rtype::system::HealthSystem::update (
    const float dt ) [inline], [virtual]
```

Update the Health System object.

## Parameters

<i>dt</i>	
-----------	--

Reimplemented from [engine::ecs::System< rtype::config::ComponentCount, rtype::config::SystemCount >](#).

#### 3.29.3.2 update() [2/2]

```
void rtype::system::HealthSystem::update (
    const float dt ) [inline], [virtual]
```

Update the Health System object.

## Parameters

<i>dt</i>	
-----------	--

Reimplemented from [engine::ecs::System< rtype::config::ComponentCount, rtype::config::SystemCount >](#).

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

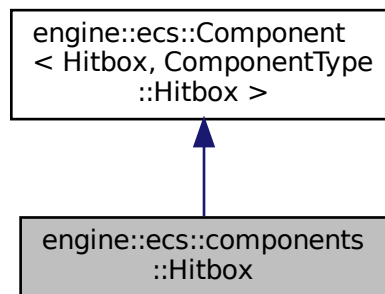
- rtype-client/include/systems/HealthSystem.hpp

## 3.30 engine::ecs::components::Hitbox Struct Reference

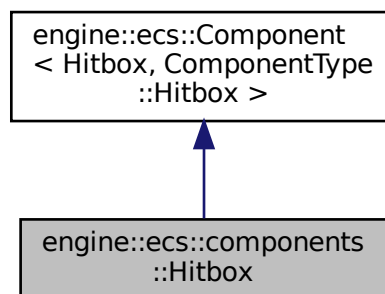
[Hitbox](#) component.

```
#include <Components.hpp>
```

Inheritance diagram for engine::ecs::components::Hitbox:



Collaboration diagram for engine::ecs::components::Hitbox:



## Public Member Functions

- **Hitbox** (float Width, float Height, bool IsTrigger=false)
- **Hitbox** (float Width, float Height, const std::string &Tag, const std::string &TriggerTags, bool IsTrigger=false)

## Public Attributes

- float **width**
- float **height**
- bool **collided**
- bool **isTrigger**
- std::string **tag**
- std::string **triggerTags**

## Additional Inherited Members

### 3.30.1 Detailed Description

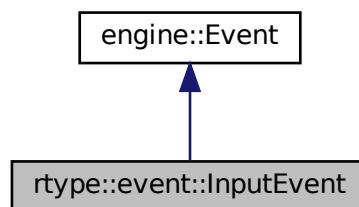
[Hitbox](#) component.

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

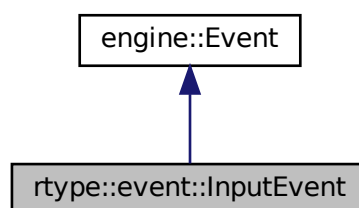
- engine/include/ecs/Components.hpp

## 3.31 rtype::event::InputEvent Class Reference

Inheritance diagram for rtype::event::InputEvent:



Collaboration diagram for rtype::event::InputEvent:



## Public Attributes

- int **key**
- int **playerId**

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

- game/include/events/InputEvent.hpp

## 3.32 rtype::LevelManager Class Reference

Level manager class to manage the creation of levels.

```
#include <LevelManager.hpp>
```

### Public Member Functions

- [LevelManager](#) ([engine::ecs::EntityManager](#)< rtype::config::ComponentCount, rtype::config::SystemCount > &entityManager)  
*Construct a new Level Manager object.*
- void [createLevel](#) (int level)  
*Create a Level object.*
- void [createBossLevel](#) (int level)  
*Create a Boss Level object.*
- int [getLevel](#) () const  
*Get the Level object.*
- void [setLevel](#) (int level)
- [LevelManager](#) ([engine::ecs::EntityManager](#)< rtype::config::ComponentCount, rtype::config::SystemCount > &entityManager)  
*Construct a new Level Manager object.*
- void [createLevel](#) (int level)  
*Create a Level object.*
- void [createBossLevel](#) (int level)  
*Create a Boss Level object.*
- int [getLevel](#) () const  
*Get the Level object.*
- void [setLevel](#) (int level)

### 3.32.1 Detailed Description

Level manager class to manage the creation of levels.

### 3.32.2 Constructor & Destructor Documentation

#### 3.32.2.1 LevelManager() [1/2]

```
rtype::LevelManager::LevelManager (
    engine::ecs::EntityManager< rtype::config::ComponentCount, rtype::config::↵
SystemCount > & entityManager )
```

Construct a new Level Manager object.



## Parameters

<i>entityManager</i>	
----------------------	--

**3.32.2.2 LevelManager()** [2/2]

```
rtype::LevelManager::LevelManager (
    engine::ecs::EntityManager< rtype::config::ComponentCount, rtype::config::↵
    SystemCount > & entityManager )
```

Construct a new Level Manager object.

## Parameters

<i>entityManager</i>	
----------------------	--

**3.32.3 Member Function Documentation****3.32.3.1 createBossLevel()** [1/2]

```
void rtype::LevelManager::createBossLevel (
    int level )
```

Create a Boss Level object.

## Parameters

<i>level</i>	
--------------	--

**3.32.3.2 createBossLevel()** [2/2]

```
void rtype::LevelManager::createBossLevel (
    int level )
```

Create a Boss Level object.

## Parameters

<i>level</i>	
--------------	--

**3.32.3.3 createLevel()** [1/2]

```
void rtype::LevelManager::createLevel (
    int level )
```

Create a Level object.

**Parameters**

<i>level</i>	
--------------	--

**3.32.3.4 createLevel()** [2/2]

```
void rtype::LevelManager::createLevel (
    int level )
```

Create a Level object.

**Parameters**

<i>level</i>	
--------------	--

**3.32.3.5 getLevel()** [1/2]

```
int rtype::LevelManager::getLevel ( ) const
```

Get the Level object.

**Returns**

int

**3.32.3.6 getLevel()** [2/2]

```
int rtype::LevelManager::getLevel ( ) const
```

Get the Level object.

**Returns**

int

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

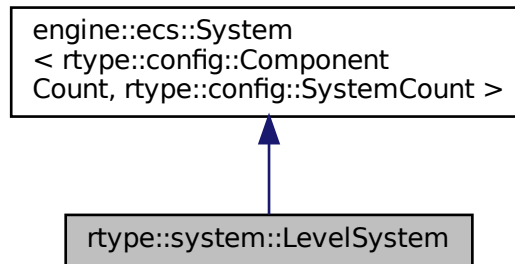
- rtype-client/include/LevelManager.hpp

### 3.33 rtype::system::LevelSystem Class Reference

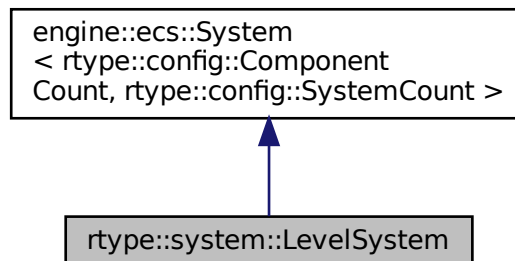
[LevelSystem](#) class which is used to have the operation of stages.

```
#include <LevelSystem.hpp>
```

Inheritance diagram for rtype::system::LevelSystem:



Collaboration diagram for rtype::system::LevelSystem:



#### Public Member Functions

- [LevelSystem](#) ([engine::ecs::EntityManager](#)< `rtype::config::ComponentCount`, `rtype::config::SystemCount` > &`entityManager`, [rtype::LevelManager](#) `levelManager`)  
*Construct a new Level System object.*
- void [update](#) (`const float dt`)  
*Update function to get levels.*

## Additional Inherited Members

### 3.33.1 Detailed Description

[LevelSystem](#) class which is used to have the operation of stages.

### 3.33.2 Constructor & Destructor Documentation

#### 3.33.2.1 LevelSystem()

```
rtype::system::LevelSystem::LevelSystem (
    engine::ecs::EntityManager< rtype::config::ComponentCount, rtype::config::↵
SystemCount > & entityManager,
    rtype::LevelManager levelManager ) [inline]
```

Construct a new Level System object.

##### Parameters

<i>entityManager</i>	
<i>levelManager</i>	

### 3.33.3 Member Function Documentation

#### 3.33.3.1 update()

```
void rtype::system::LevelSystem::update (
    const float dt ) [inline], [virtual]
```

Update function to get levels.

##### Parameters

<i>dt</i>	
-----------	--

Reimplemented from [engine::ecs::System< rtype::config::ComponentCount, rtype::config::SystemCount >](#).

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

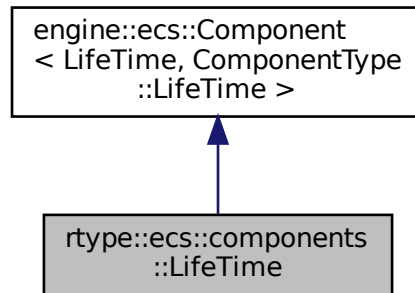
- `game/include/systems/LevelSystem.hpp`

## 3.34 rtype::ecs::components::LifeTime Struct Reference

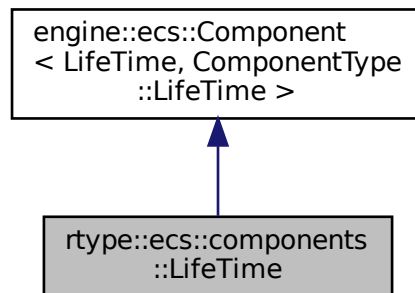
[LifeTime](#) struct which contains the life time info of the entity.

```
#include <Components.hpp>
```

Inheritance diagram for rtype::ecs::components::LifeTime:



Collaboration diagram for rtype::ecs::components::LifeTime:



### Public Member Functions

- **LifeTime** (float lifeTime)

### Public Attributes

- float **lifeTime**

## Additional Inherited Members

### 3.34.1 Detailed Description

[LifeTime](#) struct which contains the life time info of the entity.

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

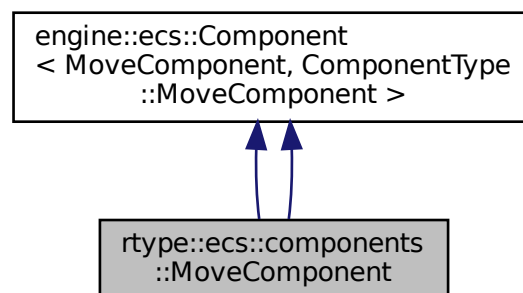
- `rtype-client/include/Components.hpp`

## 3.35 `rtype::ecs::components::MoveComponent` Struct Reference

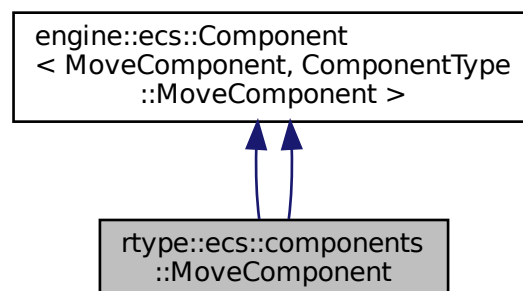
[MoveComponent](#) struct which contains the movement of the entity.

```
#include <Components.hpp>
```

Inheritance diagram for `rtype::ecs::components::MoveComponent`:



Collaboration diagram for `rtype::ecs::components::MoveComponent`:



## Additional Inherited Members

### 3.35.1 Detailed Description

[MoveComponent](#) struct which contains the movement of the entity.

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

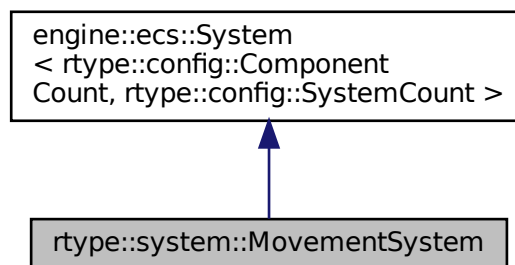
- `rtype-client/include/Components.hpp`

## 3.36 rtype::system::MovementSystem Class Reference

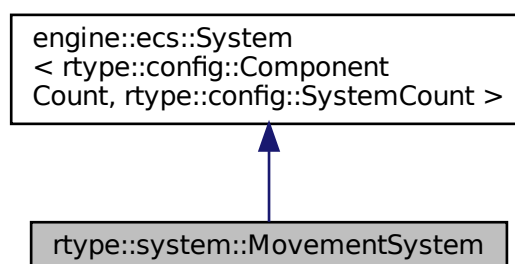
Movement system class used to move the entities.

```
#include <MovementSystem.hpp>
```

Inheritance diagram for `rtype::system::MovementSystem`:



Collaboration diagram for `rtype::system::MovementSystem`:



## Public Member Functions

- [MovementSystem](#) ([engine::ecs::EntityManager](#) < [rtype::config::ComponentCount](#), [rtype::config::SystemCount](#) > &[entityManager](#))  
Construct a new Movement System object.
- void [handleEvent](#) (const [engine::Event](#) &[event](#))  
Handle the events of the Movement System object.

## Additional Inherited Members

### 3.36.1 Detailed Description

Movement system class used to move the entities.

### 3.36.2 Constructor & Destructor Documentation

#### 3.36.2.1 MovementSystem()

```
rtype::system::MovementSystem::MovementSystem (
    engine::ecs::EntityManager < rtype::config::ComponentCount, rtype::config::SystemCount > & entityManager ) [inline]
```

Construct a new Movement System object.

#### Parameters

<a href="#">entityManager</a>	
-------------------------------	--

### 3.36.3 Member Function Documentation

#### 3.36.3.1 handleEvent()

```
void rtype::system::MovementSystem::handleEvent (
    const engine::Event & event ) [inline], [virtual]
```

Handle the events of the Movement System object.

#### Parameters

<a href="#">event</a>	
-----------------------	--



Reimplemented from [engine::ecs::System< rtype::config::ComponentCount, rtype::config::SystemCount >](#).

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

- `game/include/systems/MovementSystem.hpp`

## 3.37 rtype::NetworkEvent Struct Reference

Package structure.

```
#include <QueueManager.hpp>
```

### Public Attributes

- NetworkEventType [type](#)
- int [id](#)
- int [key](#)
- std::pair< float, float > [pos](#)
- ObjectType [objectType](#)

### 3.37.1 Detailed Description

Package structure.

### 3.37.2 Member Data Documentation

#### 3.37.2.1 id

```
int rtype::NetworkEvent::id
```

Specifies the ID of the entity concerned

#### 3.37.2.2 key

```
int rtype::NetworkEvent::key
```

Specifies if a key (of the keyboard) has been touched

#### 3.37.2.3 objectType

```
ObjectType rtype::NetworkEvent::objectType
```

Type of the object concerned (if one)

### 3.37.2.4 pos

```
std::pair<float, float> rtype::NetworkEvent::pos
```

Position of the object (if one)

### 3.37.2.5 type

```
NetworkEventType rtype::NetworkEvent::type
```

Specifies how the package is to be treated

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

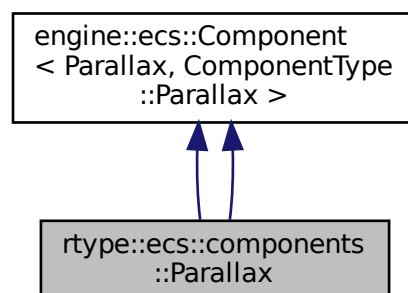
- game/include/QueueManager.hpp

## 3.38 rtype::ecs::components::Parallax Struct Reference

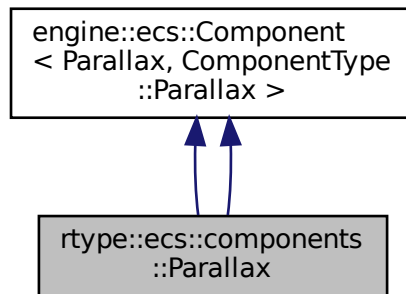
[Parallax](#) struct which contains the parallax of the entity.

```
#include <Components.hpp>
```

Inheritance diagram for rtype::ecs::components::Parallax:



Collaboration diagram for rtype::ecs::components::Parallax:



## Additional Inherited Members

### 3.38.1 Detailed Description

[Parallax](#) struct which contains the parallax of the entity.

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

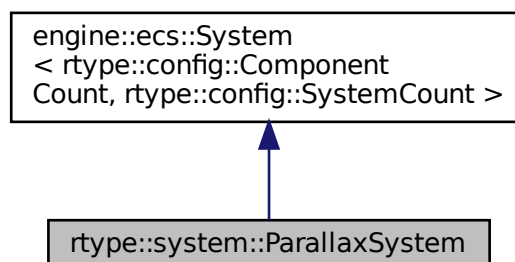
- `rtype-client/include/Components.hpp`

## 3.39 rtype::system::ParallaxSystem Class Reference

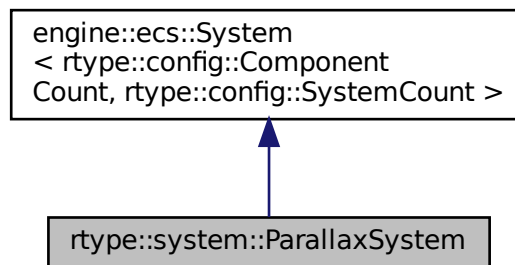
[ParallaxSystem](#) used to spin the parallax.

```
#include <ParallaxSystem.hpp>
```

Inheritance diagram for `rtype::system::ParallaxSystem`:



Collaboration diagram for `rtype::system::ParallaxSystem`:



## Public Member Functions

- [ParallaxSystem](#) ([engine::ecs::EntityManager](#)< `rtype::config::ComponentCount`, `rtype::config::SystemCount` > &`entityManager`)  
Construct a new *Parallax System* object.
- void [update](#) (const float dt)  
Update the *parallax system* so that it spins infinitely.

## Additional Inherited Members

### 3.39.1 Detailed Description

[ParallaxSystem](#) used to spin the parallax.

### 3.39.2 Constructor & Destructor Documentation

#### 3.39.2.1 ParallaxSystem()

```

rtype::system::ParallaxSystem::ParallaxSystem (
    engine::ecs::EntityManager< rtype::config::ComponentCount, rtype::config::SystemCount > & entityManager ) [inline]

```

Construct a new *Parallax System* object.

#### Parameters

<code>entityManager</code>	
----------------------------	--

### 3.39.3 Member Function Documentation

#### 3.39.3.1 update()

```
void rtype::system::ParallaxSystem::update (
    const float dt ) [inline], [virtual]
```

Update the parallax system so that it spins infinitely.

##### Parameters

<i>dt</i>	
-----------	--

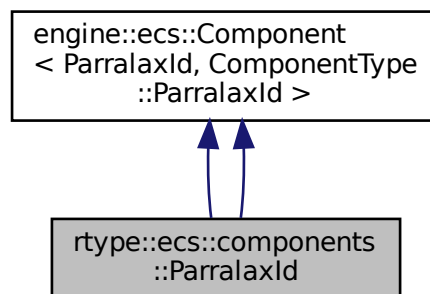
Reimplemented from [engine::ecs::System< rtype::config::ComponentCount, rtype::config::SystemCount >](#).

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

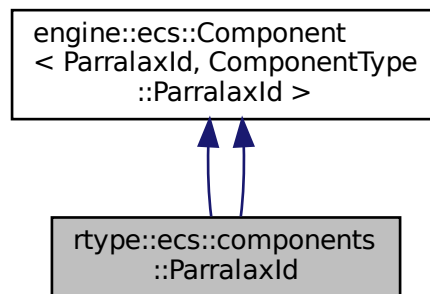
- game/include/systems/ParallaxSystem.hpp

## 3.40 rtype::ecs::components::ParralaxId Struct Reference

Inheritance diagram for rtype::ecs::components::ParralaxId:



Collaboration diagram for `rtype::ecs::components::ParralaxId`:



### Public Member Functions

- **ParralaxId** (int parralaxId)
- **ParralaxId** (int parralaxId)

### Public Attributes

- int **id**

### Additional Inherited Members

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

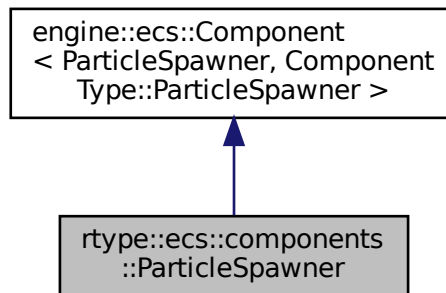
- `rtype-client/include/Components.hpp`

## 3.41 `rtype::ecs::components::ParticleSpawner` Struct Reference

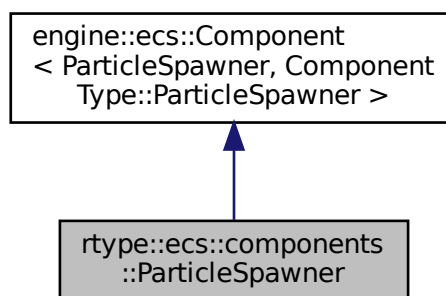
[ParticleSpawner](#) struct which contains the particle spawner info of the entity.

```
#include <Components.hpp>
```

Inheritance diagram for rtype::ecs::components::ParticleSpawner:



Collaboration diagram for rtype::ecs::components::ParticleSpawner:



## Public Member Functions

- **ParticleSpawner** (const std::string &configFilePath)

## Public Attributes

- int **nbParticles**
- int **nbParticlesSpawned**
- float **timer**
- float **timeBetweenSpawn**
- float **particleLifeTime**
- sf::Vector2f **rangeVelocityX**
- sf::Vector2f **rangeVelocityY**
- sf::Vector2f **rangeSpawnX**
- sf::Vector2f **rangeSpawnY**
- bool **isLooping**
- float **fadeScale**
- sf::Sprite **sprite**
- int **spriteLayout**

## Additional Inherited Members

### 3.41.1 Detailed Description

[ParticleSystem](#) struct which contains the particle spawner info of the entity.

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

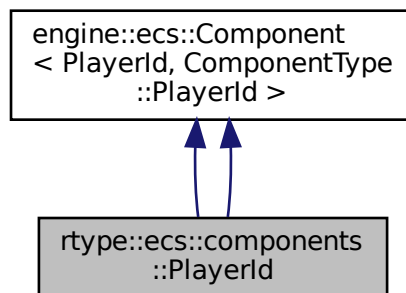
- `rtype-client/include/Components.hpp`

## 3.42 `rtype::ecs::components::PlayerId` Struct Reference

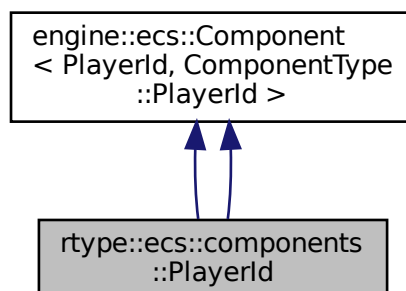
[PlayerId](#) struct which contains the player id.

```
#include <Components.hpp>
```

Inheritance diagram for `rtype::ecs::components::PlayerId`:



Collaboration diagram for `rtype::ecs::components::PlayerId`:





## Public Member Functions

- **PlayerId** (int playerId, bool owner)
- **PlayerId** (int playerId, bool owner)

## Public Attributes

- int **id**
- bool **isOwner**

## Additional Inherited Members

### 3.42.1 Detailed Description

[PlayerId](#) struct which contains the player id.

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

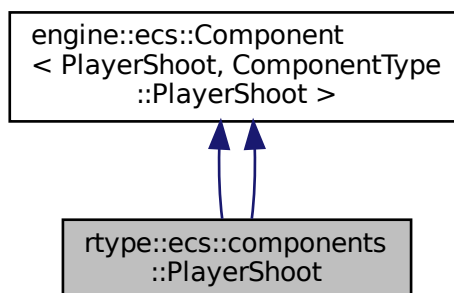
- rtype-client/include/Components.hpp

## 3.43 rtype::ecs::components::PlayerShoot Struct Reference

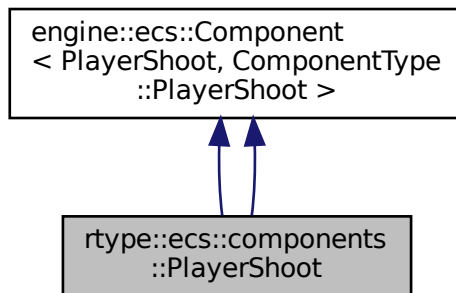
[PlayerShoot](#) struct which contains the player shoot.

```
#include <Components.hpp>
```

Inheritance diagram for rtype::ecs::components::PlayerShoot:



Collaboration diagram for `rtype::ecs::components::PlayerShoot`:



### Public Attributes

- `bool isSpecialBulletBonusActivated`

### Additional Inherited Members

#### 3.43.1 Detailed Description

[PlayerShoot](#) struct which contains the player shoot.

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

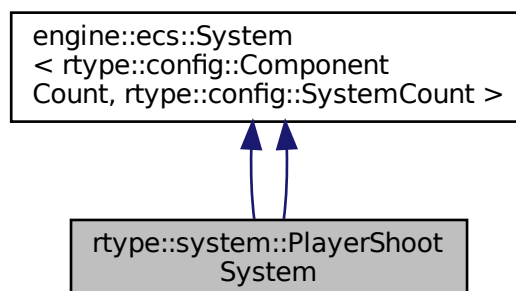
- `rtype-client/include/Components.hpp`

## 3.44 rtype::system::PlayerShootSystem Class Reference

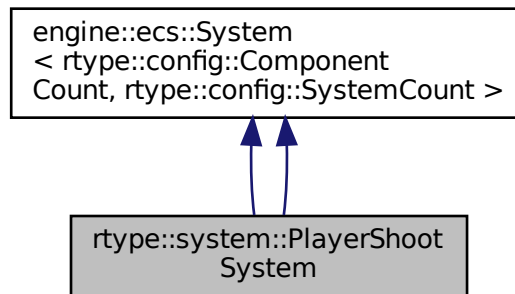
PlayerShoot system class used to let the player shoot.

```
#include <PlayerShootSystem.hpp>
```

Inheritance diagram for `rtype::system::PlayerShootSystem`:



Collaboration diagram for rtype::system::PlayerShootSystem:



## Public Member Functions

- [PlayerShootSystem](#) ([engine::ecs::EntityManager](#) < `rtype::config::ComponentCount`, `rtype::config::SystemCount` > &entityManager)  
Construct a new Player Shoot System object.
- void [handleEvent](#) (const [engine::Event](#) &event)  
Handle the events of the Player Shoot System object.
- [PlayerShootSystem](#) ([engine::ecs::EntityManager](#) < `rtype::config::ComponentCount`, `rtype::config::SystemCount` > &entityManager)  
Construct a new Player Shoot System object.
- void [handleEvent](#) (const [engine::Event](#) &event)  
Handle the events of the Player Shoot System object.

## Additional Inherited Members

### 3.44.1 Detailed Description

PlayerShoot system class used to let the player shoot.

### 3.44.2 Constructor & Destructor Documentation

#### 3.44.2.1 PlayerShootSystem() [1/2]

```

rtype::system::PlayerShootSystem::PlayerShootSystem (
    engine::ecs::EntityManager < rtype::config::ComponentCount, rtype::config::SystemCount > & entityManager ) [inline]
  
```

Construct a new Player Shoot System object.

## Parameters

<i>entityManager</i>	
----------------------	--

**3.44.2.2 PlayerShootSystem()** [2/2]

```

rtype::system::PlayerShootSystem::PlayerShootSystem (
    engine::ecs::EntityManager< rtype::config::ComponentCount, rtype::config::SystemCount > & entityManager ) [inline]

```

Construct a new Player Shoot System object.

## Parameters

<i>entityManager</i>	
----------------------	--

**3.44.3 Member Function Documentation****3.44.3.1 handleEvent()** [1/2]

```

void rtype::system::PlayerShootSystem::handleEvent (
    const engine::Event & event ) [inline], [virtual]

```

Handle the events of the Player Shoot System object.

## Parameters

<i>event</i>	
--------------	--

Reimplemented from [engine::ecs::System< rtype::config::ComponentCount, rtype::config::SystemCount >](#).

**3.44.3.2 handleEvent()** [2/2]

```

void rtype::system::PlayerShootSystem::handleEvent (
    const engine::Event & event ) [inline], [virtual]

```

Handle the events of the Player Shoot System object.

## Parameters

<i>event</i>	
--------------	--

Reimplemented from [engine::ecs::System< rtype::config::ComponentCount, rtype::config::SystemCount >](#).

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

- `rtype-client/include/systems/PlayerShootSystem.hpp`

## 3.45 rtype::QueueManager Class Reference

### Public Member Functions

- void **pushInQueue** ([NetworkEvent](#) event)
- [NetworkEvent](#) **popEvent** ()
- bool **isEmpty** ()

### Static Public Member Functions

- static [QueueManager](#) & **getInstance** ()

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

- `game/include/QueueManager.hpp`

## 3.46 rtype::ScoreManager Class Reference

Score manager class to manage the score.

```
#include <ScoreManager.hpp>
```

### Public Member Functions

- [ScoreManager](#) ([engine::ecs::EntityManager](#)< rtype::config::ComponentCount, rtype::config::SystemCount > &entityManager)  
*Construct a new Score Manager object.*
- void [displayScore](#) (sf::RenderWindow &window)  
*Display the score.*
- void [updateScore](#) (int score)  
*Update the score.*
- int [getScore](#) () const  
*Get the Score object.*
- [ScoreManager](#) ([engine::ecs::EntityManager](#)< rtype::config::ComponentCount, rtype::config::SystemCount > &entityManager)  
*Construct a new Score Manager object.*
- void [displayScore](#) (sf::RenderWindow &window)  
*Display the score.*
- void [updateScore](#) (int score)  
*Update the score.*
- int [getScore](#) () const  
*Get the Score object.*

### 3.46.1 Detailed Description

Score manager class to manage the score.

### 3.46.2 Constructor & Destructor Documentation

#### 3.46.2.1 ScoreManager() [1/2]

```
rtype::ScoreManager::ScoreManager (
    engine::ecs::EntityManager< rtype::config::ComponentCount, rtype::config::↵
SystemCount > & entityManager )
```

Construct a new Score Manager object.

Parameters

<i>entityManager</i>	
----------------------	--

#### 3.46.2.2 ScoreManager() [2/2]

```
rtype::ScoreManager::ScoreManager (
    engine::ecs::EntityManager< rtype::config::ComponentCount, rtype::config::↵
SystemCount > & entityManager )
```

Construct a new Score Manager object.

Parameters

<i>entityManager</i>	
----------------------	--

### 3.46.3 Member Function Documentation

#### 3.46.3.1 displayScore() [1/2]

```
void rtype::ScoreManager::displayScore (
    sf::RenderWindow & window )
```

Display the score.

## Parameters

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

**3.46.3.2 displayScore()** [2/2]

```
void rtype::ScoreManager::displayScore (
    sf::RenderWindow & window )
```

Display the score.

## Parameters

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

**3.46.3.3 getScore()** [1/2]

```
int rtype::ScoreManager::getScore ( ) const
```

Get the Score object.

## Returns

int

**3.46.3.4 getScore()** [2/2]

```
int rtype::ScoreManager::getScore ( ) const
```

Get the Score object.

## Returns

int

**3.46.3.5 updateScore()** [1/2]

```
void rtype::ScoreManager::updateScore (
    int score )
```

Update the score.

## Parameters

<i>score</i>	
--------------	--

**3.46.3.6 updateScore() [2/2]**

```
void rtype::ScoreManager::updateScore (
    int score )
```

Update the score.

## Parameters

<i>score</i>	
--------------	--

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

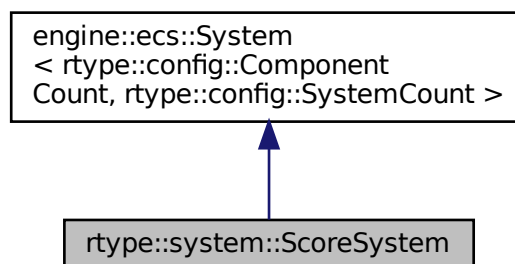
- rtype-client/include/ScoreManager.hpp

**3.47 rtype::system::ScoreSystem Class Reference**

[ScoreSystem](#) class to have the score the players.

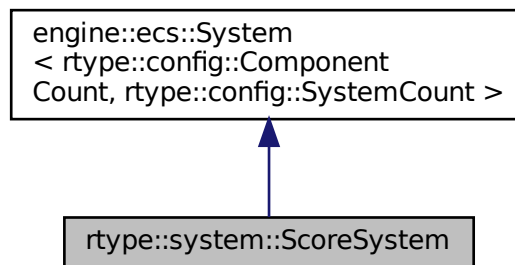
```
#include <ScoreSystem.hpp>
```

Inheritance diagram for rtype::system::ScoreSystem:





Collaboration diagram for rtype::system::ScoreSystem:



## Public Member Functions

- `ScoreSystem` (`engine::ecs::EntityManager` `< rtype::config::ComponentCount, rtype::config::SystemCount >` `&entityManager`, `rtype::ScoreManager` `scoreManager`)  
Construct a new Score System object.
- void `update` (`const float dt`)  
Update the scoreSystem class have the score.
- void `draw` (`engine::graphical::GraphicalWindow &window`)  
Draw the score.

## Additional Inherited Members

### 3.47.1 Detailed Description

`ScoreSystem` class to have the score the players.

### 3.47.2 Constructor & Destructor Documentation

#### 3.47.2.1 ScoreSystem()

```

rtype::system::ScoreSystem::ScoreSystem (
    engine::ecs::EntityManager< rtype::config::ComponentCount, rtype::config::SystemCount > &entityManager,
    rtype::ScoreManager scoreManager ) [inline]
  
```

Construct a new Score System object.

## Parameters

<i>entityManager</i>	
<i>scoreManager</i>	

### 3.47.3 Member Function Documentation

#### 3.47.3.1 draw()

```
void rtype::system::ScoreSystem::draw (
    engine::graphical::GraphicalWindow & window ) [inline], [virtual]
```

Draw the score.

## Parameters

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

Reimplemented from [engine::ecs::System< rtype::config::ComponentCount, rtype::config::SystemCount >](#).

#### 3.47.3.2 update()

```
void rtype::system::ScoreSystem::update (
    const float dt ) [inline], [virtual]
```

Update the scoreSystem class have the score.

## Parameters

<i>dt</i>	
-----------	--

Reimplemented from [engine::ecs::System< rtype::config::ComponentCount, rtype::config::SystemCount >](#).

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

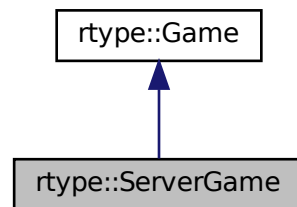
- `game/include/systems/ScoreSystem.hpp`

## 3.48 rtype::ServerGame Class Reference

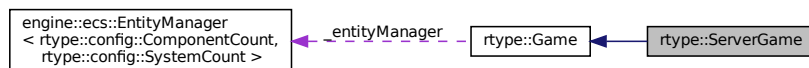
Instance of the server version of the game (without graphics but having all powers)

```
#include <ServerGame.hpp>
```

Inheritance diagram for rtype::ServerGame:



Collaboration diagram for rtype::ServerGame:



## Public Member Functions

- [ServerGame](#) ()  
*Constructs a game environment without any player.*
- void [gameLoop](#) () override  
*Game loop of the game to repeat while the game isn't over.*
- bool [isOver](#) () override  
*Says if the game is over or not.*
- float [getFps](#) () const override  
*Gets you the Fps.*
- void [handleEvent](#) (rtype::NetworkEvent content) override  
*Handle the events coming from the clients.*
- void [gameUpdate](#) () override  
*Update the game.*

## Additional Inherited Members

### 3.48.1 Detailed Description

Instance of the server version of the game (without graphics but having all powers)

## 3.48.2 Constructor & Destructor Documentation

### 3.48.2.1 ServerGame()

```
rtype::ServerGame::ServerGame ( )
```

Constructs a game environment without any player.

#### Parameters

<i>queue</i>	Queue of the server to fill if there are information to send to the clients
--------------	---

## 3.48.3 Member Function Documentation

### 3.48.3.1 getFps()

```
float rtype::ServerGame::getFps ( ) const [override], [virtual]
```

Gets you the Fps.

#### Returns

Float representation the actual fps (always -1 for the server)

Implements [rtype::Game](#).

### 3.48.3.2 handleEvent()

```
void rtype::ServerGame::handleEvent (
    rtype::NetworkEvent content ) [override], [virtual]
```

Handle the events coming from the clients.

#### Parameters

<i>content</i>	Package containing the information from the client
----------------	--

Implements [rtype::Game](#).

### 3.48.3.3 isOver()

```
bool rtype::ServerGame::isOver ( ) [override], [virtual]
```

Says if the game is over or not.

#### Returns

Boolean that represent the game state

Implements [rtype::Game](#).

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

- `rtype-server/include/ServerGame.hpp`

## 3.49 server::serverInstance Class Reference

Instance of server handling a game.

```
#include <serverInstance.hpp>
```

### Public Member Functions

- [serverInstance](#) ()  
*Create a server on port 2001.*
- void [lifeCycle](#) ()  
*Handle the life cycle of the server.*

### 3.49.1 Detailed Description

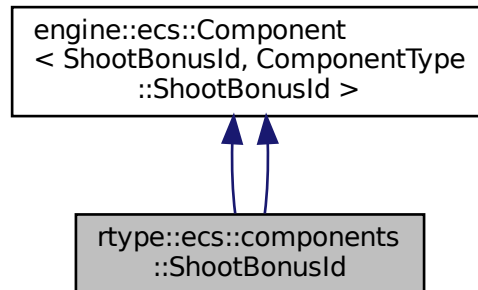
Instance of server handling a game.

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

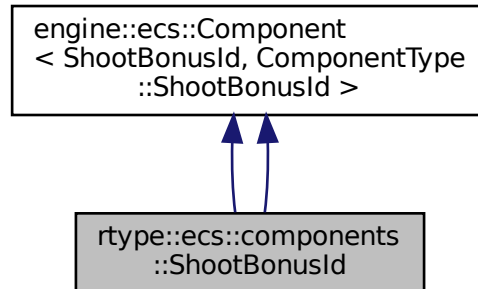
- `rtype-server/include/serverInstance.hpp`

### 3.50 rtype::ecs::components::ShootBonusId Struct Reference

Inheritance diagram for rtype::ecs::components::ShootBonusId:



Collaboration diagram for rtype::ecs::components::ShootBonusId:



#### Public Member Functions

- **ShootBonusId** (int shootBonusId)
- **ShootBonusId** (int shootBonusId)

#### Public Attributes

- int id

## Additional Inherited Members

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

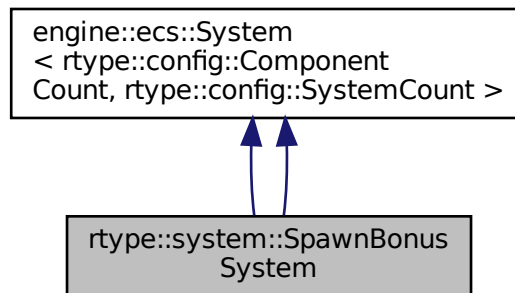
- rtype-client/include/Components.hpp

## 3.51 rtype::system::SpawnBonusSystem Class Reference

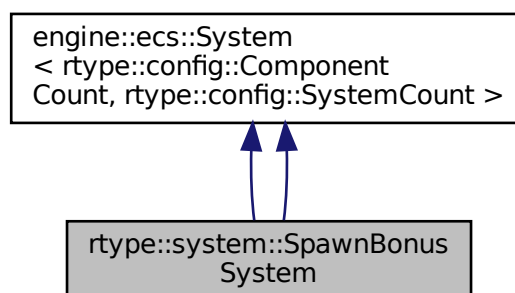
Clean entity system class to remove entities.

```
#include <SpawnBonusSystem.hpp>
```

Inheritance diagram for rtype::system::SpawnBonusSystem:



Collaboration diagram for rtype::system::SpawnBonusSystem:



## Public Member Functions

- [SpawnBonusSystem](#) ([engine::ecs::EntityManager](#)< [rtype::config::ComponentCount](#), [rtype::config::SystemCount](#) > &[entityManager](#))  
Construct a new Clean Entity System object.
- void [update](#) (const float dt)  
Update the Spawn Bonus System object.
- [SpawnBonusSystem](#) ([engine::ecs::EntityManager](#)< [rtype::config::ComponentCount](#), [rtype::config::SystemCount](#) > &[entityManager](#))  
Construct a new Clean Entity System object.
- void [update](#) (const float dt)  
Update the Spawn Bonus System object.

## Additional Inherited Members

### 3.51.1 Detailed Description

Clean entity system class to remove entities.

### 3.51.2 Constructor & Destructor Documentation

#### 3.51.2.1 SpawnBonusSystem() [1/2]

```
rtype::system::SpawnBonusSystem::SpawnBonusSystem (
    engine::ecs::EntityManager< rtype::config::ComponentCount, rtype::config::SystemCount > & entityManager ) [inline]
```

Construct a new Clean Entity System object.

#### Parameters

<a href="#">entityManager</a>	
-------------------------------	--

#### 3.51.2.2 SpawnBonusSystem() [2/2]

```
rtype::system::SpawnBonusSystem::SpawnBonusSystem (
    engine::ecs::EntityManager< rtype::config::ComponentCount, rtype::config::SystemCount > & entityManager ) [inline]
```

Construct a new Clean Entity System object.



## Parameters

<i>entityManager</i>	
----------------------	--

### 3.51.3 Member Function Documentation

#### 3.51.3.1 update() [1/2]

```
void rtype::system::SpawnBonusSystem::update (
    const float dt ) [inline], [virtual]
```

Update the Spawn Bonus System object.

## Parameters

<i>dt</i>	
-----------	--

Reimplemented from [engine::ecs::System< rtype::config::ComponentCount, rtype::config::SystemCount >](#).

#### 3.51.3.2 update() [2/2]

```
void rtype::system::SpawnBonusSystem::update (
    const float dt ) [inline], [virtual]
```

Update the Spawn Bonus System object.

## Parameters

<i>dt</i>	
-----------	--

Reimplemented from [engine::ecs::System< rtype::config::ComponentCount, rtype::config::SystemCount >](#).

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

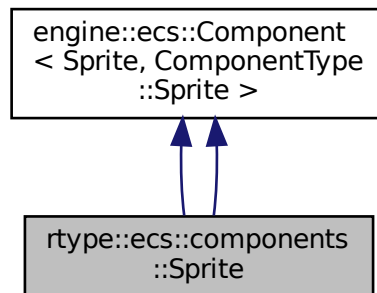
- [rtype-client/include/systems/SpawnBonusSystem.hpp](#)

## 3.52 rtype::ecs::components::Sprite Struct Reference

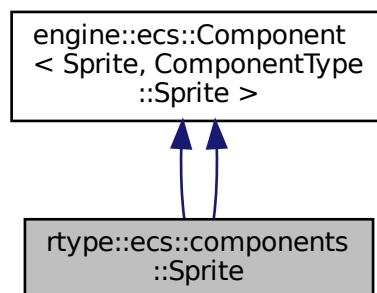
[Sprite](#) component.

```
#include <Components.hpp>
```

Inheritance diagram for `rtype::ecs::components::Sprite`:



Collaboration diagram for `rtype::ecs::components::Sprite`:



## Public Member Functions

- **Sprite** (const `sf::Texture` &texture, int spriteLayer=0, `sf::Vector2< float >` scale=`sf::Vector2< float >(1.0f, 1.0f)`)
- **Sprite** (const `sf::Texture` &texture, const `sf::IntRect` &rect, int spriteLayer=0, `sf::Vector2< float >` scale=`sf::Vector2< float >(1.0f, 1.0f)`)
- **Sprite** (const `sf::Sprite` &sprite, int spriteLayer=0)
- **Sprite** (const `sf::Texture` &texture, int spriteLayer=0, `sf::Vector2< float >` scale=`sf::Vector2< float >(1.0f, 1.0f)`)
- **Sprite** (const `sf::Texture` &texture, const `sf::IntRect` &rect, int spriteLayer=0, `sf::Vector2< float >` scale=`sf::Vector2< float >(1.0f, 1.0f)`)
- **Sprite** (const `sf::Sprite` &sprite, int spriteLayer=0)

## Public Attributes

- `sf::Sprite` **sprite**
- int **layer**

## Additional Inherited Members

### 3.52.1 Detailed Description

[Sprite](#) component.

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

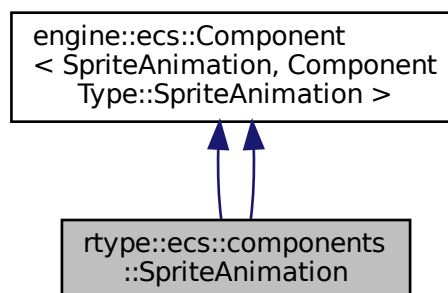
- `rtype-client/include/Components.hpp`

## 3.53 rtype::ecs::components::SpriteAnimation Struct Reference

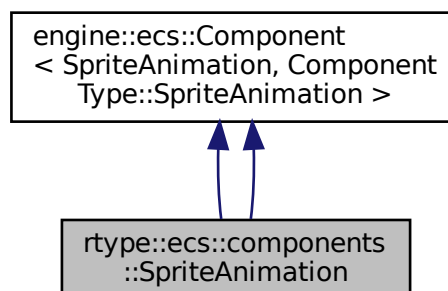
[SpriteAnimation](#) component.

```
#include <Components.hpp>
```

Inheritance diagram for `rtype::ecs::components::SpriteAnimation`:



Collaboration diagram for `rtype::ecs::components::SpriteAnimation`:



## Public Member Functions

- **SpriteAnimation** (sf::Sprite &sprite, float timePerFrame, std::vector< sf::IntRect > frames)
- **SpriteAnimation** (sf::Sprite &sprite, float timePerFrame, std::vector< sf::IntRect > frames)

## Public Attributes

- std::vector< sf::IntRect > **frames**
- std::shared\_ptr< sf::Sprite > **sprite**
- int **currentFrame** = 0
- float **timePerFrame**
- float **accumulatedTime** = 0

## Additional Inherited Members

### 3.53.1 Detailed Description

[SpriteAnimation](#) component.

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

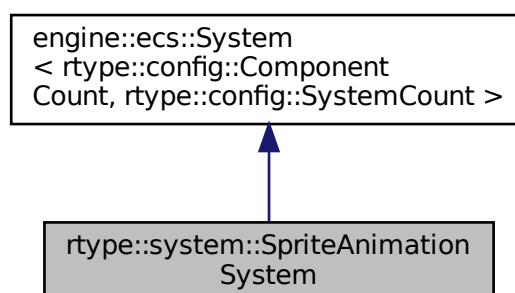
- rtype-client/include/Components.hpp

## 3.54 rtype::system::SpriteAnimationSystem Class Reference

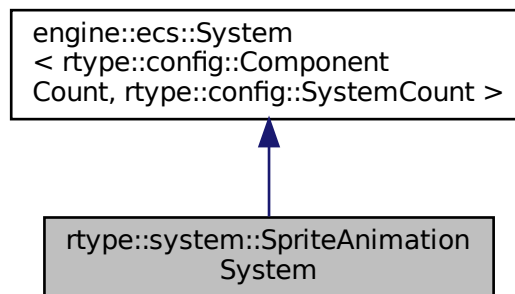
SpriteAnimation system class used to animate sprites.

```
#include <SpriteAnimationSystem.hpp>
```

Inheritance diagram for rtype::system::SpriteAnimationSystem:



Collaboration diagram for rtype::system::SpriteAnimationSystem:



## Public Member Functions

- [SpriteAnimationSystem](#) ([engine::ecs::EntityManager](#)< `rtype::config::ComponentCount`, `rtype::config::SystemCount` > &`entityManager`)  
*Construct a new Sprite Animation System object.*
- void [update](#) (const float dt)  
*Update the SpriteAnimation system.*

## Additional Inherited Members

### 3.54.1 Detailed Description

SpriteAnimation system class used to animate sprites.

### 3.54.2 Constructor & Destructor Documentation

#### 3.54.2.1 SpriteAnimationSystem()

```

rtype::system::SpriteAnimationSystem::SpriteAnimationSystem (
    engine::ecs::EntityManager< rtype::config::ComponentCount, rtype::config::SystemCount > & entityManager ) [inline]
  
```

Construct a new Sprite Animation System object.

#### Parameters

<code>entityManager</code>	
----------------------------	--

### 3.54.3 Member Function Documentation

#### 3.54.3.1 update()

```
void rtype::system::SpriteAnimationSystem::update (
    const float dt ) [inline], [virtual]
```

Update the SpriteAnimation system.

##### Parameters

<i>dt</i>	
-----------	--

Reimplemented from [engine::ecs::System< rtype::config::ComponentCount, rtype::config::SystemCount >](#).

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

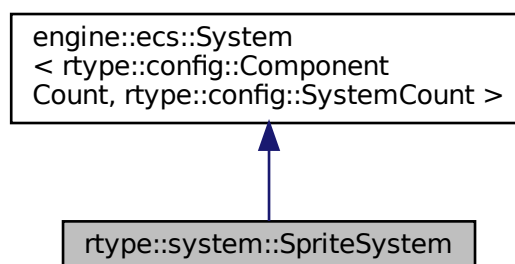
- [rtype-client/include/systems/SpriteAnimationSystem.hpp](#)

## 3.55 rtype::system::SpriteSystem Class Reference

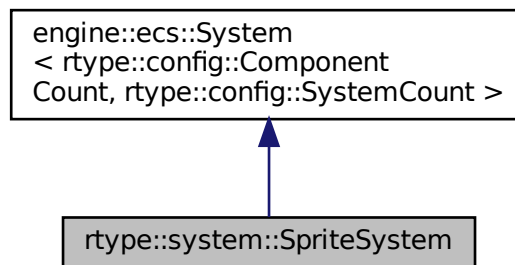
[SpriteSystem](#) class used to draw the sprites of the game.

```
#include <SpriteSystem.hpp>
```

Inheritance diagram for rtype::system::SpriteSystem:



Collaboration diagram for rtype::system::SpriteSystem:



## Public Member Functions

- [SpriteSystem](#) ([engine::ecs::EntityManager](#) < `rtype::config::ComponentCount`, `rtype::config::SystemCount` > &entityManager)  
Construct a new Sprite System object.
- void [draw](#) ([engine::graphical::GraphicalWindow](#) &window)  
Draw the sprites.

## Additional Inherited Members

### 3.55.1 Detailed Description

[SpriteSystem](#) class used to draw the sprites of the game.

### 3.55.2 Constructor & Destructor Documentation

#### 3.55.2.1 SpriteSystem()

```

rtype::system::SpriteSystem::SpriteSystem (
    engine::ecs::EntityManager< rtype::config::ComponentCount, rtype::config::SystemCount > & entityManager ) [inline]

```

Construct a new Sprite System object.

#### Parameters

<code>entityManager</code>	
----------------------------	--

### 3.55.3 Member Function Documentation

#### 3.55.3.1 draw()

```
void rtype::system::SpriteSystem::draw (  
    engine::graphical::GraphicalWindow & window ) [inline], [virtual]
```

Draw the sprites.

##### Parameters

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

Reimplemented from [engine::ecs::System< rtype::config::ComponentCount, rtype::config::SystemCount >](#).

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

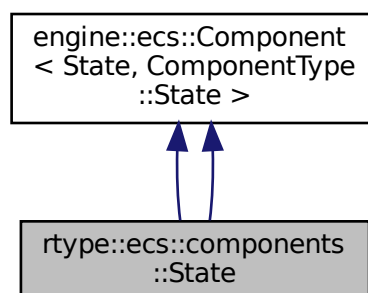
- [rtype-client/include/systems/SpriteSystem.hpp](#)

## 3.56 rtype::ecs::components::State Struct Reference

[State](#) struct which contains the state of the entity.

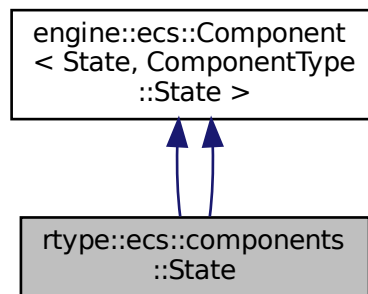
```
#include <Components.hpp>
```

Inheritance diagram for `rtype::ecs::components::State`:





Collaboration diagram for rtype::ecs::components::State:



### Public Attributes

- `bool toRemove`

### Additional Inherited Members

#### 3.56.1 Detailed Description

[State](#) struct which contains the state of the entity.

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

- `rtype-client/include/Components.hpp`

## 3.57 engine::ecs::System< ComponentCount, SystemCount > Class Template Reference

BaseSystem Interface used to store systems.

```
#include <System.hpp>
```

## Protected Member Functions

- `template<typename... Ts>`  
`void setRequirements ()`  
*Set the Requirements object.*
- `const std::vector< Entity > & getManagedEntities () const`  
*Get the Managed Entities object.*
- `virtual void onManagedEntityAdded ([[maybe_unused]] Entity entity)`  
*Add an entity to the system.*
- `virtual void onManagedEntityRemoved ([[maybe_unused]] Entity entity)`  
*Remove an entity from the system.*
- `virtual void update (const float dt)`  
*Update the system.*
- `virtual void draw (engine::graphical::GraphicalWindow &window)`  
*Draw the system.*
- `virtual void handleEvent (const Event &event)`  
*Handle an event.*

### 3.57.1 Detailed Description

```
template<std::size_t ComponentCount, std::size_t SystemCount>
class engine::ecs::System< ComponentCount, SystemCount >
```

BaseSystem Interface used to store systems.

#### Template Parameters

<i>ComponentCount</i>	
<i>SystemCount</i>	

### 3.57.2 Member Function Documentation

#### 3.57.2.1 `draw()`

```
template<std::size_t ComponentCount, std::size_t SystemCount>
virtual void engine::ecs::System< ComponentCount, SystemCount >::draw (
    engine::graphical::GraphicalWindow & window ) [inline], [protected], [virtual]
```

Draw the system.

#### Parameters

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

Reimplemented in [rtype::system::SpriteSystem](#), and [rtype::system::ScoreSystem](#).

### 3.57.2.2 getManagedEntities()

```
template<std::size_t ComponentCount, std::size_t SystemCount>
const std::vector<Entity>& engine::ecs::System< ComponentCount, SystemCount >::getManagedEntities ( ) const [inline], [protected]
```

Get the Managed Entities object.

#### Returns

const std::vector<Entity>&

### 3.57.2.3 handleEvent()

```
template<std::size_t ComponentCount, std::size_t SystemCount>
virtual void engine::ecs::System< ComponentCount, SystemCount >::handleEvent (
    const Event & event ) [inline], [protected], [virtual]
```

Handle an event.

#### Parameters

<i>event</i>	
--------------	--

Reimplemented in [rtype::system::PlayerShootSystem](#), [rtype::system::PlayerShootSystem](#), and [rtype::system::MovementSystem](#).

### 3.57.2.4 onManagedEntityAdded()

```
template<std::size_t ComponentCount, std::size_t SystemCount>
virtual void engine::ecs::System< ComponentCount, SystemCount >::onManagedEntityAdded (
    [[maybe_unused] ] Entity entity ) [inline], [protected], [virtual]
```

Add an entity to the system.

#### Parameters

<i>entity</i>	
---------------	--

### 3.57.2.5 onManagedEntityRemoved()

```
template<std::size_t ComponentCount, std::size_t SystemCount>
```

```
virtual void engine::ecs::System< ComponentCount, SystemCount >::onManagedEntityRemoved (
    [[maybe_unused]] Entity entity ) [inline], [protected], [virtual]
```

Remove an entity from the system.

#### Parameters

<i>entity</i>	
---------------	--

### 3.57.2.6 setRequirements()

```
template<std::size_t ComponentCount, std::size_t SystemCount>
template<typename... Ts>
void engine::ecs::System< ComponentCount, SystemCount >::setRequirements ( ) [inline], [protected]
```

Set the Requirements object.

#### Template Parameters

<i>Ts</i>	
-----------	--

### 3.57.2.7 update()

```
template<std::size_t ComponentCount, std::size_t SystemCount>
virtual void engine::ecs::System< ComponentCount, SystemCount >::update (
    const float dt ) [inline], [protected], [virtual]
```

Update the system.

#### Parameters

<i>dt</i>	
-----------	--

Reimplemented in [rtype::system::SpawnBonusSystem](#), [rtype::system::HealthSystem](#), [rtype::system::EnemyShootSystem](#), [rtype::system::BossHiveShootSystem](#), [rtype::system::BossAlienShootSystem](#), [rtype::system::SpriteAnimationSystem](#), [rtype::system::SpawnBonusSystem](#), [rtype::system::HealthSystem](#), [rtype::system::EnemyShootSystem](#), [rtype::system::BossHiveShootSystem](#), [rtype::system::BossAlienShootSystem](#), [rtype::system::TransformSystem](#), [rtype::system::ScoreSystem](#), [rtype::system::ParallaxSystem](#), [rtype::system::LevelSystem](#), [rtype::system::CollisionSystem](#), and [rtype::system::CleanEntitySystem](#).

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

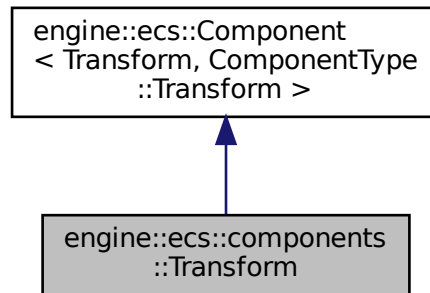
- [engine/include/ecs/System.hpp](#)

## 3.58 engine::ecs::components::Transform Struct Reference

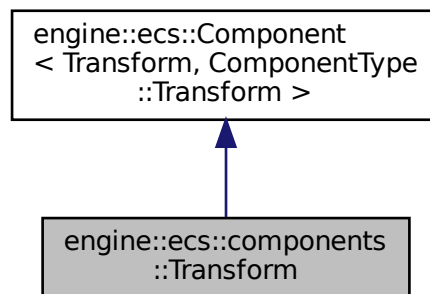
[Transform](#) component.

```
#include <Components.hpp>
```

Inheritance diagram for engine::ecs::components::Transform:



Collaboration diagram for engine::ecs::components::Transform:



### Public Member Functions

- **Transform** (float X=0.0, float Y=0.0)

### Public Attributes

- float **x**
- float **y**
- float **prevX**
- float **prevY**

## Additional Inherited Members

### 3.58.1 Detailed Description

[Transform](#) component.

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

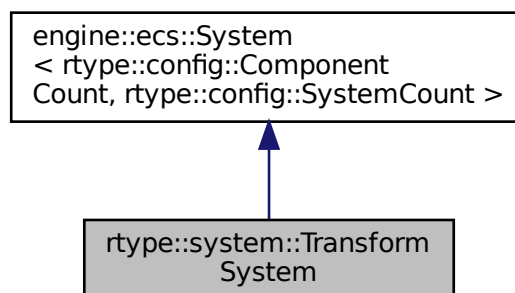
- engine/include/ecs/Components.hpp

## 3.59 rtype::system::TransformSystem Class Reference

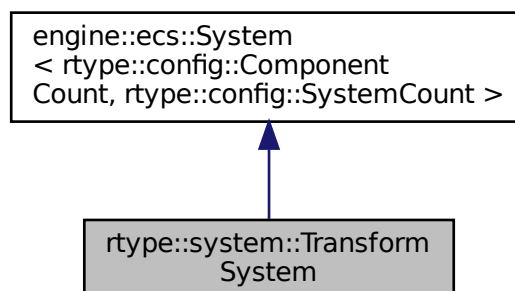
[TransformSystem](#) class used to update the position of the entities.

```
#include <TransformSystem.hpp>
```

Inheritance diagram for rtype::system::TransformSystem:



Collaboration diagram for rtype::system::TransformSystem:



## Public Member Functions

- [TransformSystem](#) ([engine::ecs::EntityManager](#) < rtype::config::ComponentCount, rtype::config::SystemCount > &entityManager)  
*Construct a new Transform System object.*
- void [update](#) (const float dt)  
*Update the Transform System object.*

## Additional Inherited Members

### 3.59.1 Detailed Description

[TransformSystem](#) class used to update the position of the entities.

### 3.59.2 Constructor & Destructor Documentation

#### 3.59.2.1 TransformSystem()

```
rtype::system::TransformSystem::TransformSystem (
    engine::ecs::EntityManager < rtype::config::ComponentCount, rtype::config::SystemCount > & entityManager ) [inline]
```

Construct a new Transform System object.

#### Parameters

<i>entityManager</i>	
----------------------	--

### 3.59.3 Member Function Documentation

#### 3.59.3.1 update()

```
void rtype::system::TransformSystem::update (
    const float dt ) [inline], [virtual]
```

Update the Transform System object.

#### Parameters

<i>dt</i>	
-----------	--

Reimplemented from [engine::ecs::System< rtype::config::ComponentCount, rtype::config::SystemCount >](#).

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

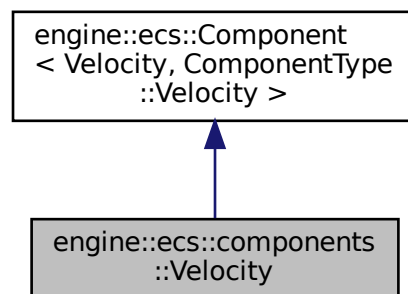
- `game/include/systems/TransformSystem.hpp`

### 3.60 engine::ecs::components::Velocity Struct Reference

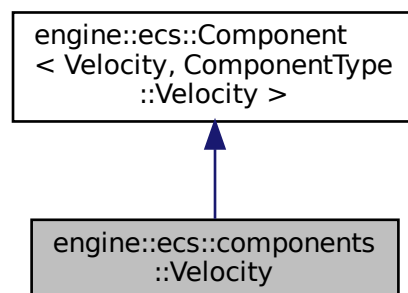
[Velocity](#) component.

```
#include <Components.hpp>
```

Inheritance diagram for engine::ecs::components::Velocity:



Collaboration diagram for engine::ecs::components::Velocity:



#### Public Member Functions

- **Velocity** (float X=0.0, float Y=0.0)



## Public Attributes

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

## Additional Inherited Members

### 3.60.1 Detailed Description

[Velocity](#) component.

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

- engine/include/ecs/Components.hpp



# Index

~ClientGame  
  rtype::ClientGame, [25](#)

add  
  engine::ecs::ComponentContainer< T, ComponentCount, SystemCount >, [31](#)

addComponent  
  engine::ecs::EntityManager< ComponentCount, SystemCount >, [48](#)

BossAlienShootSystem  
  rtype::system::BossAlienShootSystem, [16](#), [17](#)

BossHiveShootSystem  
  rtype::system::BossHiveShootSystem, [20](#)

CleanEntitySystem  
  rtype::system::CleanEntitySystem, [23](#)

client::clientInstance, [26](#)  
  clientInstance, [27](#)

ClientGame  
  rtype::ClientGame, [25](#)

clientInstance  
  client::clientInstance, [27](#)

CollisionSystem  
  rtype::system::CollisionSystem, [28](#)

create  
  engine::ecs::EntityContainer< ComponentCount, SystemCount >, [44](#)

createAsteroid  
  rtype::EntityTemplate, [58](#), [60](#)

createBossAlien  
  rtype::EntityTemplate, [60](#), [61](#)

createBossAlienBullet  
  rtype::EntityTemplate, [61](#)

createBossHive  
  rtype::EntityTemplate, [62](#)

createBossHiveBullet  
  rtype::EntityTemplate, [63](#)

createBossLevel  
  rtype::LevelManager, [87](#)

createBottomBoundary  
  rtype::EntityTemplate, [64](#)

createEnemyBeetle  
  rtype::EntityTemplate, [64](#), [66](#)

createEnemyBullet  
  rtype::EntityTemplate, [66](#)

createEnemyCrab  
  rtype::EntityTemplate, [67](#)

createEntity  
  engine::ecs::EntityManager< ComponentCount, SystemCount >, [49](#)

createHealthBonus  
  rtype::EntityTemplate, [68](#)

createLevel  
  rtype::LevelManager, [88](#)

createParallax  
  rtype::EntityTemplate, [68](#), [69](#)

createPlayer  
  rtype::EntityTemplate, [69](#), [70](#)

createPlayerBullet  
  rtype::EntityTemplate, [70](#), [71](#)

createSpecialPlayerBullet  
  rtype::EntityTemplate, [71](#)

createSpecialShootBonus  
  rtype::EntityTemplate, [72](#)

createSystem  
  engine::ecs::EntityManager< ComponentCount, SystemCount >, [49](#)

createTopBoundary  
  rtype::EntityTemplate, [73](#)

dispatchEvent  
  engine::ecs::EntityManager< ComponentCount, SystemCount >, [49](#)

displayScore  
  rtype::ScoreManager, [108](#), [109](#)

draw  
  engine::ecs::System< ComponentCount, SystemCount >, [128](#)  
  rtype::system::ScoreSystem, [112](#)  
  rtype::system::SpriteSystem, [126](#)

drawSystems  
  engine::ecs::EntityManager< ComponentCount, SystemCount >, [50](#)

EnemyShootSystem  
  rtype::system::EnemyShootSystem, [41](#)

engine::ecs::BaseComponentContainer, [13](#)  
  reserve, [13](#)  
  tryRemove, [14](#)

engine::ecs::Component< T, Type >, [29](#)

engine::ecs::ComponentContainer< T, ComponentCount, SystemCount >, [30](#)  
  add, [31](#)  
  get, [32](#)  
  getOwner, [32](#)  
  remove, [33](#)  
  reserve, [33](#)  
  tryRemove, [33](#)

engine::ecs::components::Hitbox, 83  
 engine::ecs::components::Transform, 131  
 engine::ecs::components::Velocity, 134  
 engine::ecs::EntityContainer< ComponentCount, SystemCount >, 44  
     create, 44  
     getBitset, 45  
     getEntityToBitset, 45  
     getEntityToComponent, 45  
     getEntityToManagedEntity, 46  
     remove, 46  
     reserve, 46  
 engine::ecs::EntityManager< ComponentCount, SystemCount >, 47  
     addComponent, 48  
     createEntity, 49  
     createSystem, 49  
     dispatchEvent, 49  
     drawSystems, 50  
     getComponent, 50  
     getComponents, 51  
     getOwner, 53  
     hasComponent, 53  
     hasComponents, 54  
     registerComponent, 54  
     removeComponent, 55  
     removeEntity, 55  
     reserve, 55  
     updateSystems, 56  
 engine::ecs::System< ComponentCount, SystemCount >, 127  
     draw, 128  
     getManagedEntities, 129  
     handleEvent, 129  
     onManagedEntityAdded, 129  
     onManagedEntityRemoved, 129  
     setRequirements, 130  
     update, 130  
 engine::Event, 74  
 EntityTemplate  
     rtype::EntityTemplate, 58  
 Game  
     rtype::Game, 76  
 get  
     engine::ecs::ComponentContainer< T, ComponentCount, SystemCount >, 32  
 getBitset  
     engine::ecs::EntityContainer< ComponentCount, SystemCount >, 45  
 getComponent  
     engine::ecs::EntityManager< ComponentCount, SystemCount >, 50  
 getComponents  
     engine::ecs::EntityManager< ComponentCount, SystemCount >, 51  
 getEntityManager  
     rtype::Game, 77  
 getEntityToBitset  
     engine::ecs::EntityContainer< ComponentCount, SystemCount >, 45  
 getEntityToComponent  
     engine::ecs::EntityContainer< ComponentCount, SystemCount >, 45  
 getEntityToManagedEntity  
     engine::ecs::EntityContainer< ComponentCount, SystemCount >, 46  
 getFps  
     rtype::ClientGame, 25  
     rtype::Game, 77  
     rtype::ServerGame, 114  
 getInstance  
     rtype::AssetManager, 9  
 getIp  
     rtype::ClientGame, 25  
 getLevel  
     rtype::LevelManager, 88  
 getManagedEntities  
     engine::ecs::System< ComponentCount, SystemCount >, 129  
 getOwner  
     engine::ecs::ComponentContainer< T, ComponentCount, SystemCount >, 32  
     engine::ecs::EntityManager< ComponentCount, SystemCount >, 53  
 getPlayerId  
     rtype::ClientGame, 26  
 getScore  
     rtype::ScoreManager, 109  
 getSound  
     rtype::AssetManager, 10  
 getSoundBuffer  
     rtype::AssetManager, 10  
 getTexture  
     rtype::AssetManager, 10  
 handleEvent  
     engine::ecs::System< ComponentCount, SystemCount >, 129  
     rtype::Game, 77  
     rtype::ServerGame, 114  
     rtype::system::MovementSystem, 94  
     rtype::system::PlayerShootSystem, 106  
 hasComponent  
     engine::ecs::EntityManager< ComponentCount, SystemCount >, 53  
 hasComponents  
     engine::ecs::EntityManager< ComponentCount, SystemCount >, 54  
 HealthSystem  
     rtype::system::HealthSystem, 82  
 id  
     rtype::NetworkEvent, 95  
 isOver  
     rtype::ClientGame, 26  
     rtype::Game, 78  
     rtype::ServerGame, 114

key  
  rtype::NetworkEvent, 95

LevelManager  
  rtype::LevelManager, 86, 87

LevelSystem  
  rtype::system::LevelSystem, 90

MovementSystem  
  rtype::system::MovementSystem, 94

objectType  
  rtype::NetworkEvent, 95

onManagedEntityAdded  
  engine::ecs::System< ComponentCount, System-  
  Count >, 129

onManagedEntityRemoved  
  engine::ecs::System< ComponentCount, System-  
  Count >, 129

ParallaxSystem  
  rtype::system::ParallaxSystem, 98

PlayerShootSystem  
  rtype::system::PlayerShootSystem, 105, 106

playSound  
  rtype::AssetManager, 11

pos  
  rtype::NetworkEvent, 95

registerComponent  
  engine::ecs::EntityManager< ComponentCount,  
  SystemCount >, 54

remove  
  engine::ecs::ComponentContainer< T, Compon-  
  nentCount, SystemCount >, 33  
  engine::ecs::EntityContainer< ComponentCount,  
  SystemCount >, 46

removeComponent  
  engine::ecs::EntityManager< ComponentCount,  
  SystemCount >, 55

removeEntity  
  engine::ecs::EntityManager< ComponentCount,  
  SystemCount >, 55

reserve  
  engine::ecs::BaseComponentContainer, 13  
  engine::ecs::ComponentContainer< T, Compon-  
  nentCount, SystemCount >, 33  
  engine::ecs::EntityContainer< ComponentCount,  
  SystemCount >, 46  
  engine::ecs::EntityManager< ComponentCount,  
  SystemCount >, 55

rtype::AssetManager, 9  
  getInstance, 9  
  getSound, 10  
  getSoundBuffer, 10  
  getTexture, 10  
  playSound, 11

rtype::ClientGame, 24  
  ~ClientGame, 25

ClientGame, 25  
  getFps, 25  
  getIp, 25  
  getPlayerID, 26  
  isOver, 26

rtype::debug::Debugger, 35

rtype::ecs::components::AsteroidId, 12

rtype::ecs::components::BossAlien, 14

rtype::ecs::components::BossHive, 18

rtype::ecs::components::DebugTag, 35

rtype::ecs::components::Enemy, 36

rtype::ecs::components::EnemyBeetleId, 38

rtype::ecs::components::EnemyCrabId, 39

rtype::ecs::components::Ennemild, 43

rtype::ecs::components::Health, 78

rtype::ecs::components::HealthBonusId, 80

rtype::ecs::components::LifeTime, 91

rtype::ecs::components::MoveComponent, 92

rtype::ecs::components::Parallax, 96

rtype::ecs::components::ParrallaxId, 99

rtype::ecs::components::ParticleSpawner, 100

rtype::ecs::components::PlayerId, 102

rtype::ecs::components::PlayerShoot, 103

rtype::ecs::components::ShootBonusId, 116

rtype::ecs::components::Sprite, 119

rtype::ecs::components::SpriteAnimation, 121

rtype::ecs::components::State, 126

rtype::EntityTemplate, 56  
  createAsteroid, 58, 60  
  createBossAlien, 60, 61  
  createBossAlienBullet, 61  
  createBossHive, 62  
  createBossHiveBullet, 63  
  createBottomBoundary, 64  
  createEnemyBeetle, 64, 66  
  createEnemyBullet, 66  
  createEnemyCrab, 67  
  createHealthBonus, 68  
  createParallax, 68, 69  
  createPlayer, 69, 70  
  createPlayerBullet, 70, 71  
  createSpecialPlayerBullet, 71  
  createSpecialShootBonus, 72  
  createTopBoundary, 73  
  EntityTemplate, 58

rtype::event::DeathEvent, 34

rtype::event::InputEvent, 85

rtype::Game, 74  
  Game, 76  
  getEntityManager, 77  
  getFps, 77  
  handleEvent, 77  
  isOver, 78

rtype::LevelManager, 86  
  createBossLevel, 87  
  createLevel, 88  
  getLevel, 88  
  LevelManager, 86, 87

- rtype::NetworkEvent, 95
  - id, 95
  - key, 95
  - objectType, 95
  - pos, 95
  - type, 96
- rtype::QueueManager, 107
- rtype::ScoreManager, 107
  - displayScore, 108, 109
  - getScore, 109
  - ScoreManager, 108
  - updateScore, 109, 110
- rtype::ServerGame, 112
  - getFps, 114
  - handleEvent, 114
  - isOver, 114
  - ServerGame, 114
- rtype::system::BossAlienShootSystem, 15
  - BossAlienShootSystem, 16, 17
  - update, 17
- rtype::system::BossHiveShootSystem, 19
  - BossHiveShootSystem, 20
  - update, 21
- rtype::system::CleanEntitySystem, 21
  - CleanEntitySystem, 23
  - update, 23
- rtype::system::CollisionSystem, 27
  - CollisionSystem, 28
  - update, 29
- rtype::system::EnemyShootSystem, 40
  - EnemyShootSystem, 41
  - update, 42
- rtype::system::HealthSystem, 81
  - HealthSystem, 82
  - update, 83
- rtype::system::LevelSystem, 89
  - LevelSystem, 90
  - update, 90
- rtype::system::MovementSystem, 93
  - handleEvent, 94
  - MovementSystem, 94
- rtype::system::ParallaxSystem, 97
  - ParallaxSystem, 98
  - update, 99
- rtype::system::PlayerShootSystem, 104
  - handleEvent, 106
  - PlayerShootSystem, 105, 106
- rtype::system::ScoreSystem, 110
  - draw, 112
  - ScoreSystem, 111
  - update, 112
- rtype::system::SpawnBonusSystem, 117
  - SpawnBonusSystem, 118
  - update, 119
- rtype::system::SpriteAnimationSystem, 122
  - SpriteAnimationSystem, 123
  - update, 124
- rtype::system::SpriteSystem, 124
  - draw, 126
  - SpriteSystem, 125
- rtype::system::TransformSystem, 132
  - TransformSystem, 133
  - update, 133
- ScoreManager
  - rtype::ScoreManager, 108
- ScoreSystem
  - rtype::system::ScoreSystem, 111
- server::serverInstance, 115
- ServerGame
  - rtype::ServerGame, 114
- setRequirements
  - engine::ecs::System< ComponentCount, SystemCount >, 130
- SpawnBonusSystem
  - rtype::system::SpawnBonusSystem, 118
- SpriteAnimationSystem
  - rtype::system::SpriteAnimationSystem, 123
- SpriteSystem
  - rtype::system::SpriteSystem, 125
- TransformSystem
  - rtype::system::TransformSystem, 133
- tryRemove
  - engine::ecs::BaseComponentContainer, 14
  - engine::ecs::ComponentContainer< T, ComponentCount, SystemCount >, 33
- type
  - rtype::NetworkEvent, 96
- update
  - engine::ecs::System< ComponentCount, SystemCount >, 130
  - rtype::system::BossAlienShootSystem, 17
  - rtype::system::BossHiveShootSystem, 21
  - rtype::system::CleanEntitySystem, 23
  - rtype::system::CollisionSystem, 29
  - rtype::system::EnemyShootSystem, 42
  - rtype::system::HealthSystem, 83
  - rtype::system::LevelSystem, 90
  - rtype::system::ParallaxSystem, 99
  - rtype::system::ScoreSystem, 112
  - rtype::system::SpawnBonusSystem, 119
  - rtype::system::SpriteAnimationSystem, 124
  - rtype::system::TransformSystem, 133
- updateScore
  - rtype::ScoreManager, 109, 110
- updateSystems
  - engine::ecs::EntityManager< ComponentCount, SystemCount >, 56