

# Reference Manual

Generated by Doxygen 1.6.1

Fri Nov 28 23:26:52 2014



# Contents

<b>1</b>	<b>Class Index</b>	<b>1</b>
1.1	Class Hierarchy . . . . .	1
<b>2</b>	<b>Class Index</b>	<b>3</b>
2.1	Class List . . . . .	3
<b>3</b>	<b>Class Documentation</b>	<b>5</b>
3.1	Armor Class Reference . . . . .	5
3.1.1	Detailed Description . . . . .	5
3.1.2	Member Function Documentation . . . . .	5
3.1.2.1	use . . . . .	5
3.2	Attributes Class Reference . . . . .	7
3.2.1	Detailed Description . . . . .	7
3.2.2	Member Function Documentation . . . . .	8
3.2.2.1	getAgility . . . . .	8
3.2.2.2	getArmor . . . . .	8
3.2.2.3	getDamage . . . . .	8
3.2.2.4	getHealthPoints . . . . .	8
3.2.2.5	getStrength . . . . .	8
3.2.2.6	getVitality . . . . .	8
3.2.2.7	setAgility . . . . .	9
3.2.2.8	setArmor . . . . .	9
3.2.2.9	setDamage . . . . .	9
3.2.2.10	setHealthPoints . . . . .	9
3.2.2.11	setStrength . . . . .	9
3.2.2.12	setVitality . . . . .	9
3.3	Entity Class Reference . . . . .	10
3.3.1	Detailed Description . . . . .	10
3.3.2	Member Function Documentation . . . . .	11

3.3.2.1	attack . . . . .	11
3.3.2.2	getAttributes . . . . .	11
3.3.2.3	getInventory . . . . .	11
3.3.2.4	getLoot . . . . .	11
3.3.2.5	getName . . . . .	11
3.3.2.6	hasLoot . . . . .	12
3.3.2.7	isAlive . . . . .	12
3.3.2.8	setInventory . . . . .	12
3.3.2.9	setName . . . . .	12
3.4	EntityBat Class Reference . . . . .	13
3.4.1	Member Function Documentation . . . . .	13
3.4.1.1	attack . . . . .	13
3.4.1.2	getLoot . . . . .	13
3.4.1.3	hasLoot . . . . .	14
3.4.1.4	isAlive . . . . .	14
3.5	EntityContainer Class Reference . . . . .	15
3.5.1	Detailed Description . . . . .	15
3.5.2	Constructor & Destructor Documentation . . . . .	15
3.5.2.1	EntityContainer . . . . .	15
3.5.3	Member Function Documentation . . . . .	15
3.5.3.1	getLoot . . . . .	15
3.5.3.2	hasLoot . . . . .	16
3.6	EntityDragon Class Reference . . . . .	17
3.6.1	Detailed Description . . . . .	17
3.6.2	Member Function Documentation . . . . .	17
3.6.2.1	attack . . . . .	17
3.6.2.2	getLoot . . . . .	18
3.6.2.3	hasLoot . . . . .	18
3.6.2.4	isAlive . . . . .	18
3.7	EntityFactory Class Reference . . . . .	19
3.7.1	Member Function Documentation . . . . .	19
3.7.1.1	getEnemy . . . . .	19
3.7.1.2	getInstance . . . . .	19
3.7.1.3	makeInventory . . . . .	19
3.8	EntityPlayer Class Reference . . . . .	21
3.8.1	Detailed Description . . . . .	21

3.8.2	Constructor & Destructor Documentation	22
3.8.2.1	EntityPlayer	22
3.8.2.2	~EntityPlayer	22
3.8.3	Member Function Documentation	22
3.8.3.1	attack	22
3.8.3.2	equipArmor	22
3.8.3.3	equipWeapon	22
3.8.3.4	getLoot	22
3.8.3.5	hasLoot	22
3.8.3.6	isAlive	23
3.8.3.7	usePotion	23
3.9	EntitySkeleton Class Reference	24
3.9.1	Detailed Description	24
3.9.2	Member Function Documentation	24
3.9.2.1	attack	24
3.9.2.2	getLoot	24
3.9.2.3	hasLoot	25
3.9.2.4	isAlive	25
3.10	EntitySpider Class Reference	26
3.10.1	Detailed Description	26
3.10.2	Member Function Documentation	26
3.10.2.1	attack	26
3.10.2.2	getLoot	26
3.10.2.3	hasLoot	27
3.10.2.4	isAlive	27
3.11	EntityTester Class Reference	28
3.12	EntityTroll Class Reference	29
3.12.1	Detailed Description	29
3.12.2	Member Function Documentation	29
3.12.2.1	attack	29
3.12.2.2	getLoot	29
3.12.2.3	hasLoot	30
3.12.2.4	isAlive	30
3.13	Game Class Reference	31
3.13.1	Detailed Description	32
3.13.2	Member Function Documentation	32

3.13.2.1	<a href="#">getDebug</a>	32
3.13.2.2	<a href="#">getDifficulty</a>	32
3.13.2.3	<a href="#">getOption</a>	32
3.13.2.4	<a href="#">log</a>	33
3.13.2.5	<a href="#">openInventory</a>	33
3.13.2.6	<a href="#">print</a>	33
3.13.2.7	<a href="#">setDebug</a>	33
3.13.2.8	<a href="#">setDifficulty</a>	33
3.14	<a href="#">GameTest Class Reference</a>	34
3.15	<a href="#">Item Class Reference</a>	35
3.15.1	<a href="#">Detailed Description</a>	36
3.15.2	<a href="#">Member Function Documentation</a>	36
3.15.2.1	<a href="#">getDescription</a>	36
3.15.2.2	<a href="#">getName</a>	36
3.15.2.3	<a href="#">isArmor</a>	36
3.15.2.4	<a href="#">isPotion</a>	36
3.15.2.5	<a href="#">isWeapon</a>	36
3.15.2.6	<a href="#">use</a>	36
3.16	<a href="#">ItemFactory Class Reference</a>	37
3.16.1	<a href="#">Detailed Description</a>	37
3.16.2	<a href="#">Member Function Documentation</a>	37
3.16.2.1	<a href="#">getInstance</a>	37
3.16.2.2	<a href="#">getItem</a>	37
3.17	<a href="#">ItemTest Class Reference</a>	38
3.18	<a href="#">Map Class Reference</a>	39
3.18.1	<a href="#">Detailed Description</a>	39
3.18.2	<a href="#">Constructor &amp; Destructor Documentation</a>	39
3.18.2.1	<a href="#">Map</a>	39
3.18.3	<a href="#">Member Function Documentation</a>	40
3.18.3.1	<a href="#">displayDescription</a>	40
3.18.3.2	<a href="#">generate</a>	40
3.18.3.3	<a href="#">getCurRoom</a>	40
3.18.3.4	<a href="#">movePlayer</a>	40
3.19	<a href="#">MapTest Class Reference</a>	41
3.20	<a href="#">Potion Class Reference</a>	42
3.20.1	<a href="#">Detailed Description</a>	42

3.20.2	Member Function Documentation	42
3.20.2.1	use	42
3.21	Room Class Reference	43
3.21.1	Detailed Description	43
3.21.2	Member Function Documentation	44
3.21.2.1	generate	44
3.21.2.2	getDescription	44
3.22	RoomArmory Class Reference	45
3.22.1	Detailed Description	45
3.22.2	Member Function Documentation	45
3.22.2.1	generate	45
3.22.2.2	getDescription	45
3.22.2.3	isEmpty	46
3.23	RoomBoss Class Reference	47
3.23.1	Detailed Description	47
3.23.2	Member Function Documentation	47
3.23.2.1	generate	47
3.23.2.2	getDescription	47
3.23.2.3	isEmpty	48
3.24	RoomHall Class Reference	49
3.24.1	Detailed Description	49
3.24.2	Member Function Documentation	49
3.24.2.1	generate	49
3.24.2.2	getDescription	49
3.24.2.3	isEmpty	50
3.25	RoomJail Class Reference	51
3.25.1	Detailed Description	51
3.25.2	Member Function Documentation	51
3.25.2.1	generate	51
3.25.2.2	getDescription	51
3.25.2.3	isEmpty	52
3.26	RoomStandard Class Reference	53
3.26.1	Detailed Description	53
3.26.2	Member Function Documentation	53
3.26.2.1	generate	53
3.26.2.2	getDescription	53

3.26.2.3	isEmpty	54
3.27	Weapon Class Reference	55
3.27.1	Detailed Description	55
3.27.2	Member Function Documentation	55
3.27.2.1	use	55



# Chapter 1

## Class Index

### 1.1 Class Hierarchy

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

Attributes . . . . .	7
Entity . . . . .	10
EntityBat . . . . .	13
EntityContainer . . . . .	15
EntityDragon . . . . .	17
EntityPlayer . . . . .	21
EntitySkeleton . . . . .	24
EntitySpider . . . . .	26
EntityTroll . . . . .	29
EntityFactory . . . . .	19
EntityTester . . . . .	28
Game . . . . .	31
GameTest . . . . .	34
Item . . . . .	35
Armor . . . . .	5
Potion . . . . .	42
Weapon . . . . .	55
ItemFactory . . . . .	37
ItemTest . . . . .	38
Map . . . . .	39
MapTest . . . . .	41
Room . . . . .	43
RoomArmory . . . . .	45
RoomBoss . . . . .	47
RoomHall . . . . .	49
RoomJail . . . . .	51
RoomStandard . . . . .	53



# Chapter 2

## Class Index

### 2.1 Class List

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

<a href="#">Armor</a> ( <a href="#">Armor</a> Class to be generated by the <a href="#">ItemFactory</a> ) . . . . .	5
<a href="#">Attributes</a> (Class <a href="#">Attribute</a> ) . . . . .	7
<a href="#">Entity</a> ( <a href="#">Entity</a> class ) . . . . .	10
<a href="#">EntityBat</a> . . . . .	13
<a href="#">EntityContainer</a> (CLASS SCRAPPED DUE TO TIME CONSTRAINTS ) . . . . .	15
<a href="#">EntityDragon</a> (Class <a href="#">EntityDragon</a> ) . . . . .	17
<a href="#">EntityFactory</a> . . . . .	19
<a href="#">EntityPlayer</a> ( <a href="#">Entity</a> class ) . . . . .	21
<a href="#">EntitySkeleton</a> ( <a href="#">EntitySkeleton</a> ) . . . . .	24
<a href="#">EntitySpider</a> (Class <a href="#">EntitySpider</a> ) . . . . .	26
<a href="#">EntityTester</a> . . . . .	28
<a href="#">EntityTroll</a> (Class <a href="#">EntityTroll</a> ) . . . . .	29
<a href="#">Game</a> . . . . .	31
<a href="#">GameTest</a> . . . . .	34
<a href="#">Item</a> ( <a href="#">Item</a> Class pure virtual interface for item classes ) . . . . .	35
<a href="#">ItemFactory</a> ( <a href="#">ItemFactory</a> class for creation of weapons, armor, and potions ) . . . . .	37
<a href="#">ItemTest</a> . . . . .	38
<a href="#">Map</a> ( <a href="#">Map</a> Class ) . . . . .	39
<a href="#">MapTest</a> . . . . .	41
<a href="#">Potion</a> ( <a href="#">Weapon</a> Class to be generated by the <a href="#">ItemFactory</a> ) . . . . .	42
<a href="#">Room</a> (Class <a href="#">Room</a> ) . . . . .	43
<a href="#">RoomArmory</a> (Class <a href="#">RoomArmory</a> ) . . . . .	45
<a href="#">RoomBoss</a> (Class <a href="#">RoomBoss</a> ) . . . . .	47
<a href="#">RoomHall</a> (Class <a href="#">RoomHall</a> ) . . . . .	49
<a href="#">RoomJail</a> (Class <a href="#">RoomJail</a> ) . . . . .	51
<a href="#">RoomStandard</a> (Class <a href="#">RoomStandard</a> ) . . . . .	53
<a href="#">Weapon</a> ( <a href="#">Weapon</a> Class to be generated by the <a href="#">ItemFactory</a> ) . . . . .	55



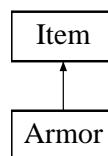
## Chapter 3

# Class Documentation

### 3.1 Armor Class Reference

[Armor](#) Class to be generated by the [ItemFactory](#).

`#include <Armor.h>`Inheritance diagram for Armor::



#### Public Member Functions

- `bool` [use](#) ()

#### 3.1.1 Detailed Description

[Armor](#) Class to be generated by the [ItemFactory](#).

#### 3.1.2 Member Function Documentation

##### 3.1.2.1 `bool Armor::use () [virtual]`

the armor

Detailed Description

#### Returns:

`bool` if the item was successfully used

Implements [Item](#).

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

- [Armor.h](#)
- [Armor.cc](#)

## 3.2 Attributes Class Reference

class Attribute

```
#include <Attributes.h>
```

### Public Member Functions

- int const [getStrength](#) ()  
*Gets the current strength value.*
- int const [getVitality](#) ()  
*Gets the current vitality value.*
- int const [getAgility](#) ()
- int const [getHealthPoints](#) ()  
*Gets the current health points value.*
- int const [getArmor](#) ()  
*Gets the current armor value.*
- int const [getDamage](#) ()  
*Gets the current damage value.*
- void [setStrength](#) (int x)  
*Sets the strength value.*
- void [setVitality](#) (int x)  
*Sets the vitality value.*
- void [setAgility](#) (int x)  
*Sets the Agility value.*
- void [setHealthPoints](#) (int x)  
*Sets the Health Points value.*
- void [setArmor](#) (int x)  
*Sets the [Armor](#) value.*
- void [setDamage](#) (int x)  
*Sets the Damage value.*

### 3.2.1 Detailed Description

class Attribute

## 3.2.2 Member Function Documentation

### 3.2.2.1 `int const Attributes::getAgility () [inline]`

Gets the current agility value

**Returns:**

Returns the current agility value

### 3.2.2.2 `int const Attributes::getArmor () [inline]`

Gets the current armor value.

**Returns:**

Returns the current armor value

### 3.2.2.3 `int const Attributes::getDamage () [inline]`

Gets the current damage value.

**Returns:**

Returns the current damage value

### 3.2.2.4 `int const Attributes::getHealthPoints () [inline]`

Gets the current health points value.

**Returns:**

Returns the current health points value

### 3.2.2.5 `int const Attributes::getStrength () [inline]`

Gets the current strength value.

**Returns:**

Returns the current strength value

### 3.2.2.6 `int const Attributes::getVitality () [inline]`

Gets the current vitality value.

**Returns:**

Returns the current vitality value



**3.2.2.7 void Attributes::setAgility (int *x*) [inline]**

Sets the Agility value.

**Parameters:**

← *Agility* value to be set

**3.2.2.8 void Attributes::setArmor (int *x*) [inline]**

Sets the *Armor* value.

**Parameters:**

← *Armor* value to be set

**3.2.2.9 void Attributes::setDamage (int *x*) [inline]**

Sets the Damage value.

**Parameters:**

← *Damage* value to be set

**3.2.2.10 void Attributes::setHealthPoints (int *x*) [inline]**

Sets the Health Points value.

**Parameters:**

← *Health* Points value to be set

**3.2.2.11 void Attributes::setStrength (int *x*) [inline]**

Sets the strength value.

**Parameters:**

← *Strength* value to be set

**3.2.2.12 void Attributes::setVitality (int *x*) [inline]**

Sets the vitality value.

**Parameters:**

← *vitality* value to be set

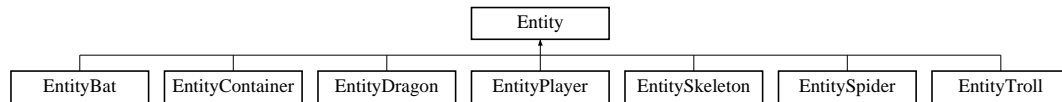
The documentation for this class was generated from the following file:

- Attributes.h

### 3.3 Entity Class Reference

[Entity](#) class.

#include <Entity.h>Inheritance diagram for Entity::



#### Public Member Functions

- virtual bool [attack](#) ([Entity](#) &entity)=0  
*Attacks the given entity.*
- virtual bool [isAlive](#) ()=0  
*Returns true if the entity is alive, false if dead.*
- virtual void [getLoot](#) ([Entity](#) &entity)=0  
*Updates the entity so that it can make decisions per turn.*
- virtual bool [hasLoot](#) ()=0  
*Returns true if the entity has items in their inventory, false otherwise.*
- std::vector< [Item](#) \* > & [getInventory](#) ()  
*Returns the entity's inventory of items.*
- [Attributes](#) & [getAttributes](#) ()  
*Returns the entity's [Attributes](#) object.*
- void [setAttributes](#) ([Attributes](#) x)
- void [setInventory](#) (std::vector< [Item](#) \* > x)  
*Sets the attribute object to another attribute object.*
- void [setName](#) (std::string &y)  
*Sets the name of the [Entity](#).*
- std::string [getName](#) ()  
*Gets the name of the [Entity](#).*

#### 3.3.1 Detailed Description

[Entity](#) class.

## 3.3.2 Member Function Documentation

### 3.3.2.1 virtual bool Entity::attack (Entity & *entity*) [pure virtual]

Attacks the given entity.

**Parameters:**

← *entity,a* reference to the entity that will be attacked

### 3.3.2.2 Attributes& Entity::getAttributes () [inline]

Returns the entity's [Attributes](#) object.

**Returns:**

Returns a reference the entity's [Attributes](#) object

### 3.3.2.3 std::vector<Item\*>& Entity::getInventory () [inline]

Returns the entity's inventory of items.

**Returns:**

Returns a vector of items that the entity owns

### 3.3.2.4 virtual void Entity::getLoot (Entity & *entity*) [pure virtual]

Updates the entity so that it can make decisions per turn.

**Returns:**

Returns void Gives the loot from the entity to the given entity (transfers inventory to the referenced entity)

**Parameters:**

← *entity,transfers* the inventory from the given entity to the current entity

**Returns:**

Returns void

### 3.3.2.5 std::string Entity::getName () [inline]

Gets the name of the [Entity](#).

**Returns:**

Name of the [Entity](#)

**3.3.2.6 virtual bool Entity::hasLoot () [pure virtual]**

Returns true if the entity has items in their inventory, false otherwise.

**Returns:**

Returns void

Implemented in [EntityBat](#), [EntityContainer](#), [EntityDragon](#), [EntityPlayer](#), [EntitySkeleton](#), [EntitySpider](#), and [EntityTroll](#).

**3.3.2.7 virtual bool Entity::isAlive () [pure virtual]**

Returns true if the entity is alive, false if dead.

**Returns:**

Returns true if the entity is alive, false if dead

Implemented in [EntityBat](#), [EntityDragon](#), [EntityPlayer](#), [EntitySkeleton](#), [EntitySpider](#), and [EntityTroll](#).

**3.3.2.8 void Entity::setInventory (std::vector< Item \* > x) [inline]**

Sets the attribute abject to another attribute object.

**Parameters:**

← *Attributes* to set

**3.3.2.9 void Entity::setName (std::string & y) [inline]**

Sets the name of the [Entity](#).

**Parameters:**

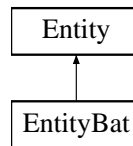
← *Name* to set to

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

- Entity.h

## 3.4 EntityBat Class Reference

Inheritance diagram for EntityBat::



### Public Member Functions

- bool [attack](#) ([Entity](#) &entity)  
*Function that calls an attack on the given Bat.*
- bool [isAlive](#) ()  
*Function that checks to see if the Bat is alive, returns a boolean value to determine such.*
- void [update](#) ()  
*Function that updates the Bat to make its next decision \*UNUSED\*.*
- void [getLoot](#) ([Entity](#) &entity)  
*Function that gives the loot to the selected Bat.*
- bool [hasLoot](#) ()  
*Function that checks to see if the current Bat has loot or not.*

### 3.4.1 Member Function Documentation

#### 3.4.1.1 bool EntityBat::attack (Entity & entity)

Function that calls an attack on the given Bat.

**Parameters:**

← [Entity](#) &entity

#### 3.4.1.2 void EntityBat::getLoot (Entity & entity)

Function that gives the loot to the selected Bat.

**Parameters:**

← [Entity](#) &entity

#### 3.4.1.3 `bool EntityBat::hasLoot ()` `[virtual]`

Function that checks to see if the current Bat has loot or not.

**Returns:**

boolean if bat has loot yes(true), no(false)

Implements [Entity](#).

#### 3.4.1.4 `bool EntityBat::isAlive ()` `[virtual]`

Function that checks to see if the Bat is alive, returns a boolean value to determine such.

**Returns:**

boolean returns if the bat is alive(true) or dead(false)

Implements [Entity](#).

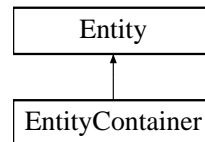
The documentation for this class was generated from the following files:

- EntityBat.h
- EntityBat.cc

## 3.5 EntityContainer Class Reference

CLASS SCRAPPED DUE TO TIME CONSTRAINTS.

#include <EntityContainer.h> Inheritance diagram for EntityContainer::



### Public Member Functions

- [EntityContainer](#) ([Entity](#) &entity)  
*Constructor for Container that creates a Container in the world.*
- void [getLoot](#) ([Entity](#) &entity)  
*Function that gives the currently selected Container loot.*
- bool [hasLoot](#) ()  
*Function that checks to see if the [Entity](#) contains loot or not.*

### 3.5.1 Detailed Description

CLASS SCRAPPED DUE TO TIME CONSTRAINTS.

### 3.5.2 Constructor & Destructor Documentation

#### 3.5.2.1 EntityContainer::EntityContainer (Entity & entity)

Constructor for Container that creates a Container in the world. detailed description

##### Parameters:

← [Entity](#) &entity

##### Returns:

none

### 3.5.3 Member Function Documentation

#### 3.5.3.1 void EntityContainer::getLoot (Entity & entity)

Function that gives the currently selected Container loot. detailed description

##### Parameters:

← [EntityContainer](#) &entity

**Returns:**

void

**3.5.3.2 bool EntityContainer::hasLoot () [virtual]**

Function that checks to see if the [Entity](#) contains loot or not. detailed description

**Parameters:**

← *none*

**Returns:**

boolean

Implements [Entity](#).

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

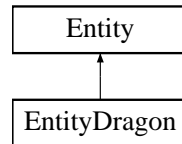
- EntityContainer.h



## 3.6 EntityDragon Class Reference

class [EntityDragon](#)

#include <EntityDragon.h>Inheritance diagram for EntityDragon::



### Public Member Functions

- [EntityDragon](#) ()  
*Constructor for the Dragon that puts a Dragon into the world.*
- void [attack](#) ([Entity](#) &entity)  
*Function that calls an attack on the currently selected Dragon.*
- bool [isAlive](#) ()  
*Function that checks to see if the [Entity](#) is alive and returns a boolean value that does such.*
- void [update](#) ()  
*Function that updates the Entity's next decision UNUSED.*
- void [getLoot](#) ([Entity](#) &entity)  
*Function that gives loot to the currently selected Dragon.*
- bool [hasLoot](#) ()  
*Function that checks to see if the [Entity](#) has loot and returns a boolean value to determine such.*

### 3.6.1 Detailed Description

class [EntityDragon](#)

### 3.6.2 Member Function Documentation

#### 3.6.2.1 void EntityDragon::attack (Entity & entity)

Function that calls an attack on the currently selected Dragon.

#### Parameters:

← [EntityDragon](#) &entity

### 3.6.2.2 void EntityDragon::getLoot (Entity & *entity*)

Function that gives loot to the currently selected Dragon.

#### Parameters:

← *EntityDragon* &entity

### 3.6.2.3 bool EntityDragon::hasLoot () [virtual]

Function that checks to see if the [Entity](#) has loot and returns a boolean value to determine such.

#### Returns:

boolean if yes(true) or no(false)

Implements [Entity](#).

### 3.6.2.4 bool EntityDragon::isAlive () [virtual]

Function that checks to see if the [Entity](#) is alive and returns a boolean value that does such.

#### Parameters:

← *none*

#### Returns:

boolean to check if alive(true) or dead(false)

Implements [Entity](#).

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

- EntityDragon.h

## 3.7 EntityFactory Class Reference

### Public Member Functions

- `Entity * getEnemy (ENTITYTYPE e, Room *x)`  
*Returns a generated enemy entity.*
- `std::vector< Item * > makeInventory (Room *x)`  
*Returns an inventory containing a random item, uses room as a seed.*

### Static Public Member Functions

- `static EntityFactory * getInstance ()`  
*Grabs the current single instance of EntityFactory.*

#### 3.7.1 Member Function Documentation

##### 3.7.1.1 Entity \* EntityFactory::getEnemy (ENTITYTYPE e, Room \* x)

Returns a generated enemy entity.

##### Parameters:

- ← `Entity` enum you want to make
- ← `Room` where entity is made, This is used to scale the entity as the dungeon progresses

##### Returns:

Returns the generated enemy

##### 3.7.1.2 EntityFactory \* EntityFactory::getInstance () [static]

Grabs the current single instance of `EntityFactory`.

##### Returns:

`EntityFactory`, the current instance of the entity factory

##### 3.7.1.3 std::vector< Item \* > EntityFactory::makeInventory (Room \* x)

Returns an inventory containing a random item, uses room as a seed.

##### Parameters:

- ← `Room` to make the item in, this is used to scale the item

##### Returns:

Returns an inventory containing a random item

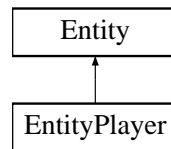
The documentation for this class was generated from the following files:

- EntityFactory.h
- EntityFactory.cc

## 3.8 EntityPlayer Class Reference

[Entity](#) class.

`#include <EntityPlayer.h>`Inheritance diagram for EntityPlayer::



### Public Member Functions

- [EntityPlayer](#) ()  
*constructor for player*
- [~EntityPlayer](#) ()  
*destructor for player*
- bool [attack](#) ([Entity](#) &entity)  
*Attacks the given entity.*
- bool [isAlive](#) ()  
*Returns true if the entity is alive, false if dead.*
- void [getLoot](#) ([Entity](#) &entity)  
*Gives the loot from the entity to the given entity (transfers inventory to the referenced entity).*
- bool [hasLoot](#) ()  
*Returns true if the entity has items in their inventory, false otherwise.*
- void [equipArmor](#) ([Item](#) \*item)  
*Updates the player's stats based on their new item.*
- void [equipWeapon](#) ([Item](#) \*item)  
*equips item to available spot*
- void [usePotion](#) ([Item](#) \*item, int x)  
*uses potion*

### 3.8.1 Detailed Description

[Entity](#) class.

## 3.8.2 Constructor & Destructor Documentation

### 3.8.2.1 EntityPlayer::EntityPlayer ()

constructor for player constructs the player

### 3.8.2.2 EntityPlayer::~~EntityPlayer ()

destructor for player destructs the player pointer

## 3.8.3 Member Function Documentation

### 3.8.3.1 bool EntityPlayer::attack (Entity & *entity*)

Attacks the given entity. Detailed description

#### Parameters:

← *entity,a* reference to the entity that will be attacked

### 3.8.3.2 void EntityPlayer::equipArmor (Item \* *item*)

Updates the player's stats based on their new item. Detailed description

#### Parameters:

*none* equips item to available spot Changes the equipped weapon slot then forces updates to player's stats

*none*

### 3.8.3.3 void EntityPlayer::equipWeapon (Item \* *item*)

equips item to available spot Changes the equipped armor slot then forces update to player's stats

### 3.8.3.4 void EntityPlayer::getLoot (Entity & *entity*)

Gives the loot from the entity to the given entity (transfers inventory to the referenced entity). Detailed description

#### Parameters:

← *entity,the* entity you want to get its inventory from

### 3.8.3.5 bool EntityPlayer::hasLoot () [virtual]

Returns true if the entity has items in their inventory, false otherwise. Detailed description

#### Returns:

boolean returns if the entity has loot or not

Implements [Entity](#).

#### 3.8.3.6 bool EntityPlayer::isAlive () [virtual]

Returns true if the entity is alive, false if dead. Detailed description

##### Returns:

Returns true if the entity is alive, false if dead

Implements [Entity](#).

#### 3.8.3.7 void EntityPlayer::usePotion (Item \* *item*, int *x*)

uses potion Uses the potion and gets rid of it

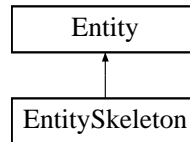
The documentation for this class was generated from the following files:

- EntityPlayer.h
- EntityPlayer.cc

## 3.9 EntitySkeleton Class Reference

[EntitySkeleton](#).

#include <EntitySkeleton.h> Inheritance diagram for EntitySkeleton::



### Public Member Functions

- bool [attack](#) ([Entity](#) &entity)  
*Function that will call an attack on the Skeleton provided.*
- bool [isAlive](#) ()  
*Function that checks to see if the Skeleton is alive or not, a boolean value does so.*
- void [update](#) ()  
*Function that updates the Skeleton's next decision.*
- void [getLoot](#) ([Entity](#) &entity)  
*Function that gives loot to the selected Skeleton.*
- bool [hasLoot](#) ()  
*Function that checks to see if the [Entity](#) has loot and returns a boolean value to do so.*

### 3.9.1 Detailed Description

[EntitySkeleton](#).

### 3.9.2 Member Function Documentation

#### 3.9.2.1 bool EntitySkeleton::attack (Entity & entity)

Function that will call an attack on the Skeleton provided.

**Parameters:**

← [Entity](#) &entity

#### 3.9.2.2 void EntitySkeleton::getLoot (Entity & entity)

Function that gives loot to the selected Skeleton.

**Parameters:**

← [Entity](#) &entity



### 3.9.2.3 bool EntitySkeleton::hasLoot () [virtual]

Function that checks to see if the [Entity](#) has loot and returns a boolean value to do so.

**Returns:**

boolean to see if has loot(true) or not(false)

Implements [Entity](#).

### 3.9.2.4 bool EntitySkeleton::isAlive () [virtual]

Function that checks to see if the Skeleton is alive or not, a boolean value does so.

**Returns:**

boolean to see if alive(true) or dead(false)

Implements [Entity](#).

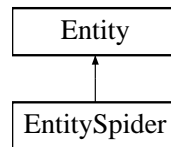
The documentation for this class was generated from the following files:

- EntitySkeleton.h
- EntitySkeleton.cc

## 3.10 EntitySpider Class Reference

class [EntitySpider](#)

#include <EntitySpider.h> Inheritance diagram for EntitySpider::



### Public Member Functions

- bool [attack](#) ([Entity](#) &entity)  
*Function that will call an attack on the Spider given.*
- bool [isAlive](#) ()  
*Function checks to see if the Spider is currently alive or not and returns a boolean value to determine so.*
- void [update](#) ()  
*Function updates the Spider to make a decision one thing at a time.*
- void [getLoot](#) ([Entity](#) &entity)  
*Function that gives loot to the Spider passed in.*
- bool [hasLoot](#) ()  
*Function checks to see if Spider passed in contains loot.*

### 3.10.1 Detailed Description

class [EntitySpider](#)

### 3.10.2 Member Function Documentation

#### 3.10.2.1 bool EntitySpider::attack (Entity & entity)

Function that will call an attack on the Spider given.

**Parameters:**

← [Entity](#) &entity

#### 3.10.2.2 void EntitySpider::getLoot (Entity & entity)

Function that gives loot to the Spider passed in.

**Parameters:**

← [Entity](#) &entity

### 3.10.2.3 bool EntitySpider::hasLoot () [virtual]

Function checks to see if Spider passed in contains loot.

**Returns:**

boolean if has loot(true) or not(false)

Implements [Entity](#).

### 3.10.2.4 bool EntitySpider::isAlive () [virtual]

Function checks to see if the Spider is currently alive or not and returns a boolean value to determine so.

**Returns:**

boolean to see if alive(true) or dead(false)

Implements [Entity](#).

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

- EntitySpider.h
- EntitySpider.cc

## 3.11 EntityTester Class Reference

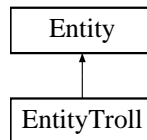
The documentation for this class was generated from the following files:

- EntityTest.h
- EntityTest.cc

## 3.12 EntityTroll Class Reference

class [EntityTroll](#)

#include <EntityTroll.h> Inheritance diagram for EntityTroll::



### Public Member Functions

- bool [attack](#) ([Entity](#) &entity)  
*Function that calls an attack on the given Troll.*
- bool [isAlive](#) ()  
*Function that checks to see if the [Entity](#) is alive and returns a boolean value to determine such.*
- void [getLoot](#) ([Entity](#) &entity)  
*Function that updates the Entity's next decision.*
- bool [hasLoot](#) ()  
*Function that checks to see if the [Entity](#) has loot and returns a boolean value to do so.*

### 3.12.1 Detailed Description

class [EntityTroll](#)

### 3.12.2 Member Function Documentation

#### 3.12.2.1 bool EntityTroll::attack (Entity & entity)

Function that calls an attack on the given Troll.

##### Parameters:

← [EntityTroll](#) &entity

#### 3.12.2.2 void EntityTroll::getLoot (Entity & entity)

Function that updates the Entity's next decision. Function that gives loot to the currently selected Troll

##### Parameters:

← [EntityTroll](#) &entity

### 3.12.2.3 `bool EntityTroll::hasLoot ()` **[virtual]**

Function that checks to see if the [Entity](#) has loot and returns a boolean value to do so.

**Returns:**

boolean if has loot(true) or not(false)

Implements [Entity](#).

### 3.12.2.4 `bool EntityTroll::isAlive ()` **[virtual]**

Function that checks to see if the [Entity](#) is alive and returns a boolean value to determine such.

**Returns:**

boolean if alive(true) or dead(false)

Implements [Entity](#).

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

- EntityTroll.h
- EntityTroll.cc

## 3.13 Game Class Reference

```
#include <Game.h>
```

### Public Member Functions

- `~Game ()`  
*game destructor*
- `void start ()`  
*Initializes and starts a new game.*
- `void createCharacter ()`  
*Asks user details to create their character.*
- `bool generateMap ()`  
*Generates the game map and sets the generated map to the current game map.*
- `void mainLoop ()`  
*Main loop of the game, responsible for updating game logic/game world.*
- `void roomOption ()`  
*checks the rooms options*
- `void combat (Room *room)`  
*checks the combat choices of the room*
- `void inventory ()`  
*checks the inventory of the player*
- `void displayOptions ()`  
*Displays the available options that the user has in each room based on player state.*
- `void openInventory (EntityPlayer &player)`  
*Displays the users inventory to the screen.*
- `int getOption ()`  
*Asks the user for a integer input.*
- `void print (std::string text)`  
*Outputs the given text to the screen.*
- `int getDifficulty () const`  
*Returns the difficulty of the game.*
- `void setDifficulty (int difficulty)`  
*sets the difficulty of the game*
- `void log (std::string message)`

*logs a debug message to the console*

- void `setDebug` (bool d)  
*sets debug messages on or off*
- bool `getDebug` () const  
*gets the current debug state on or off*
- void `exit` ()  
*Exits the game safely.*
- void `clear` ()  
*Clears the console screen.*

### 3.13.1 Detailed Description

/brief `Game` class

### 3.13.2 Member Function Documentation

#### 3.13.2.1 bool Game::getDebug () const

gets the current debug state on or off

**Returns:**

bool, true if on false if off

#### 3.13.2.2 int Game::getDifficulty () const

Returns the difficulty of the game.

**Returns:**

Returns the difficulty of the game

#### 3.13.2.3 int Game::getOption ()

Asks the user for a integer input.

**Returns:**

Returns the option the user picked



**3.13.2.4 void Game::log (std::string *message*)**

logs a debug message to the console

**Parameters:**

← *message* to log to console

**3.13.2.5 void Game::openInventory (EntityPlayer & *player*)**

Displays the users inventory to the screen.

**Parameters:**

*inout*] *player*, the player you wish to display inventory of

**3.13.2.6 void Game::print (std::string *text*)**

Outputs the given text to the screen.

**Parameters:**

← *Text* to be outputed to the screen

**3.13.2.7 void Game::setDebug (bool *d*)**

sets debug messages on or off [in] the toggle on or off

**3.13.2.8 void Game::setDifficulty (int *difficulty*)**

sets the difficulty of the game

**Parameters:**

← *difficulty,the* difficulty you want to set the game to (1 - 3)

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

- Game.h
- Game.cc

## 3.14 GameTest Class Reference

### Public Member Functions

- void **setUp** ()
- void **testMapWorks** ()
- void **tearDown** ()

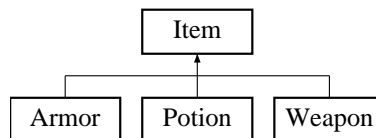
The documentation for this class was generated from the following files:

- GameTest.h
- GameTest.cc

## 3.15 Item Class Reference

**Item** Class pure virtual interface for item classes.

`#include <Item.h>`Inheritance diagram for `Item::`



### Public Member Functions

- virtual `~Item ()`  
*virtual destructor*
- `std::string getName ()`  
*Gets the name of the item and returns it.*
- `std::string getDescription ()`  
*Gets the description of the item and returns it.*
- virtual `bool use ()=0`  
*Equips the item.*
- `void setName (std::string n)`  
*sets name*
- `void setDescription (std::string n)`  
*sets description*
- `Attributes & getStats ()`  
*gets stats*
- `bool isWeapon ()`  
*checks if weapon*
- `bool isArmor ()`  
*checks if armor*
- `bool isPotion ()`  
*checks if potion*
- `void setNum (int num)`  
*sets the number*

### 3.15.1 Detailed Description

[Item](#) Class pure virtual interface for item classes.

### 3.15.2 Member Function Documentation

#### 3.15.2.1 `std::string Item::getDescription () [inline]`

Gets the description of the item and returns it. Detailed description /return string of the items description

#### 3.15.2.2 `std::string Item::getName () [inline]`

Gets the name of the item and returns it.

**Returns:**

string containing the item name

#### 3.15.2.3 `bool Item::isArmor () [inline]`

checks ifarmor

**Returns:**

boolean telling whether it is armor or not

#### 3.15.2.4 `bool Item::isPotion () [inline]`

checks if potion

**Returns:**

boolean telling whether it is potion or not

#### 3.15.2.5 `bool Item::isWeapon () [inline]`

checks if weapon

**Returns:**

boolean telling whether it is weapon or not

#### 3.15.2.6 `virtual bool Item::use () [pure virtual]`

Equips the item. Detailed description /return bool if the item was sucesfully used

Implemented in [Armor](#), [Potion](#), and [Weapon](#).

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

- [Item.h](#)

## 3.16 ItemFactory Class Reference

[ItemFactory](#) class for creation of weapons, armor, and potions.

```
#include <ItemFactory.h>
```

### Public Member Functions

- [Item](#) \* [getItem](#) (ITEMTYPE i, [Room](#) \*roomIn)  
*Creates a item based on the given type and level Detailed description.*

### Static Public Member Functions

- static [ItemFactory](#) \* [getInstance](#) ()  
*Gets current instance of the item factory.*

#### 3.16.1 Detailed Description

[ItemFactory](#) class for creation of weapons, armor, and potions.

#### 3.16.2 Member Function Documentation

##### 3.16.2.1 [ItemFactory](#) \* [ItemFactory::getInstance](#) () [static]

Gets current instance of the item factory.

##### Returns:

[ItemFactory](#), the current instance of the item factory

##### 3.16.2.2 [Item](#) \* [ItemFactory::getItem](#) (ITEMTYPE i, [Room](#) \* roomIn)

Creates a item based on the given type and level Detailed description.

##### Parameters:

- ← *ItemType,the* type of item you want to create
- ← *level,Integer* passed in by room to determine the power of the item

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

- ItemFactory.h
- ItemFactory.cc

## 3.17 ItemTest Class Reference

### Public Member Functions

- void [createItem](#) ()  
*create the Items*
- void [destructItem](#) ()  
*destruct items created*
- void [isUsed](#) ()  
*check to see if item is actually used properly*
- void [properDescription](#) ()  
*check to see if the proper description is returned*
- void [properName](#) ()  
*check to see if the proper item name is returned*
- void [isRightItem](#) ()  
*check to see if its the right item*

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

- ItemTest.h
- ItemTest.cc

## 3.18 Map Class Reference

Map Class.

```
#include <Map.h>
```

### Public Member Functions

- **Map** (int diff, **EntityPlayer** \*p)  
*Constructs the map of the game.*
- **~Map** ()  
*Deconstructs the map of the game.*
- bool **generate** ()  
*Generates map of the rooms.*
- void **movePlayer** (**Room** \*room)  
*Moves player to given room in map.*
- bool **displayDescription** ()  
*Displays description of the room.*
- **Room** \* **getCurRoom** () const  
*gets the hieght of the room in the map tree.*
- void **getRandomRoom** (**Room** \*&c)
- int **getX** ()
- int **getY** ()
- std::vector< int > **getOpVec** ()
- **Room** \* **getMap** (int x, int y)
- std::vector< std::vector< **Room** \* > > **getFullMap** ()

### 3.18.1 Detailed Description

Map Class.

### 3.18.2 Constructor & Destructor Documentation

#### 3.18.2.1 Map::Map (int diff, EntityPlayer \* p)

Constructs the map of the game.

**Parameters:**

← *difficulty,the* difficulty of the game.

*inout]* &player, a reference to the player.

### 3.18.3 Member Function Documentation

#### 3.18.3.1 `bool Map::displayDescription ()`

Displays description of the room.

**Returns:**

Bool: returns true if description succesfully prints.

#### 3.18.3.2 `bool Map::generate ()`

Generates map of the rooms.

**Returns:**

bool : returns true if map is generated correctly.

#### 3.18.3.3 `Room* Map::getCurRoom () const [inline]`

gets the hieght of the room in the map tree. /param[in] &room, a reference to the room that the height is need for. /return const int: returns the hieght of the room in the map tree.

#### 3.18.3.4 `void Map::movePlayer (Room * room)`

Moves player to given room in map.

**Parameters:**

← *the* pointer to the room you want to move the player to

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

- Map.h
- Map.cc



## 3.19 MapTest Class Reference

### Public Member Functions

- void **setUp** ()
- void **tearDown** ()
- void **generation** ()
- void **isDescribed** ()

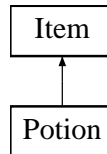
The documentation for this class was generated from the following files:

- MapTest.h
- MapTest.cc

## 3.20 Potion Class Reference

[Weapon](#) Class to be generated by the [ItemFactory](#).

`#include <Potion.h>`Inheritance diagram for Potion::



### Public Member Functions

- `bool use ()`  
*Equips the potion.*

#### 3.20.1 Detailed Description

[Weapon](#) Class to be generated by the [ItemFactory](#).

#### 3.20.2 Member Function Documentation

##### 3.20.2.1 `bool Potion::use () [virtual]`

Equips the potion. Detailed description /return bool if the potion was sucessfully used

Implements [Item](#).

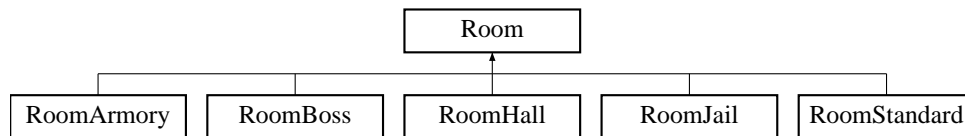
The documentation for this class was generated from the following files:

- `Potion.h`
- `Potion.cc`

## 3.21 Room Class Reference

Class [Room](#).

#include <Room.h> Inheritance diagram for Room::



### Public Member Functions

- virtual void [generate](#) ()=0  
*Generates the room.*
- virtual std::string [getDescription](#) ()=0  
*Gets the description for the room.*
- void [setNorth](#) ([Room](#) \*p)
- void [setSouth](#) ([Room](#) \*p)
- void [setEast](#) ([Room](#) \*p)
- void [setWest](#) ([Room](#) \*p)
- [Room](#) \* [getNorth](#) ()
- [Room](#) \* [getSouth](#) ()
- [Room](#) \* [getEast](#) ()
- [Room](#) \* [getWest](#) ()
- void [setX](#) (int x)
- void [setY](#) (int y)
- int [getX](#) ()
- int [getY](#) ()
- int [getOptions](#) (int x)
- std::vector< int > [setOptions](#) ()
- [Room](#) & [operator=](#) (const [Room](#) &r)
- bool [operator==](#) (const [Room](#) &r1)
- void [setVEC](#) (int x, int y)
- void [setEntity](#) ([Entity](#) \*x)
- [Entity](#) \* [getEntity](#) ()

### 3.21.1 Detailed Description

Class [Room](#).

### 3.21.2 Member Function Documentation

#### 3.21.2.1 `virtual void Room::generate () [pure virtual]`

Generates the room. This function generates which enemies will appear in the room with player

**Parameters:**

← *none*

**Returns:**

void

Implemented in [RoomArmory](#), [RoomBoss](#), [RoomHall](#), [RoomJail](#), and [RoomStandard](#).

#### 3.21.2.2 `virtual std::string Room::getDescription () [pure virtual]`

Gets the description for the room. This function tells the user what is in and around the room

**Parameters:**

← *none*

**Returns:**

string-that describes the room's condition

Implemented in [RoomArmory](#), [RoomBoss](#), [RoomHall](#), [RoomJail](#), and [RoomStandard](#).

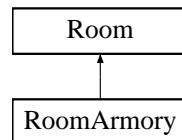
The documentation for this class was generated from the following file:

- Room.h

## 3.22 RoomArmory Class Reference

Class [RoomArmory](#).

`#include <RoomArmory.h>`Inheritance diagram for RoomArmory::



### Public Member Functions

- void [generate](#) ()  
*Generates the armory room.*
- std::string [getDescription](#) ()  
*Gets the description for the armory room.*
- bool [isEmpty](#) ()  
*Checks to see if the room enemies are in the armory room.*

### 3.22.1 Detailed Description

Class [RoomArmory](#).

### 3.22.2 Member Function Documentation

#### 3.22.2.1 void RoomArmory::generate () [virtual]

Generates the armory room. This function generates which enemies will appear in the armory with player

##### Parameters:

← *none*

##### Returns:

void

Implements [Room](#).

#### 3.22.2.2 std::string RoomArmory::getDescription () [virtual]

Gets the description for the armory room. This function tells the user what is in and around the armory

##### Parameters:

← *none*

**Returns:**

string-that describes the armory's condition

Implements [Room](#).

**3.22.2.3 bool RoomArmory::isEmpty ()**

Checks to see if the room enemies are in the armory room. This function checks to see if anything is in the armory with player

**Parameters:**

← *none*

**Returns:**

bool - true if the armory is empty otherwise false

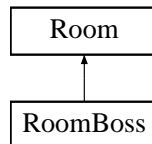
The documentation for this class was generated from the following files:

- RoomArmory.h
- RoomArmory.cc

## 3.23 RoomBoss Class Reference

Class [RoomBoss](#).

#include <RoomBoss.h>Inheritance diagram for RoomBoss::



### Public Member Functions

- void [generate](#) ()  
*Generates the boss room.*
- std::string [getDescription](#) ()  
*Gets the description for the boss room.*
- bool [isEmpty](#) ()  
*Checks to see if the room enemies are in the boss room.*

#### 3.23.1 Detailed Description

Class [RoomBoss](#).

#### 3.23.2 Member Function Documentation

##### 3.23.2.1 void RoomBoss::generate () [virtual]

Generates the boss room. This function generates the boss that will appear in the boss room with player

##### Parameters:

← *none*

##### Returns:

void

Implements [Room](#).

##### 3.23.2.2 std::string RoomBoss::getDescription () [virtual]

Gets the description for the boss room. This function tells the user what is in and around the boss room

##### Parameters:

← *none*

**Returns:**

string-that describes the boss room's condition

Implements [Room](#).

**3.23.2.3 bool RoomBoss::isEmpty ()**

Checks to see if the room enemies are in the boss room. This function checks to see if anything is in the room with playe

**Parameters:**

← *none*

**Returns:**

bool - true if the boss room is empty otherwise false

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

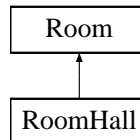
- RoomBoss.h
- RoomBoss.cc



## 3.24 RoomHall Class Reference

Class [RoomHall](#).

`#include <RoomHall.h>`Inheritance diagram for RoomHall::



### Public Member Functions

- void [generate](#) ()  
*Generates the hall room.*
- std::string [getDescription](#) ()  
*Gets the description for the hall room.*
- bool [isEmpty](#) ()  
*Checks to see if the room enemies are in the hall room.*

### 3.24.1 Detailed Description

Class [RoomHall](#).

### 3.24.2 Member Function Documentation

#### 3.24.2.1 void RoomHall::generate () [virtual]

Generates the hall room. This function generates which enemies will appear in the hall with player

**Parameters:**

← *none*

**Returns:**

void

Implements [Room](#).

#### 3.24.2.2 std::string RoomHall::getDescription () [virtual]

Gets the description for the hall room. This function tells the user what is in and around the hall

**Parameters:**

← *none*

**Returns:**

string-that describes the room's condition

Implements [Room](#).

**3.24.2.3 bool RoomHall::isEmpty ()**

Checks to see if the room enemies are in the hall room. This function checks to see if anything is in the hall with player

**Parameters:**

← *none*

**Returns:**

bool - true if the room is empty otherwise false

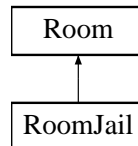
The documentation for this class was generated from the following files:

- RoomHall.h
- RoomHall.cc

## 3.25 RoomJail Class Reference

Class [RoomJail](#).

`#include <RoomJail.h>`Inheritance diagram for RoomJail::



### Public Member Functions

- void [generate](#) ()  
*Generates the jail room.*
- std::string [getDescription](#) ()  
*Gets the description for the jail room.*
- bool [isEmpty](#) ()  
*Checks to see if the room enemies are in the jail room.*

### 3.25.1 Detailed Description

Class [RoomJail](#).

### 3.25.2 Member Function Documentation

#### 3.25.2.1 void RoomJail::generate () [virtual]

Generates the jail room. This function generates which enemies will appear in the jail with player

**Parameters:**

← *none*

**Returns:**

void

Implements [Room](#).

#### 3.25.2.2 std::string RoomJail::getDescription () [virtual]

Gets the description for the jail room. This function tells the user what is in and around the jail

**Parameters:**

← *none*

**Returns:**

string-that describes the jail's condition

Implements [Room](#).

**3.25.2.3 bool RoomJail::isEmpty ()**

Checks to see if the room enemies are in the jail room. This function checks to see if anything is in the jail with player

**Parameters:**

← *none*

**Returns:**

bool - true if the jail is empty otherwise false

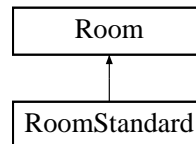
The documentation for this class was generated from the following files:

- RoomJail.h
- RoomJail.cc

## 3.26 RoomStandard Class Reference

Class [RoomStandard](#).

`#include <RoomStandard.h>`Inheritance diagram for RoomStandard::



### Public Member Functions

- void [generate](#) ()  
*Generates the standard room.*
- std::string [getDescription](#) ()  
*Gets the description for the standard room.*
- bool [isEmpty](#) ()  
*Checks to see if the room enemies are in the standard room.*

### 3.26.1 Detailed Description

Class [RoomStandard](#).

### 3.26.2 Member Function Documentation

#### 3.26.2.1 void RoomStandard::generate () [virtual]

Generates the standard room. This function generates which enemies will appear in the room with player

##### Parameters:

← *none*

##### Returns:

void

Implements [Room](#).

#### 3.26.2.2 std::string RoomStandard::getDescription () [virtual]

Gets the description for the standard room. This function tells the user what is in and around the room

##### Parameters:

← *none*

**Returns:**

string-that describes the room's condition

Implements [Room](#).

**3.26.2.3 bool RoomStandard::isEmpty ()**

Checks to see if the room enemies are in the standard room. This function checks to see if anything is in the room with player

**Parameters:**

← *none*

**Returns:**

bool- true if the room is empty otherwise false

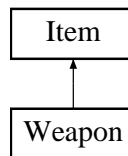
The documentation for this class was generated from the following files:

- RoomStandard.h
- RoomStandard.cc

## 3.27 Weapon Class Reference

[Weapon](#) Class to be generated by the [ItemFactory](#).

`#include <Weapon.h>`Inheritance diagram for `Weapon::`



### Public Member Functions

- `bool use ()`  
*Equips the weapon.*

### 3.27.1 Detailed Description

[Weapon](#) Class to be generated by the [ItemFactory](#).

### 3.27.2 Member Function Documentation

#### 3.27.2.1 `bool Weapon::use () [virtual]`

Equips the weapon. /return bool if the item was used

Implements [Item](#).

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

- `Weapon.h`
- `Weapon.cc`

# Index

- ~EntityPlayer
  - EntityPlayer, [22](#)
- Armor, [5](#)
  - use, [5](#)
- attack
  - Entity, [11](#)
  - EntityBat, [13](#)
  - EntityDragon, [17](#)
  - EntityPlayer, [22](#)
  - EntitySkeleton, [24](#)
  - EntitySpider, [26](#)
  - EntityTroll, [29](#)
- Attributes, [7](#)
  - getAgility, [8](#)
  - getArmor, [8](#)
  - getDamage, [8](#)
  - getHealthPoints, [8](#)
  - getStrength, [8](#)
  - getVitality, [8](#)
  - setAgility, [8](#)
  - setArmor, [9](#)
  - setDamage, [9](#)
  - setHealthPoints, [9](#)
  - setStrength, [9](#)
  - setVitality, [9](#)
- displayDescription
  - Map, [40](#)
- Entity, [10](#)
  - attack, [11](#)
  - getAttributes, [11](#)
  - getInventory, [11](#)
  - getLoot, [11](#)
  - getName, [11](#)
  - hasLoot, [11](#)
  - isAlive, [12](#)
  - setInventory, [12](#)
  - setName, [12](#)
- EntityBat, [13](#)
  - attack, [13](#)
  - getLoot, [13](#)
  - hasLoot, [13](#)
  - isAlive, [14](#)
- EntityContainer, [15](#)
  - EntityContainer, [15](#)
  - getLoot, [15](#)
  - hasLoot, [16](#)
- EntityDragon, [17](#)
  - attack, [17](#)
  - getLoot, [17](#)
  - hasLoot, [18](#)
  - isAlive, [18](#)
- EntityFactory, [19](#)
  - getEnemy, [19](#)
  - getInstance, [19](#)
  - makeInventory, [19](#)
- EntityPlayer, [21](#)
  - ~EntityPlayer, [22](#)
  - attack, [22](#)
  - EntityPlayer, [22](#)
  - equipArmor, [22](#)
  - equipWeapon, [22](#)
  - getLoot, [22](#)
  - hasLoot, [22](#)
  - isAlive, [22](#)
  - usePotion, [23](#)
- EntitySkeleton, [24](#)
  - attack, [24](#)
  - getLoot, [24](#)
  - hasLoot, [24](#)
  - isAlive, [25](#)
- EntitySpider, [26](#)
  - attack, [26](#)
  - getLoot, [26](#)
  - hasLoot, [26](#)
  - isAlive, [27](#)
- EntityTester, [28](#)
- EntityTroll, [29](#)
  - attack, [29](#)
  - getLoot, [29](#)
  - hasLoot, [29](#)
  - isAlive, [30](#)
- equipArmor
  - EntityPlayer, [22](#)
- equipWeapon
  - EntityPlayer, [22](#)
- Game, [31](#)



- getDebug, 32
- getDifficulty, 32
- getOption, 32
- log, 32
- openInventory, 33
- print, 33
- setDebug, 33
- setDifficulty, 33
- GameTest, 34
- generate
  - Map, 40
  - Room, 44
  - RoomArmory, 45
  - RoomBoss, 47
  - RoomHall, 49
  - RoomJail, 51
  - RoomStandard, 53
- getAgility
  - Attributes, 8
- getArmor
  - Attributes, 8
- getAttributes
  - Entity, 11
- getCurRoom
  - Map, 40
- getDamage
  - Attributes, 8
- getDebug
  - Game, 32
- getDescription
  - Item, 36
  - Room, 44
  - RoomArmory, 45
  - RoomBoss, 47
  - RoomHall, 49
  - RoomJail, 51
  - RoomStandard, 53
- getDifficulty
  - Game, 32
- getEnemy
  - EntityFactory, 19
- getHealthPoints
  - Attributes, 8
- getInstance
  - EntityFactory, 19
  - ItemFactory, 37
- getInventory
  - Entity, 11
- getItem
  - ItemFactory, 37
- getLoot
  - Entity, 11
  - EntityBat, 13
  - EntityContainer, 15
  - EntityDragon, 17
  - EntityPlayer, 22
  - EntitySkeleton, 24
  - EntitySpider, 26
  - EntityTroll, 29
- getName
  - Entity, 11
  - Item, 36
- getOption
  - Game, 32
- getStrength
  - Attributes, 8
- getVitality
  - Attributes, 8
- hasLoot
  - Entity, 11
  - EntityBat, 13
  - EntityContainer, 16
  - EntityDragon, 18
  - EntityPlayer, 22
  - EntitySkeleton, 24
  - EntitySpider, 26
  - EntityTroll, 29
- isAlive
  - Entity, 12
  - EntityBat, 14
  - EntityDragon, 18
  - EntityPlayer, 22
  - EntitySkeleton, 25
  - EntitySpider, 27
  - EntityTroll, 30
- isArmor
  - Item, 36
- isEmpty
  - RoomArmory, 46
  - RoomBoss, 48
  - RoomHall, 50
  - RoomJail, 52
  - RoomStandard, 54
- isPotion
  - Item, 36
- isWeapon
  - Item, 36
- Item, 35
  - getDescription, 36
  - getName, 36
  - isArmor, 36
  - isPotion, 36
  - isWeapon, 36
  - use, 36
- ItemFactory, 37
  - getInstance, 37

- getItem, 37
- ItemTest, 38
- log
  - Game, 32
- makeInventory
  - EntityFactory, 19
- Map, 39
  - displayDescription, 40
  - generate, 40
  - getCurRoom, 40
  - Map, 39
  - movePlayer, 40
- MapTest, 41
- movePlayer
  - Map, 40
- openInventory
  - Game, 33
- Potion, 42
  - use, 42
- print
  - Game, 33
- Room, 43
  - generate, 44
  - getDescription, 44
- RoomArmory, 45
  - generate, 45
  - getDescription, 45
  - isEmpty, 46
- RoomBoss, 47
  - generate, 47
  - getDescription, 47
  - isEmpty, 48
- RoomHall, 49
  - generate, 49
  - getDescription, 49
  - isEmpty, 50
- RoomJail, 51
  - generate, 51
  - getDescription, 51
  - isEmpty, 52
- RoomStandard, 53
  - generate, 53
  - getDescription, 53
  - isEmpty, 54
- setAgility
  - Attributes, 8
- setArmor
  - Attributes, 9
- setDamage
  - Attributes, 9
- setDebug
  - Game, 33
- setDifficulty
  - Game, 33
- setHealthPoints
  - Attributes, 9
- setInventory
  - Entity, 12
- setName
  - Entity, 12
- setStrength
  - Attributes, 9
- setVitality
  - Attributes, 9
- use
  - Armor, 5
  - Item, 36
  - Potion, 42
  - Weapon, 55
- usePotion
  - EntityPlayer, 23
- Weapon, 55
  - use, 55