# Rogue Reborn

Generated by Doxygen 1.8.12

# Contents

# Chapter 1

# Hierarchical Index

## 1.1 Class Hierarchy

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

# Chapter 2

# Class Index

## 2.1 Class List

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

# Chapter 3

# File Index

## 3.1 File List

Here is a list of all documented files with brief descriptions:

# Chapter 4

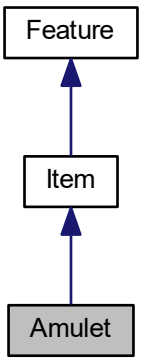
# Class Documentation

## 4.1 Amulet Class Reference

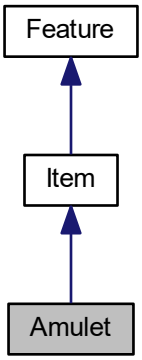
Represents the Amulet of Yendor.

```
#include <amulet.h>
```

Inheritance diagram for Amulet:



Collaboration diagram for Amulet:




### Public Member Functions

- Amulet (Coord, Item::Context)

  *Constructs an Amulet instance.*

### Additional Inherited Members

## 4.1.1 Detailed Description

Represents the Amulet of Yendor.

## 4.1.2 Constructor & Destructor Documentation

### 4.1.2.1 Amulet()

```
Amulet::Amulet (
            Coord location,
            Item::Context context )
```

Constructs an Amulet instance.

**Parameters**

| | | |
|---|---|---|
| in | *location* | Amulet location |
| in | *context* | Amulet context |

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

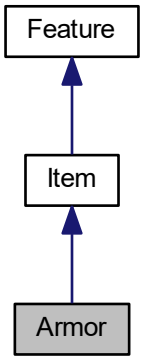
- include/amulet.h
- amulet.cpp

## 4.2 Armor Class Reference

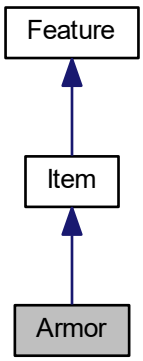
Represents armor.

`#include <armor.h>`

Inheritance diagram for Armor:




Collaboration diagram for Armor:



### Public Member Functions

- Armor (Coord)

  *Constructs an Armor instance with a random type.*
- Armor (Coord, Item::Context, int)

  *Constructs an Armor instance.*
- int getRating ()

  *Gets the rating.*

### Additional Inherited Members

## 4.2.1 Detailed Description

Represents armor.

## 4.2.2 Constructor & Destructor Documentation

#### 4.2.2.1 Armor() [1/2]

```
Armor::Armor (
            Coord location )
```

Constructs an Armor instance with a random type.

**Parameters**

| | | |
|---|---|---|
| in | *location* | Armor location |

#### 4.2.2.2 Armor() [2/2]

```
Armor::Armor (
            Coord location,
            Item::Context context,
            int type )
```

Constructs an Armor instance.

**Parameters**

| | | |
|---|---|---|
| in | *location* | Armor location |
| in | *context* | Armor context |
| in | *type* | Armor type |

## 4.2.3 Member Function Documentation

#### 4.2.3.1 getRating()

```
int Armor::getRating ( )
```

Gets the rating.

**Returns**

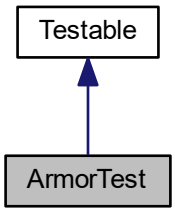
The rating.

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

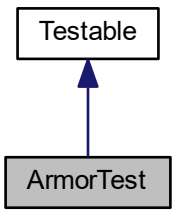
- include/armor.h
- armor.cpp

## 4.3 ArmorTest Class Reference

Inheritance diagram for ArmorTest:




Collaboration diagram for ArmorTest:



### Public Member Functions

- bool **test** ()

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

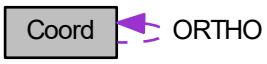
- test/test.armor.cpp

## 4.4 Coord Class Reference

Represents a location within the dungeon or on the screen.

`#include <coord.h>`

Collaboration diagram for Coord:




### Public Member Functions

- Coord (int, int)
  *(x,y) constructor.*
- Coord ()
  *(0,0) constructor.*
- int & operator[ ] (int)
  *Access param dimension magnitude.*
- Coord operator+ (const Coord &)
  *Add two coords together.*
- Coord operator- (const Coord &)
  *Subtract two coords.*
- Coord operator∗ (const int &)
  *Multiply all vector items by scalar.*
- Coord & operator+= (const Coord &)
  *Augmented assignment for addition.*
- Coord & operator-= (const Coord &)
  *Augmented assignment for subtraction.*
- bool operator< (const Coord &) const
  *Order coords by overall magnitude.*
- Coord & operator∗= (const int &)
  *Multiply two coords (item by item).*
- bool operator== (const Coord &)

*True if all vector items equal.*

- bool operator!= (const Coord &)

  *Inverse of == operator.*
- Coord asScreen ()

  *Convert position in level to position in screen.*
- Coord copy ()

  *Return a copy of this coord.*
- bool isAdjacentTo (const Coord &) const

  *Return distance(taxicab) <= 1.*
- std::string toString () const

  *Format as x, y.*
- int distanceTo (const Coord &) const

  *Maximum distance in either dimension.*

### Static Public Attributes

- static Coord ORTHO [4]

  *Set of unit vectors.*

## 4.4.1 Detailed Description

Represents a location within the dungeon or on the screen.

## 4.4.2 Constructor & Destructor Documentation

### 4.4.2.1 Coord() [1/2]

```
Coord::Coord (
            int x,
            int y )
```

(x,y) constructor.

### 4.4.2.2 Coord() [2/2]

```
Coord::Coord ( )
```

(0,0) constructor.

## 4.4.3 Member Function Documentation

### 4.4.3.1 asScreen()

```
Coord Coord::asScreen ( )
```

Convert position in level to position in screen.

#### 4.4.3.2 copy()

```
Coord Coord::copy ( )
```

Return a copy of this coord.

#### 4.4.3.3 isAdjacentTo()

```
bool Coord::isAdjacentTo (
            const Coord & other ) const
```

Return distance(taxicab) <= 1.

#### 4.4.3.4 operator"!=()

```
bool Coord::operator!= (
            const Coord & other )
```

Inverse of == operator.

#### 4.4.3.5 operator*()

```
Coord Coord::operator* (
            const int & scalar )
```

Multiply all vector items by scalar.

#### 4.4.3.6 operator*=()

```
Coord & Coord::operator*= (
            const int & scalar )
```

Multiply two coords (item by item).

#### 4.4.3.7 operator+()

```
Coord Coord::operator+ (
            const Coord & other )
```

Add two coords together.

#### 4.4.3.8 operator+=()

```
Coord & Coord::operator+= (
            const Coord & other )
```

Augmented assignment for addition.

#### 4.4.3.9 operator-()

```
Coord Coord::operator- (
            const Coord & other )
```

Subtract two coords.

#### 4.4.3.10 operator-=()

```
Coord & Coord::operator-= (
            const Coord & other )
```

Augmented assignment for subtraction.

#### 4.4.3.11 operator<()

```
bool Coord::operator< (
            const Coord & other ) const
```

Order coords by overall magnitude.

#### 4.4.3.12 operator==()

```
bool Coord::operator== (
            const Coord & other )
```

True if all vector items equal.

#### 4.4.3.13 operator[]()

```
int & Coord::operator[] (
            int dimension )
```

Access param dimension magnitude.

#### 4.4.3.14 toString()

```
std::string Coord::toString ( ) const
```

Format as x, y.

### 4.4.4 Member Data Documentation

#### 4.4.4.1 ORTHO

```
Coord Coord::ORTHO  [static]
```

**Initial value:**

```
= {Coord(0,1), Coord(1,0),
                          Coord(0,-1), Coord(-1,0)}
```

Set of unit vectors.

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

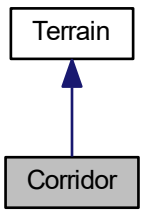
- include/coord.h
- coord.cpp

## 4.5 Corridor Class Reference

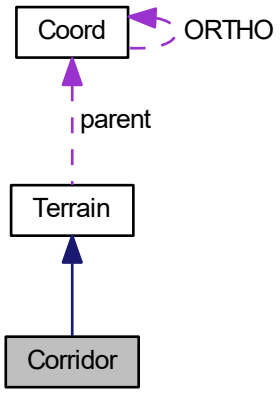
Regular corridor tile.

```
#include <tiles.h>
```

Inheritance diagram for Corridor:



Collaboration diagram for Corridor:




### Additional Inherited Members

#### 4.5.1 Detailed Description

Regular corridor tile.

Has limited visibility and full passability

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

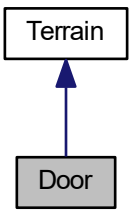
- include/tiles.h
- tiles.cpp

## 4.6 Door Class Reference

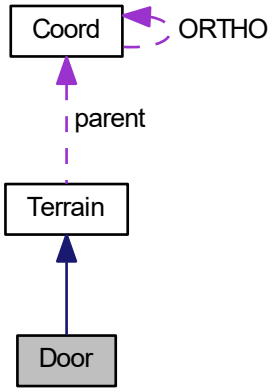
Door tile.

```
#include <tiles.h>
```

Inheritance diagram for Door:



Collaboration diagram for Door:




### Additional Inherited Members

#### 4.6.1 Detailed Description

Door tile.

Only cosmetically different from corridor tile.

**See also**

Corridor

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

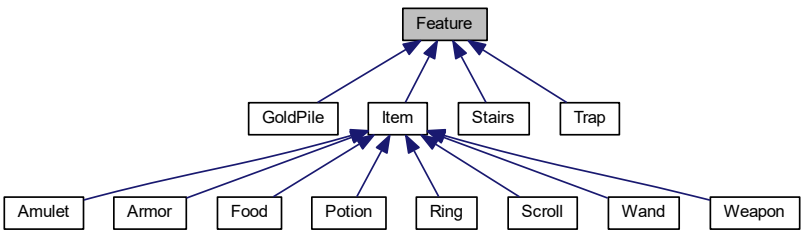
- include/tiles.h
- tiles.cpp

## 4.7 Feature Class Reference

Models a 'thing' in the dungeon that has position and may be visible.

`#include <feature.h>`

Inheritance diagram for Feature:



### Public Member Functions

- Feature (char, Coord)

  *Constructor for symbol, location.*
- char getSymbol ()

  *Getter for symbol.*
- Coord getLocation ()

  *Getter for location.*
- void setLocation (Coord)

  *Setter for location.*
- virtual ~Feature ()

  *Destructor.*

## 4.7.1 Detailed Description

Models a 'thing' in the dungeon that has position and may be visible.

This is to provide a common superclass to various classes that would otherwise cause duplicate code, such as items, staircases, traps, etc

## 4.7.2 Constructor & Destructor Documentation

### 4.7.2.1 Feature()

```
Feature::Feature (
            char symbol,
            Coord location )
```

Constructor for symbol, location.

### 4.7.2.2 ~Feature()

```
Feature::~Feature ( )  [virtual]
```

Destructor.

## 4.7.3 Member Function Documentation

### 4.7.3.1 getLocation()

```
Coord Feature::getLocation ( )
```

Getter for location.

**See also**

location

#### 4.7.3.2 getSymbol()

```
char Feature::getSymbol ( )
```

Getter for symbol.

**See also**

symbol

#### 4.7.3.3 setLocation()

```
void Feature::setLocation (
            Coord newLoc )
```

Setter for location.

**See also**

location

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

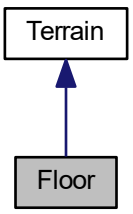
- include/feature.h
- feature.cpp

## 4.8 Floor Class Reference

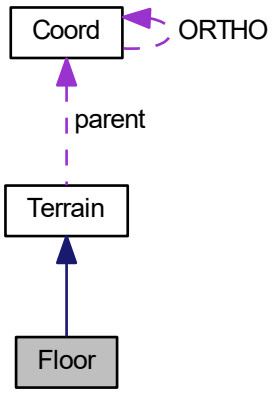
Regular dungeon floor.

```
#include <tiles.h>
```

Inheritance diagram for Floor:



Collaboration diagram for Floor:




### Additional Inherited Members

#### 4.8.1 Detailed Description

Regular dungeon floor.

Has full visibility and passability.

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

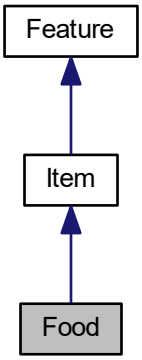
- include/tiles.h
- tiles.cpp

## 4.9 Food Class Reference

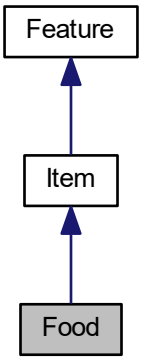
Represents food.

`#include <food.h>`

Inheritance diagram for Food:




Collaboration diagram for Food:




### Public Member Functions

- Food (Coord, Item::Context)

  *Constructs a Food instance.*
- bool activate (PlayerChar *)

  *Applies the effects derived from eating this Food.*

### Additional Inherited Members

#### 4.9.1 Detailed Description

Represents food.

### 4.9.2 Constructor & Destructor Documentation

#### 4.9.2.1 Food()

```
Food::Food (
            Coord location,
            Item::Context context )
```

Constructs a Food instance.

**Parameters**

| | | |
|---|---|---|
| in | *location* | Food location |
| in | *context* | Food context |

### 4.9.3 Member Function Documentation

#### 4.9.3.1 activate()

```
bool Food::activate (
            PlayerChar * player )
```

Applies the effects derived from eating this Food.

**Parameters**

| | |
|---|---|
| *player* | Reference to the PlayerCharacter instance |

**Returns**

A value reflecting the success of the activation operation.

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

- include/food.h
- food.cpp

## 4.10 Generator Class Reference

Light wrapper around the std library which provides various random generation utilities.

```
#include <random.h>
```

### Static Public Member Functions

- static int intFromRange (int, int)

  *Random integer from range (inclusive).*
- static double rand ()

  *Random double between 0 and 1 (inclusive).*
- static bool randBool ()

  *Random boolean.*
- static Coord randPosition (Coord, Coord)

  *Random coord in box deliniated by topleft, bottomright.*
- template<typename T >
  static void shuffle (std::vector< T > *)

  *Randomly shuffle the vector provided.*
- static int nDx (int numDice, int numFaces)

  *Rolls the designated dice and returns sum.*

## 4.10.1 Detailed Description

Light wrapper around the std library which provides various random generation utilities.

## 4.10.2 Member Function Documentation

### 4.10.2.1 intFromRange()

```
int Generator::intFromRange (
            int lower,
            int upper )  [static]
```

Random integer from range (inclusive).

### 4.10.2.2 nDx()

```
int Generator::nDx (
            int numDice,
            int numFaces )  [static]
```

Rolls the designated dice and returns sum.

### 4.10.2.3 rand()

```
double Generator::rand ( )  [static]
```

Random double between 0 and 1 (inclusive).

#### 4.10.2.4 randBool()

```
bool Generator::randBool ( )  [static]
```

Random boolean.

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

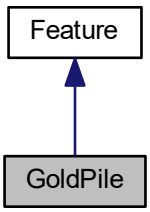
- include/random.h
- random.cpp

## 4.11 GoldPile Class Reference

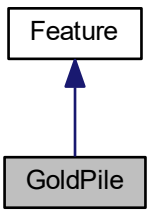
Represents a pile of gold on the ground, which can be picked up by the player to enhance their score.

```
#include <goldpile.h>
```

Inheritance diagram for GoldPile:




Collaboration diagram for GoldPile:



### Public Member Functions

- GoldPile (Coord, int)

  *Constructor of location, quantity.*
- int getQuantity ()

  *Getter for quantity.*

## 4.11.1 Detailed Description

Represents a pile of gold on the ground, which can be picked up by the player to enhance their score.

## 4.11.2 Constructor & Destructor Documentation

### 4.11.2.1 GoldPile()

```
GoldPile::GoldPile (
            Coord location,
            int quantity )
```

Constructor of location, quantity.

## 4.11.3 Member Function Documentation

### 4.11.3.1 getQuantity()

```
int GoldPile::getQuantity ( )
```

Getter for quantity.

**See also**

quantity

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

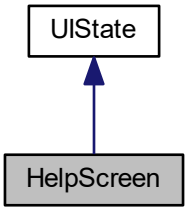
- include/goldpile.h
- goldpile.cpp

## 4.12 HelpScreen Class Reference

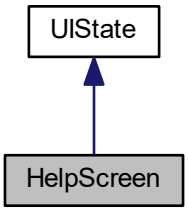
Interface state that shows the various game controls.

`#include <helpscreen.h>`

Inheritance diagram for HelpScreen:




Collaboration diagram for HelpScreen:




### Public Member Functions

- HelpScreen (PlayerChar ∗, Level ∗)
  *Constructor.*
- virtual void draw (TCODConsole ∗)
  *Render the controls.*
- virtual UIState ∗ handleInput (TCOD_key_t)
  *Handle the player input (just quitting).*

### 4.12.1 Detailed Description

Interface state that shows the various game controls.

Environment variables: input device (e.g., keyboard) and output device (e.g., monitor)

### 4.12.2 Constructor & Destructor Documentation

#### 4.12.2.1 HelpScreen()

```
HelpScreen::HelpScreen (
            PlayerChar * pc,
            Level * lvl )
```

Constructor.

### 4.12.3 Member Function Documentation

#### 4.12.3.1 draw()

```
void HelpScreen::draw (
            TCODConsole * con )  [virtual]
```

Render the controls.

Reimplemented from UIState.

#### 4.12.3.2 handleInput()

```
UIState * HelpScreen::handleInput (
            TCOD_key_t key )  [virtual]
```

Handle the player input (just quitting).

Reimplemented from UIState.

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

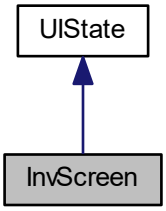
- include/helpscreen.h
- helpscreen.cpp

## 4.13 InvScreen Class Reference

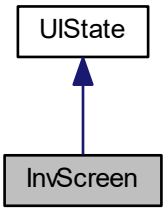
Interface state for viewing the contents of the player inventory.

```
#include <invscreen.h>
```

Inheritance diagram for InvScreen:




Collaboration diagram for InvScreen:




### Public Types

- typedef std::function< UIState *(Item *, PlayerChar *, Level *)> **transFunc**
- typedef std::function< bool(Item *)> **filtFunc**

### Public Member Functions

- InvScreen (PlayerChar *, Level *, filtFunc, transFunc, bool)

  *Constructor.*
- void draw (TCODConsole *)

  *Draw the inventory.*
- UIState * handleInput (TCOD_key_t)

  *Handle input (just the quit key).*

## 4.13.1 Detailed Description

Interface state for viewing the contents of the player inventory.

Environment variables: input device (e.g., keyboard) and output device (e.g., monitor)

### 4.13.2 Constructor & Destructor Documentation

#### 4.13.2.1 InvScreen()

```
InvScreen::InvScreen (
            PlayerChar * player,
            Level * level,
            filtFunc filter,
            transFunc trans,
            bool escapeable )
```

Constructor.

We take the playerchar and level so we can restore them once gameplay resumes. Includes filter for inventory and function for desired return state.

### 4.13.3 Member Function Documentation

#### 4.13.3.1 draw()

```
void InvScreen::draw (
            TCODConsole * con )  [virtual]
```

Draw the inventory.

Shows like-and-stackable items grouped. Makes sure to not reveal the true names of undiscovered items.

Reimplemented from UIState.

#### 4.13.3.2 handleInput()

```
UIState * InvScreen::handleInput (
            TCOD_key_t key )  [virtual]
```

Handle input (just the quit key).

Reimplemented from UIState.

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

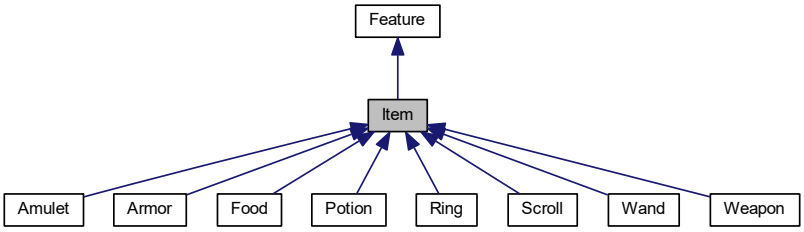
- include/invscreen.h
- invscreen.cpp

## 4.14 Item Class Reference

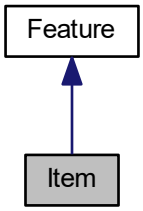
Represents a generic item.

`#include <item.h>`

Inheritance diagram for Item:




Collaboration diagram for Item:




### Public Types

- enum Context { **FLOOR**, **INVENTORY** }

  *Placement context of this Item.*

### Public Member Functions

- Item (char, Coord, Context, std::string, std::string, int, bool, bool)

  *Constructs an Item instance.*
- Item (char, Coord, Context, std::string, std::string, std::string, int, bool, bool)

  *Constructs an Item instance.*
- bool operator== (const Item &) const

  *Item equality definition.*
- bool operator< (const Item &) const

*Item 'less than' comparison definition.*

- Context getContext ()

  *Gets the context.*
- std::string getClassName ()

  *Gets the subclass name.*
- void setContext (Context)

  *Sets the context.*
- std::string getDisplayName ()

  *Gets the display name.*
- std::string getName ()

  *Gets the name.*
- int getType ()

  *Gets the type.*
- bool isIdentified ()

  *Determines if the Item is identified.*
- bool isStackable ()

  *Determines if the Item is stackable.*
- bool isThrowable ()

  *Determines if Item is throwable.*
- void setIdentified (bool)

  *Sets the identified status of this Item type.*

## Static Public Member Functions

- static std::vector< std::string > shuffleNameVector (std::vector< std::string >)

  *Returns a shuffled copy of the provided vector of names.*

## Static Public Attributes

- static const int **BASE_THROW_DMG** = 10

## Protected Attributes

- bool canStack

  *Denotes whether or not this Item can stack in the inventory.*
- bool canThrow

  *Denotes whether or not this Item can be thrown.*
- std::string className

  *Name of this Item's subclass.*
- Context context

  *Context of this Item.*
- bool cursed

  *Denotes whether or not this Item is cursed.*
- std::string name

  *Name of this Item.*
- std::string pseudoName

  *Name of the unidentified version of this Item.*
- int type

  *Type of this Item.*

### Static Protected Attributes

- static std::map< std::string, std::map< int, bool > > identified

  *Identification map of the following form: {Class Name : {Type : Status}}.*

## 4.14.1 Detailed Description

Represents a generic item.

## 4.14.2 Constructor & Destructor Documentation

### 4.14.2.1 Item() [1/2]

```
Item::Item (
            char symbol,
            Coord location,
            Item::Context context,
            std::string className,
            std::string name,
            int type,
            bool canStack,
            bool canThrow )
```

Constructs an Item instance.

**Parameters**

| | | |
|---|---|---|
| in | *symbol* | Character denoting this Item |
| in | *location* | Item location |
| in | *context* | Item context |
| in | *className* | Name of Item subclass using this constructor |
| in | *name* | Item name |
| in | *type* | Item type |
| in | *canStack* | Denotes whether or not this Item can be stacked in the inventory |
| in | *canThrow* | Denotes whether or not this Item can be thrown |

### 4.14.2.2 Item() [2/2]

```
Item::Item (
            char symbol,
            Coord location,
            Item::Context context,
            std::string className,
            std::string name,
            std::string pseudoName,
            int type,
            bool canStack,
            bool canThrow )
```

Constructs an Item instance.

**Parameters**

| | | |
|---|---|---|
| in | *symbol* | Character denoting this Item |
| in | *location* | Item location |
| in | *context* | Item context |
| in | *className* | Name of Item subclass using this constructor |
| in | *name* | Item name |
| in | *pseudoName* | Unidentified Item name |
| in | *type* | Item type |
| in | *canStack* | Denotes whether or not this Item can be stacked in the inventory |
| in | *canThrow* | Denotes whether or not this Item can be thrown |

### 4.14.3 Member Function Documentation

#### 4.14.3.1 getClassName()

```
std::string Item::getClassName ( )
```

Gets the subclass name.

**Returns**

The subclass name.

#### 4.14.3.2 getContext()

```
Item::Context Item::getContext ( )
```

Gets the context.

**Returns**

The context.

#### 4.14.3.3 getDisplayName()

```
std::string Item::getDisplayName ( )
```

Gets the display name.

**Returns**

The display name.

#### 4.14.3.4 getName()

```
std::string Item::getName ( )
```

Gets the name.

**Returns**

The name.

#### 4.14.3.5 getType()

```
int Item::getType ( )
```

Gets the type.

**Returns**

The type.

#### 4.14.3.6 isIdentified()

```
bool Item::isIdentified ( )
```

Determines if the Item is identified.

**Returns**

True if identified, False otherwise.

#### 4.14.3.7 isStackable()

```
bool Item::isStackable ( )
```

Determines if the Item is stackable.

**Returns**

True if stackable, False otherwise.

#### 4.14.3.8 isThrowable()

```
bool Item::isThrowable ( )
```

Determines if Item is throwable.

**Returns**

True if throwable, False otherwise.

#### 4.14.3.9 operator<()

```
bool Item::operator< (
            const Item & other ) const
```

Item 'less than' comparison definition.

**Parameters**

| | | |
|---|---|---|
| in | *item* | Other comparison operand |

**Returns**

True if this Item is less than the given Item, False otherwise

#### 4.14.3.10 operator==()

```
bool Item::operator== (
            const Item & other ) const
```

Item equality definition.

**Parameters**

| | | |
|---|---|---|
| in | *item* | Other equality operand |

**Returns**

True if this Item is equivalent to the given Item, False otherwise

#### 4.14.3.11 setContext()

```
void Item::setContext (
            Item::Context newContext )
```

Sets the context.

**Parameters**

| | | |
|---|---|---|
| in | *context* | New Item context |

#### 4.14.3.12 setIdentified()

```
void Item::setIdentified (
            bool newValue )
```

Sets the identified status of this Item type.

**Parameters**

| | | |
|---|---|---|
| in | *newValue* | New identified status of this Item type. |

#### 4.14.3.13 shuffleNameVector()

```
std::vector< std::string > Item::shuffleNameVector (
            std::vector< std::string > nameVector )  [static]
```

Returns a shuffled copy of the provided vector of names.

**Parameters**

| | | |
|---|---|---|
| in | *nameVector* | Vector of names |

**Returns**

The shuffled copy of the provided vector of names.

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

- include/item.h
- item.cpp

## 4.15 ItemZone Class Reference

Container for items.

```
#include <itemzone.h>
```

### Public Member Functions

- ItemZone ()
  *Constructor for empty container.*
- Item * operator[] (int)
  *Access item at index, as if ItemZone was just an array.*
- void add (Item &)
  *Add item to ItemZone, stacking if necessary.*
- bool contains (Item *)
  *Check if ItemZone contains >= 1 copies of item.*
- bool contains (const std::string &name)
  *Check if item with given name is in ItemZone.*
- std::map< char, std::vector< Item * > > & getContents ()
  *Return the contents of the zone directly.*
- bool remove (Item *)
  *Remove the given item from the zone, potentially destacking if necessary.*
- std::vector< Item * > * getItem (char)
  *Return struct corresponding to given hotkey.*
- int getSize ()
  *Return the number of distinct items.*

### 4.15.1 Detailed Description

Container for items.

**See also**

Item Tracks stackability and how it relates to capacity, provides utility functions, and tracks persistent hotkeys.

### 4.15.2 Constructor & Destructor Documentation

#### 4.15.2.1 ItemZone()

```
ItemZone::ItemZone ( )
```

Constructor for empty container.

### 4.15.3 Member Function Documentation

#### 4.15.3.1 add()

```
void ItemZone::add (
            Item & item )
```

Add item to ItemZone, stacking if necessary.

#### 4.15.3.2 contains() [1/2]

```
bool ItemZone::contains (
            Item * item )
```

Check if ItemZone contains >= 1 copies of item.

#### 4.15.3.3 contains() [2/2]

```
bool ItemZone::contains (
            const std::string & name )
```

Check if item with given name is in ItemZone.

#### 4.15.3.4 getContents()

```
std::map< char, std::vector< Item * > > & ItemZone::getContents ( )
```

Return the contents of the zone directly.

#### 4.15.3.5 getItem()

```
std::vector< Item * > * ItemZone::getItem (
            char symbol )
```

Return struct corresponding to given hotkey.

#### 4.15.3.6 getSize()

```
int ItemZone::getSize ( )
```

Return the number of distinct items.

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

- include/itemzone.h
- itemzone.cpp

## 4.16 Level Class Reference

### Public Member Functions

- **Level** (int, PlayerChar *)
- Terrain & **tileAt** (Coord)
- Terrain & **operator[ ]** (Coord)
- void **generate** ()
- bool **contains** (Coord)
- int **getDepth** ()
- PlayerChar * **getPlayer** ()
- void registerMob (Mob *)

  *Adds a mob to the mobs known by the level.*
- void removeMob (Mob *)

  *Removes a mob.*
- std::vector< Mob * > getMobs ()

  *Gets all the mobs on the level.*
- Mob * popTurnClock ()

  *Returns the mob who's turn to act is next.*
- void pushMob (Mob *, int)

  *Moves a mob back in the turn clock equal to the amount specified.*
- std::vector< Coord > bfsDiag (Coord, Coord)

  *Performs BFS to get the shortest path from the starting coordinate to the end coordinate.*
- std::vector< Coord > bfsPerp (Coord, Coord)

  *Performs BFS to get the shortest path from the starting coordinate to the end coordinate.*
- std::vector< Coord > getAdjPassable (Coord)

  *Gets the coordinates to which one can move to from a given source (3x3 box)*
- Coord throwLocation (Coord, Coord)

  *Given a start and a delta direction, returns the position of where something thrown would land.*
- std::vector< Room > & getRooms ()

  *Gets the rooms.*

- std::vector< Feature * > & getFeatures ()

  *Gets the features.*
- void removeFeature (Feature *)

  *Removes a feature.*
- void addFeature (Feature *)

  *Adds a feature.*
- Mob * monsterAt (Coord)

  *Returns the monster that is at the location.*
- bool canSee (Coord, Coord)

  *Determines ability to see each other.*
- std::vector< Coord > getNearestGold (Coord)

  *Gets the path to the nearest gold.*
- void randomizePlayerLocation ()

  *Place the player at a random empty position.*

### Static Public Member Functions

- static Coord **getSize** ()

## 4.16.1 Member Function Documentation

### 4.16.1.1 addFeature()

```
void Level::addFeature (
            Feature * feat )
```

Adds a feature.

**Parameters**

| | |
|---|---|
| *Feature* | The feature to add |

### 4.16.1.2 bfsDiag()

```
std::vector< Coord > Level::bfsDiag (
            Coord start,
            Coord end )
```

Performs BFS to get the shortest path from the starting coordinate to the end coordinate.

As opposed to bfsPerp, this algorithm is allowed to move in any of the 8 directions.

**Parameters**

| | |
|---|---|
| *Coord* | Starting point |
| *Coord* | Ending point |

**Returns**

A vector of the absolute coordinates of the shortest path, including start and end, starting at the start and moving forwards one unit vector at a time.

**See also**

bfsPerp

#### 4.16.1.3 bfsPerp()

```
std::vector< Coord > Level::bfsPerp (
            Coord start,
            Coord end )
```

Performs BFS to get the shortest path from the starting coordinate to the end coordinate.

As opposed to bfsDiag, this algorithm is allowed to move only in the 4 cardinal direcitons.

**Parameters**

| | |
|---|---|
| *Coord* | Starting point |
| *Coord* | Ending point |

**Returns**

A vector of the absolute coordinates of the shortest path, including start and end, starting at the start and moving forwards one unit vector at a time.

**See also**

bfsDiag

#### 4.16.1.4 canSee()

```
bool Level::canSee (
            Coord a,
            Coord b )
```

Determines ability to see each other.

**Parameters**

| | | |
|---|---|---|
| in | *Coord* | A |
| in | *Coord* | B |

**Returns**

True if able to see, False otherwise.

#### 4.16.1.5 getAdjPassable()

```
std::vector< Coord > Level::getAdjPassable (
            Coord ori )
```

Gets the coordinates to which one can move to from a given source (3x3 box)

**Parameters**

| | |
|---|---|
| *Coord* | Coordinate to check from |

**Returns**

A vector of coordinates onto which you can move.

#### 4.16.1.6 getFeatures()

```
std::vector< Feature * > & Level::getFeatures ( )
```

Gets the features.

**Returns**

The features.

#### 4.16.1.7 getMobs()

```
std::vector< Mob * > Level::getMobs ( )
```

Gets all the mobs on the level.

**Returns**

The mobs.

#### 4.16.1.8 getNearestGold()

```
std::vector< Coord > Level::getNearestGold (
            Coord ori )
```

Gets the path to the nearest gold.

**Parameters**

| | | |
|---|---|---|
| in | *Coord* | Origin to search from |

**Returns**

The path to the nearest gold. NULL if there is no gold to find.

#### 4.16.1.9 getRooms()

```
std::vector< Room > & Level::getRooms ( )
```

Gets the rooms.

**Returns**

The rooms.

#### 4.16.1.10 monsterAt()

```
Mob * Level::monsterAt (
            Coord s )
```

Returns the monster that is at the location.

**Parameters**

| | | |
|---|---|---|
| in | *Coord* | The location to get the monster from |

**Returns**

Returns the pointer to a monster if there is one at the specified location, NULL otherwise.

#### 4.16.1.11 popTurnClock()

```
Mob * Level::popTurnClock ( )
```

Returns the mob who's turn to act is next.

**Returns**

A mob

#### 4.16.1.12 pushMob()

```
void Level::pushMob (
            Mob * which,
            int delay )
```

Moves a mob back in the turn clock equal to the amount specified.

**Parameters**

| | |
|---|---|
| *Mob*∗ | Which mob |
| *int* | How far to push back in the clock cycle |

#### 4.16.1.13 registerMob()

```
void Level::registerMob (
            Mob * mob )
```

Adds a mob to the mobs known by the level.

**Parameters**

| | |
|---|---|
| *Mob*∗ | Pointer to the mob that is to be added |

#### 4.16.1.14 removeFeature()

```
void Level::removeFeature (
            Feature * feat )
```

Removes a feature.

**Parameters**

| | |
|---|---|
| *Feature* | The feature to remove |

#### 4.16.1.15 removeMob()

```
void Level::removeMob (
            Mob * mob )
```

Removes a mob.

**Parameters**

| | |
|---|---|
| *Mob*∗ | Pointer to the mob that is to be removed |

#### 4.16.1.16 throwLocation()

```
Coord Level::throwLocation (
            Coord start,
            Coord dir )
```

Given a start and a delta direction, returns the position of where something thrown would land.

**Parameters**

| | |
|---|---|
| *Coord* | Where the object is being thrown from |
| *Coord* | The direction in which it is being thrown (Must be a unit vector!) |

**Returns**

Final location

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

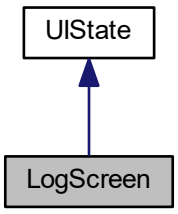
- include/level.h
- level.cpp

## 4.17 LogScreen Class Reference

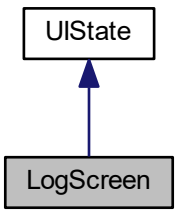
Controls the display of the event log.

`#include <logscreen.h>`

Inheritance diagram for LogScreen:




Collaboration diagram for LogScreen:



### Public Member Functions

- LogScreen (PlayerChar *, Level *)

  *Constructor, takes info so we can return to regular gameplay with it later.*
- virtual UIState * handleInput (TCOD_key_t)

  *Allow the player to leave the log screen.*
- virtual void draw (TCODConsole *)

  *Render the previous log messages, up is more recent.*

## 4.17.1 Detailed Description

Controls the display of the event log.

Environment variables: input device (e.g., keyboard) and output device (e.g., monitor)

## 4.17.2 Member Function Documentation

### 4.17.2.1 draw()

```
void LogScreen::draw (
            TCODConsole * con )  [virtual]
```

Render the previous log messages, up is more recent.

Reimplemented from UIState.

### 4.17.2.2 handleInput()

```
UIState * LogScreen::handleInput (
            TCOD_key_t key )  [virtual]
```

Allow the player to leave the log screen.

Reimplemented from UIState.

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

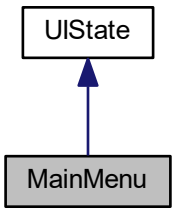
- include/logscreen.h
- logscreen.cpp

## 4.18 MainMenu Class Reference

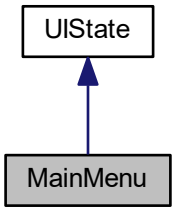
Start screen of the game.

`#include <mainmenu.h>`

Inheritance diagram for MainMenu:




Collaboration diagram for MainMenu:




### Public Member Functions

- MainMenu ()
  *Constructor.*
- virtual void draw (TCODConsole ∗)
  *Render the splash art, name prompt.*
- virtual UIState ∗ handleInput (TCOD_key_t)
  *Handle input (start game, edit name buffer).*

### 4.18.1 Detailed Description

Start screen of the game.

Should include splash art, and name prompt.

Environment variables: input device (e.g., keyboard) and output device (e.g., monitor)

### 4.18.2 Constructor & Destructor Documentation

#### 4.18.2.1 MainMenu()

```
MainMenu::MainMenu ( )
```

Constructor.

### 4.18.3 Member Function Documentation

#### 4.18.3.1 draw()

```
void MainMenu::draw (
            TCODConsole * con )  [virtual]
```

Render the splash art, name prompt.

Reimplemented from UIState.

#### 4.18.3.2 handleInput()

```
UIState * MainMenu::handleInput (
            TCOD_key_t key )  [virtual]
```

Handle input (start game, edit name buffer).

Reimplemented from UIState.

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

- include/mainmenu.h
- mainmenu.cpp

## 4.19 MasterController Class Reference

Controls the top level flow flow of the application and main game loop.

```
#include <mastercontroller.h>
```

### Public Member Functions

- MasterController ()

  *All game logic is inside, so no params needed for constructor.*
- void run ()

  *Main game loop.*

### 4.19.1 Detailed Description

Controls the top level flow flow of the application and main game loop.

Called directly from main.

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

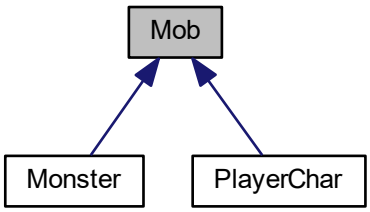
- include/mastercontroller.h
- mastercontroller.cpp

## 4.20 Mob Class Reference

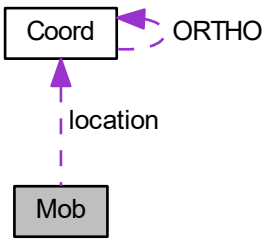
Models a creature in the dungeon, could be the player or a monster.

`#include <mob.h>`

Inheritance diagram for Mob:




Collaboration diagram for Mob:



## Public Member Functions

- Mob (char, Coord)

  *Constructor used by monsters.*
- Mob (char, Coord, std::string, int armor, int exp, int mobHP, int level)

  *Constructor.*
- virtual int **calculateDamage** ()=0
- void changeArmor (int)

  *Setter for armor.*
- int getArmor ()

  *Getter for armor.*
- int getExperience ()

  *Getter for XP.*
- int getHP ()

  *Getter for HP.*
- int getMaxHP ()

  *Getter for max HP.*
- int getLevel ()

  *Getter for mob level.*
- Coord & getLocation ()

  *Getter for mob location.*
- std::string getName ()

  *Getter for name.*
- char getSymbol ()

  *Getter for symbol.*
- virtual void hit (int)

  *Called by other entities when they deal damage.*
- bool isDead ()

  *Determines if this mob is dead.*
- void moveLocation (Coord)

  *Add current location and param together.*
- bool setCurrentHP (int)

  *Setter for current HP.*
- void setLocation (Coord)

  *Setter for location.*
- void setMaxHP (int)

  *Setter for max hitpoints.*
- virtual int turn (Level *)

  *Mob enacts its turn on the level, returns number of ticks it took.*
- virtual ~Mob ()

  *Destructor.*

## Static Public Member Functions

- static int **diceSum** (int, int)

### Protected Attributes

- int armor

  *More armor makes it more difficult for enemies to hit the mob.*
- int currentHP

  *More hitpoints indicates the mob is healthier.*
- bool dead

  *Indicates whether or not this mob is dead.*
- int exp

  *More exp indicates the mob is closer to leveling up.*
- int level

  *Higher level characters are more powerful.*
- Coord location

  *Current location within the level.*
- int maxHP

  *Maximum number of hitpoints.*
- std::string name

  *Name of the mob.*

## 4.20.1 Detailed Description

Models a creature in the dungeon, could be the player or a monster.

## 4.20.2 Constructor & Destructor Documentation

### 4.20.2.1 Mob() [1/2]

```
Mob::Mob (
            char symbol,
            Coord location )
```

Constructor used by monsters.

### 4.20.2.2 Mob() [2/2]

```
Mob::Mob (
            char symbol,
            Coord location,
            std::string name,
            int armor,
            int exp,
            int mobHP,
            int level )
```

Constructor.

**See also**

armor
exp
maxHP
level

#### 4.20.2.3 ~Mob()

```
Mob::~Mob ( )  [virtual]
```

Destructor.

### 4.20.3 Member Function Documentation

#### 4.20.3.1 changeArmor()

```
void Mob::changeArmor (
            int  )
```

Setter for armor.

**See also**

armor

#### 4.20.3.2 getArmor()

```
int Mob::getArmor ( )
```

Getter for armor.

**See also**

armor

#### 4.20.3.3 getExperience()

```
int Mob::getExperience ( )
```

Getter for XP.

**See also**

exp

#### 4.20.3.4 getHP()

```
int Mob::getHP ( )
```

Getter for HP.

**See also**

currentHP

#### 4.20.3.5 getLevel()

```
int Mob::getLevel ( )
```

Getter for mob level.

**See also**

level

#### 4.20.3.6 getLocation()

```
Coord & Mob::getLocation ( )
```

Getter for mob location.

Can be edited because it returns a reference

**See also**

location

#### 4.20.3.7 getMaxHP()

```
int Mob::getMaxHP ( )
```

Getter for max HP.

**See also**

maxHP

#### 4.20.3.8 getName()

```
std::string Mob::getName ( )
```

Getter for name.

**See also**

name

#### 4.20.3.9 getSymbol()

```
char Mob::getSymbol ( )
```

Getter for symbol.

**See also**

symbol

#### 4.20.3.10 hit()

```
void Mob::hit (
            int damage )  [virtual]
```

Called by other entities when they deal damage.

**See also**

currentHP

Reimplemented in Monster.

#### 4.20.3.11 isDead()

```
bool Mob::isDead ( )
```

Determines if this mob is dead.

**Returns**

True if this mob is dead, false otherwise

#### 4.20.3.12 moveLocation()

```
void Mob::moveLocation (
            Coord location )
```

Add current location and param together.

**See also**

location

#### 4.20.3.13 setCurrentHP()

```
bool Mob::setCurrentHP (
            int currentHP )
```

Setter for current HP.

**See also**

currentHP

#### 4.20.3.14 setLocation()

```
void Mob::setLocation (
            Coord location )
```

Setter for location.

**See also**

location

#### 4.20.3.15 setMaxHP()

```
void Mob::setMaxHP (
            int maxHP )
```

Setter for max hitpoints.

**See also**

maxHP

### 4.20.4 Member Data Documentation

#### 4.20.4.1 armor

```
int Mob::armor  [protected]
```

More armor makes it more difficult for enemies to hit the mob.

#### 4.20.4.2 exp

```
int Mob::exp  [protected]
```

More exp indicates the mob is closer to leveling up.

#### 4.20.4.3 level

```
int Mob::level  [protected]
```

Higher level characters are more powerful.

#### 4.20.4.4 location

```
Coord Mob::location  [protected]
```

Current location within the level.

#### 4.20.4.5 maxHP

```
int Mob::maxHP  [protected]
```

Maximum number of hitpoints.

#### 4.20.4.6 name

```
std::string Mob::name  [protected]
```

Name of the mob.

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

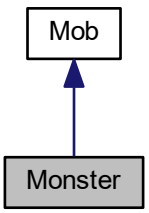
- include/mob.h
- mob.cpp

## 4.21 Monster Class Reference

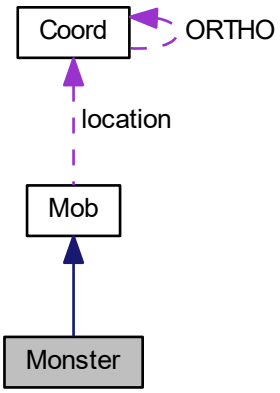
Models a monster in the dungeon.

```
#include <monster.h>
```

Inheritance diagram for Monster:



Collaboration diagram for Monster:




## Public Types

- enum Behaviour {
  **AGGRESSIVE**, **FLYING**, **REGENERATIVE**, **GREEDY**,
  **INVISIBLE** }

  *Monster flags denoting behavioural patterns.*

## Public Member Functions

- Monster (char, Coord)

  *Constructs a Monster instance of the given symbol type.*
- void aggrevate ()

  *Aggrevates this monster to attack the player.*
- virtual void hit (int dmgAmount)

  *Override mob implementation to aggrevate monster.*
- void attack (Level ∗)

  *Attempts to attack a nearby Player Character.*
- int calculateDamage ()

  *Calculates the damage of this Monster.*
- int calculateHitChance (PlayerChar ∗)

  *Calculates the hit chance of this Monster.*
- int getCarryChance ()

  *Gets the carry chance of this Monster.*
- bool **hasFlag** (Behaviour)
- bool isAwake ()

  *Gets the Monster awake state.*
- virtual int turn (Level ∗)

  *Performs the actions that make up a Monster's turn.*

### Static Public Member Functions

- static std::vector< char > getSymbolsForLevel (int)

  *Gets the valid Monster symbols based on the current dungeon depth.*
- static std::vector< char > getSymbolsForTreasure (int)

  *Gets the valid Monster symbols for a treasure room based on the current dungeon depth.*

### Additional Inherited Members

## 4.21.1 Detailed Description

Models a monster in the dungeon.

## 4.21.2 Constructor & Destructor Documentation

### 4.21.2.1 Monster()

```
Monster::Monster (
            char symbol,
            Coord location )
```

Constructs a Monster instance of the given symbol type.

**Parameters**

| | | |
|---|---|---|
| in | *symbol* | Monster symbol |
| in | *location* | Monster location |

**Exceptions**

| | |
|---|---|
| *e* | Illegal argument exception is thrown if an unknown symbol is given |

## 4.21.3 Member Function Documentation

### 4.21.3.1 attack()

```
void Monster::attack (
            Level * level )
```

Attempts to attack a nearby Player Character.

**Parameters**

| | |
|---|---|
| *level* | Reference to the current Level |

#### 4.21.3.2 calculateDamage()

```
int Monster::calculateDamage ( )  [virtual]
```

Calculates the damage of this Monster.

**Returns**

The computed damage.

Implements Mob.

#### 4.21.3.3 calculateHitChance()

```
int Monster::calculateHitChance (
            PlayerChar * player )
```

Calculates the hit chance of this Monster.

**Parameters**

| | |
|---|---|
| *Reference* | to the player character |

**Returns**

The computed hit chance.

#### 4.21.3.4 getCarryChance()

```
int Monster::getCarryChance ( )
```

Gets the carry chance of this Monster.

**Returns**

The carry chance of this Monster.

#### 4.21.3.5 getSymbolsForLevel()

```
std::vector< char > Monster::getSymbolsForLevel (
            int depth )  [static]
```

Gets the valid Monster symbols based on the current dungeon depth.

**Parameters**

| | | |
|---|---|---|
| in | *depth* | Current dungeon depth |

**Returns**

Vector of valid Monster symbols.

#### 4.21.3.6 getSymbolsForTreasure()

```
std::vector< char > Monster::getSymbolsForTreasure (
            int depth )  [static]
```

Gets the valid Monster symbols for a treasure room based on the current dungeon depth.

**Parameters**

| | | |
|---|---|---|
| in | *depth* | Current dungeon depth |

**Returns**

Vector of valid Monster symbols.

#### 4.21.3.7 hit()

```
void Monster::hit (
            int dmgAmount )  [virtual]
```

Override mob implementation to aggrevate monster.

**See also**

aggrevate

Reimplemented from Mob.

#### 4.21.3.8 isAwake()

```
bool Monster::isAwake ( )
```

Gets the Monster awake state.

**Returns**

True if the Monster is awake, False otherwise.

#### 4.21.3.9 turn()

```
int Monster::turn (
            Level * level )  [virtual]
```

Performs the actions that make up a Monster's turn.

**Parameters**

| | |
|---|---|
| *level* | Reference to the current Level |

**Returns**

Value denoting the consequential turn delay.

Reimplemented from Mob.

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

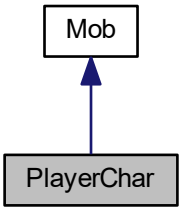
- include/monster.h
- monster.cpp

## 4.22 PlayerChar Class Reference

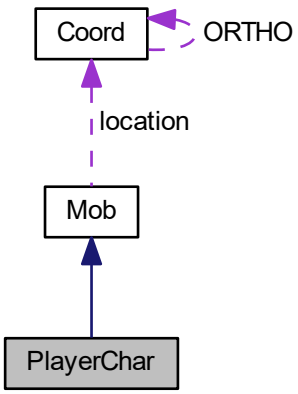
Models the user-controlled player character.

`#include <playerchar.h>`

Inheritance diagram for PlayerChar:




Collaboration diagram for PlayerChar:



## Public Member Functions

- PlayerChar (Coord, std::string)

  *Constructs a PlayerChar instance.*
- void activateItem (Item ∗)

  *Activates the provided item.*
- void addExp (int)

  *Adds the given experience to the PlayerChar.*
- void appendLog (std::string)

  *Appends the given entry to the log.*
- void attack (Monster ∗)

  *Attacks the given Mob.*
- int calculateDamage ()

  *Calculates the damage the PlayerChar will inflict.*
- int calculateHitChance (Monster ∗)

  *Calculates the hit chance of the PlayerChar.*
- void changeCurrentHP (int)

  *Increases the current HP of the PlayerChar by the passed parameter.*
- void changeFoodLife (int)

  *Increases the food life of the PlayerChar by the passed parameter.*
- void collectGold (GoldPile ∗)

  *Adds the gold contained in the given GoldPile to the PlayerChar's gold total.*
- bool dropItem (Item ∗, Level ∗)

  *Attemps to drop the given Item.*
- void eat (Food ∗)

  *Attempts to eat the given Food.*
- void equipArmor (Armor ∗)

  *Attempts to equip the given Armor.*
- void equipRingLeft (Ring ∗)

  *Attempts to equip the given Ring on the PlayerChar's left hand.*
- void equipRingRight (Ring ∗)

  *Attempts to equip the given Ring on the PlayerChar's right hand.*
- void equipWeapon (Weapon ∗)

  *Attempts to equip the given Weapon.*
- int getDexterity ()

  *Gets the PlayerChar's dexterity.*
- int getFoodLife ()

  *Gets the PlayerChar's food life.*
- int getGold ()

  *Gets the PlayerChar's gold total.*
- ItemZone & getInventory ()

  *Gets the PlayerChar's inventory.*
- int getStrength ()

  *Gets the PlayerChar's strength.*
- int getMaxStrength ()

  *Gets the PlayerChar's maximum strength.*
- int getSightRadius ()

  *Gets the PlayerChar's sight radius.*
- bool hasAmulet ()

  *Determines whether or not PlayerChar has the Amulet of Yendor.*
- void move (Coord)

*Relocates the PlayerChar and updates the food life.*

- void pickupItem (Item *)

  *Attempts to place the provided Item in the PlayerChar's inventory.*
- void quaff (Potion *, Mob *)

  *Attempts to apply the effects of the provided Potion to the given Mob.*
- void read (Scroll *, Level *)

  *Attempts to read the given Scroll.*
- bool removeArmor ()

  *Attempts to remove the PlayerChar's equipped Armor.*
- bool removeRingLeft ()

  *Attempts to remove the PlayerChar's equipped left Ring.*
- bool removeRingRight ()

  *Attempts to remove the PlayerChar's equipped right Ring.*
- bool removeWeapon ()

  *Attempts to remove the PlayerChar's equipped Weapon.*
- void setDexterity (int)

  *Sets the PlayerChar's dexterity.*
- void setFoodLife (int)

  *Sets the food life of the PlayerChar.*
- bool throwItem (Item *)

  *Attempts to throw the given Item.*
- void wait ()

  *Updates the PlayerChar's food life during a wait action.*
- bool zap (Wand *, Level *)

  *Attempts to spend a charge of the provided Wand.*
- void updateHealthRegen ()

  *Updates the PlayerChar's health according to i.*
- std::vector< std::string > & getLog ()

  *Gets the PlayerChar's log.*

## Additional Inherited Members

### 4.22.1 Detailed Description

Models the user-controlled player character.

### 4.22.2 Constructor & Destructor Documentation

#### 4.22.2.1 PlayerChar()

```
PlayerChar::PlayerChar (
            Coord location,
            std::string name )
```

Constructs a PlayerChar instance.

**Parameters**

| | | |
|---|---|---|
| in | *location* | PlayerChar location |
| in | *name* | PlayerChar name |

### 4.22.3 Member Function Documentation

#### 4.22.3.1 activateItem()

```
void PlayerChar::activateItem (
            Item *  )
```

Activates the provided item.

**Parameters**

| | |
|---|---|
| *item* | Item to be activated |

#### 4.22.3.2 addExp()

```
void PlayerChar::addExp (
            int exp )
```

Adds the given experience to the PlayerChar.

**Parameters**

| | |
|---|---|
| *exp* | Experience to be added |

#### 4.22.3.3 appendLog()

```
void PlayerChar::appendLog (
            std::string entry )
```

Appends the given entry to the log.

**Parameters**

| | | |
|---|---|---|
| in | *entry* | Entry to be appended to the log. |

#### 4.22.3.4 attack()

```
void PlayerChar::attack (
            Monster * monster )
```

Attacks the given Mob.

**Parameters**

| | |
|---|---|
| *monster* | Monster to be attacked. |

#### 4.22.3.5 calculateDamage()

```
int PlayerChar::calculateDamage ( )  [virtual]
```

Calculates the damage the PlayerChar will inflict.

**Returns**

The damage to be inflicted.

Implements Mob.

#### 4.22.3.6 calculateHitChance()

```
int PlayerChar::calculateHitChance (
            Monster * monster )
```

Calculates the hit chance of the PlayerChar.

**Parameters**

| | |
|---|---|
| *monster* | Moster to be hit |

**Returns**

The chance the PlayerChar will hit their target.

#### 4.22.3.7 changeCurrentHP()

```
void PlayerChar::changeCurrentHP (
            int amount )
```

Increases the current HP of the PlayerChar by the passed parameter.

**Parameters**

| | |
|---|---|
| *amount* | Amount to change the current HP. |

#### 4.22.3.8 changeFoodLife()

```
void PlayerChar::changeFoodLife (
            int amount )
```

Increases the food life of the PlayerChar by the passed parameter.

**Parameters**

| | |
|---|---|
| *amount* | Amount to change the food life. |

#### 4.22.3.9 collectGold()

```
void PlayerChar::collectGold (
            GoldPile * goldpile )
```

Adds the gold contained in the given GoldPile to the PlayerChar's gold total.

**Parameters**

| | |
|---|---|
| *goldPile* | GoldPile to be harvested. |

#### 4.22.3.10 dropItem()

```
bool PlayerChar::dropItem (
            Item * item,
            Level * level )
```

Attemps to drop the given Item.

**Parameters**

| | |
|---|---|
| *item* | Item to be dropped |
| *level* | Reference to the current Level |

**Returns**

True if the Item was successfully dropped, False otherwise.

#### 4.22.3.11 eat()

```
void PlayerChar::eat (
            Food * food )
```

Attempts to eat the given Food.

**Parameters**

| | |
|---|---|
| *food* | Food to be eaten. |

#### 4.22.3.12 equipArmor()

```
void PlayerChar::equipArmor (
            Armor * armor )
```

Attempts to equip the given Armor.

**Parameters**

| | |
|---|---|
| *armor* | Armor to be equipped. |

#### 4.22.3.13 equipRingLeft()

```
void PlayerChar::equipRingLeft (
            Ring * ring )
```

Attempts to equip the given Ring on the PlayerChar's left hand.

**Parameters**

| | |
|---|---|
| *ring* | Ring to be equipped. |

#### 4.22.3.14 equipRingRight()

```
void PlayerChar::equipRingRight (
            Ring * ring )
```

Attempts to equip the given Ring on the PlayerChar's right hand.

**Parameters**

| | |
|---|---|
| *ring* | Ring to be equipped. |

#### 4.22.3.15 equipWeapon()

```
void PlayerChar::equipWeapon (
            Weapon * weapon )
```

Attempts to equip the given Weapon.

**Parameters**

| | |
|---|---|
| *weapon* | Weapon to be equipped. |

#### 4.22.3.16 getDexterity()

```
int PlayerChar::getDexterity ( )
```

Gets the PlayerChar's dexterity.

**Returns**

The PlayerChar's dexterity.

#### 4.22.3.17 getFoodLife()

```
int PlayerChar::getFoodLife ( )
```

Gets the PlayerChar's food life.

**Returns**

The PlayerChar's food life.

#### 4.22.3.18 getGold()

```
int PlayerChar::getGold ( )
```

Gets the PlayerChar's gold total.

**Returns**

The PlayerChar's gold total.

#### 4.22.3.19 getInventory()

```
ItemZone & PlayerChar::getInventory ( )
```

Gets the PlayerChar's inventory.

**Returns**

The PlayerChar's inventory.

#### 4.22.3.20 getLog()

```
std::vector< std::string > & PlayerChar::getLog ( )
```

Gets the PlayerChar's log.

**Returns**

The PlayerChar's log.

#### 4.22.3.21 getMaxStrength()

```
int PlayerChar::getMaxStrength ( )
```

Gets the PlayerChar's maximum strength.

**Returns**

The PlayerChar's maximum strength.

#### 4.22.3.22 getSightRadius()

```
int PlayerChar::getSightRadius ( )
```

Gets the PlayerChar's sight radius.

**Returns**

The PlayerChar's sight radius.

#### 4.22.3.23 getStrength()

```
int PlayerChar::getStrength ( )
```

Gets the PlayerChar's strength.

**Returns**

The PlayerChar's strength.

#### 4.22.3.24 hasAmulet()

```
bool PlayerChar::hasAmulet ( )
```

Determines whether or not PlayerChar has the Amulet of Yendor.

**Returns**

True if PlayerChar has the Amulet, False otherwise.

#### 4.22.3.25 move()

```
void PlayerChar::move (
            Coord location )
```

Relocates the PlayerChar and updates the food life.

**Parameters**

| | |
|---|---|
| *location* | New PlayerChar location |

#### 4.22.3.26 pickupItem()

```
void PlayerChar::pickupItem (
            Item * item )
```

Attempts to place the provided Item in the PlayerChar's inventory.

**Parameters**

| | |
|---|---|
| *item* | Item to be inserted into the PlayerChar's inventory. |

#### 4.22.3.27 quaff()

```
void PlayerChar::quaff (
            Potion * potion,
            Mob * mob )
```

Attempts to apply the effects of the provided Potion to the given Mob.

**Parameters**

| | |
|---|---|
| *potion* | Potion to be quaffed |
| *mob* | Mob to quaff the Potion |

#### 4.22.3.28 read()

```
void PlayerChar::read (
            Scroll * scroll,
            Level * level )
```

Attempts to read the given Scroll.

**Parameters**

| | |
|---|---|
| *scroll* | Scroll to be read |
| *level* | Reference to the current Level |

#### 4.22.3.29 removeArmor()

```
bool PlayerChar::removeArmor ( )
```

Attempts to remove the PlayerChar's equipped Armor.

**Returns**

True if the operation was successful, False otherwise.

#### 4.22.3.30 removeRingLeft()

```
bool PlayerChar::removeRingLeft ( )
```

Attempts to remove the PlayerChar's equipped left Ring.

**Returns**

True if the operation was successful, False otherwise.

#### 4.22.3.31 removeRingRight()

```
bool PlayerChar::removeRingRight ( )
```

Attempts to remove the PlayerChar's equipped right Ring.

**Returns**

True if the operation was successful, False otherwise.

#### 4.22.3.32 removeWeapon()

```
bool PlayerChar::removeWeapon ( )
```

Attempts to remove the PlayerChar's equipped Weapon.

**Returns**

True if the operation was successful, False otherwise.

#### 4.22.3.33 setDexterity()

```
void PlayerChar::setDexterity (
            int dexterity )
```

Sets the PlayerChar's dexterity.

**Parameters**

| | |
|---|---|
| *dexterity* | The PlayerChar's new dexterity |

#### 4.22.3.34 setFoodLife()

```
void PlayerChar::setFoodLife (
            int foodLife )
```

Sets the food life of the PlayerChar.

**Parameters**

| | |
|---|---|
| *foodLife* | The new food life of the PlayerChar |

#### 4.22.3.35 throwItem()

```
bool PlayerChar::throwItem (
            Item * item )
```

Attempts to throw the given Item.

#### Returns

True if the Item was thrown, False otherwise.

#### 4.22.3.36 zap()

```
bool PlayerChar::zap (
            Wand * wand,
            Level * level )
```

Attempts to spend a charge of the provided Wand.

#### Parameters

| | |
|---|---|
| *wand* | Wand to be used |
| *level* | Reference to the current Level |

#### Returns

True if the operation was successful, False otherwise.

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

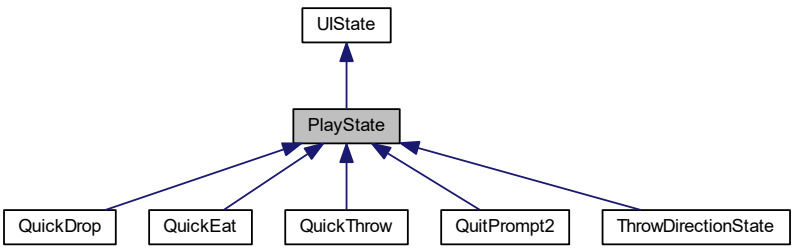
- include/playerchar.h
- playerchar.cpp

## 4.23 PlayState Class Reference

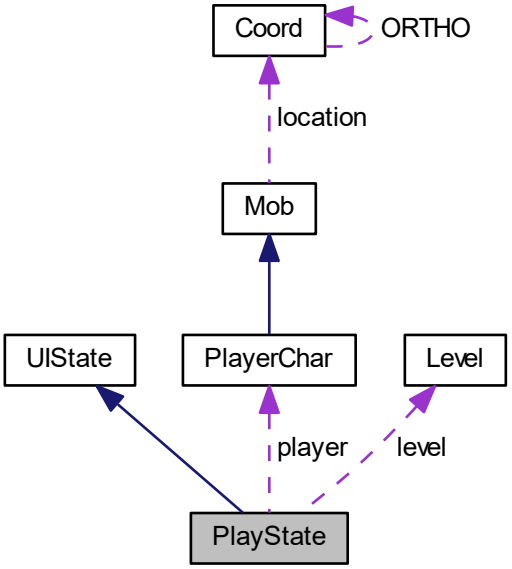
Primary interface state, showing level, player, monsters, etc.

```
#include <playstate.h>
```

Inheritance diagram for PlayState:



Collaboration diagram for PlayState:




### Public Member Functions

- PlayState (PlayerChar ∗, Level ∗)

  *Constructor.*
- virtual void draw (TCODConsole ∗)

  *Render, drawing (in this order), ui, tiles, features, mobs.*
- virtual UIState ∗ handleInput (TCOD_key_t)

  *Handle the various controls.*
- virtual ∼PlayState ()

  *Delete internal components.*

### Protected Attributes

- PlayerChar ∗ player

  *reference to player character.*
- Level ∗ level

  *Reference to current dungeon level.*

### Static Protected Attributes

- static const int **PROMPTX** = 0
- static const int **PROMPTY** = 1

### 4.23.1 Detailed Description

Primary interface state, showing level, player, monsters, etc.

### 4.23.2 Constructor & Destructor Documentation

#### 4.23.2.1 PlayState()

```
PlayState::PlayState (
            PlayerChar * play,
            Level * lvl )
```

Constructor.

#### 4.23.2.2 ~PlayState()

```
PlayState::~PlayState ( )  [virtual]
```

Delete internal components.

### 4.23.3 Member Function Documentation

#### 4.23.3.1 handleInput()

```
UIState * PlayState::handleInput (
            TCOD_key_t key )  [virtual]
```

Handle the various controls.

Reimplemented from UIState.

Reimplemented in ThrowDirectionState, QuickEat, QuickThrow, QuickDrop, and QuitPrompt2.

### 4.23.4 Member Data Documentation

#### 4.23.4.1 level

```
Level* PlayState::level  [protected]
```

Reference to current dungeon level.

#### 4.23.4.2 player

```
PlayerChar* PlayState::player  [protected]
```

reference to player character.

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

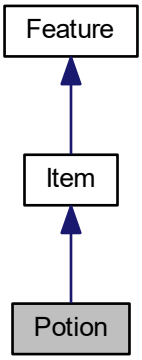
- include/playstate.h
- playstate.cpp

## 4.24 Potion Class Reference

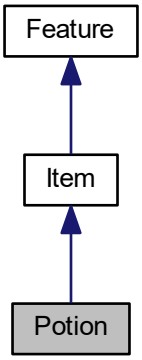
Represents potions.

`#include <potion.h>`

Inheritance diagram for Potion:




Collaboration diagram for Potion:




### Public Member Functions

- Potion (Coord)

  *Constructs a Potion instance with a random type.*
- Potion (Coord, Item::Context, int)

  *Constructs a Potion instance.*
- bool activate (Mob *)

  *Applies the effects derived from quaffing this Potion.*

### Additional Inherited Members

## 4.24.1 Detailed Description

Represents potions.

## 4.24.2 Constructor & Destructor Documentation

### 4.24.2.1 Potion() [1/2]

```
Potion::Potion (
            Coord location )
```

Constructs a Potion instance with a random type.

**Parameters**

| | | |
|---|---|---|
| in | *location* | Potion location |

### 4.24.2.2 Potion() [2/2]

```
Potion::Potion (
            Coord location,
            Item::Context context,
            int type )
```

Constructs a Potion instance.

**Parameters**

| | | |
|---|---|---|
| in | *location* | Potion location |
| in | *context* | Potion context |
| in | *type* | Potion type |

## 4.24.3 Member Function Documentation

### 4.24.3.1 activate()

```
bool Potion::activate (
            Mob * mob )
```

Applies the effects derived from quaffing this Potion.

**Parameters**

| | |
|---|---|
| *mob* | Reference to the Mob instance |

**Returns**

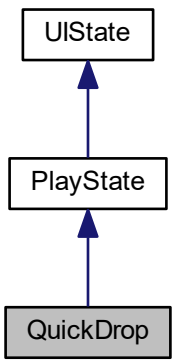
A value reflecting the success of the activation operation.

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

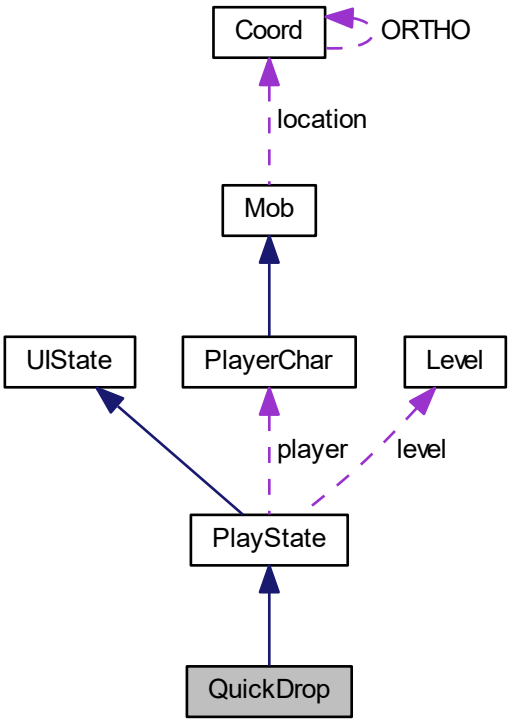
- include/potion.h
- potion.cpp

## 4.25 QuickDrop Class Reference

Inheritance diagram for QuickDrop:



Collaboration diagram for QuickDrop:




## Public Member Functions

- **QuickDrop** (PlayerChar *player, Level *level, Item *item)
- virtual UIState * handleInput (TCOD_key_t key)

  *Handle the various controls.*

## Additional Inherited Members

### 4.25.1 Member Function Documentation

#### 4.25.1.1 handleInput()

```
virtual UIState* QuickDrop::handleInput (
            TCOD_key_t key )  [inline], [virtual]
```

Handle the various controls.

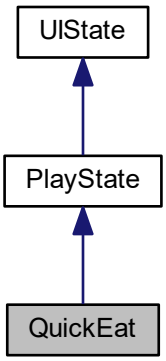
Reimplemented from PlayState.

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

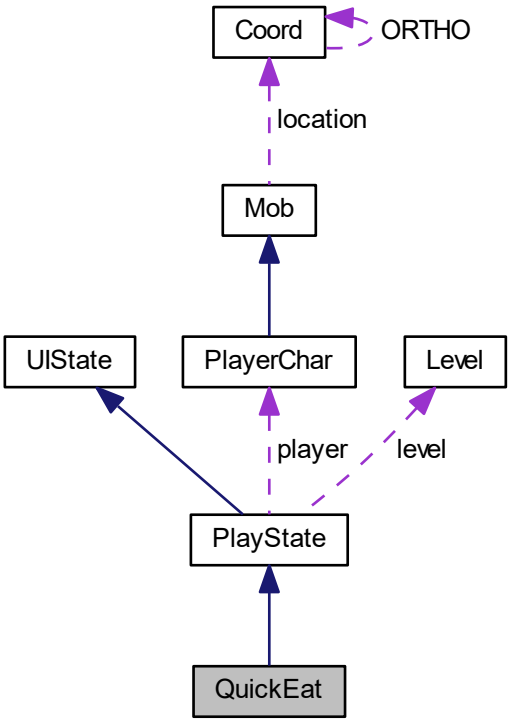
- playstate.cpp

## 4.26 QuickEat Class Reference

Inheritance diagram for QuickEat:




Collaboration diagram for QuickEat:



### Public Member Functions

- **QuickEat** (PlayerChar *player, Level *level, Item *item)
- virtual UIState * handleInput (TCOD_key_t key)

  *Handle the various controls.*

### Additional Inherited Members

### 4.26.1 Member Function Documentation

#### 4.26.1.1 handleInput()

```
virtual UIState* QuickEat::handleInput (
            TCOD_key_t key )  [inline], [virtual]
```

Handle the various controls.

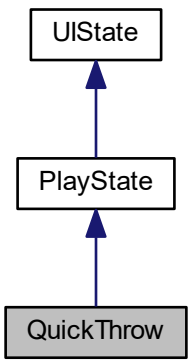
Reimplemented from PlayState.

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

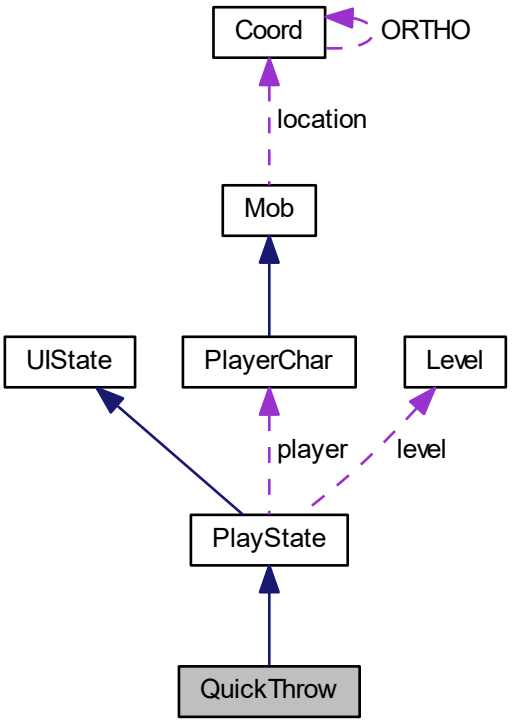
- playstate.cpp

## 4.27 QuickThrow Class Reference

Inheritance diagram for QuickThrow:



Collaboration diagram for QuickThrow:




## Public Member Functions

- **QuickThrow** (PlayerChar *player, Level *level, Item *item, Coord direction)
- virtual UIState * handleInput (TCOD_key_t key)

  *Handle the various controls.*

## Additional Inherited Members

### 4.27.1 Member Function Documentation

#### 4.27.1.1 handleInput()

```
virtual UIState* QuickThrow::handleInput (
            TCOD_key_t key )  [inline], [virtual]
```

Handle the various controls.

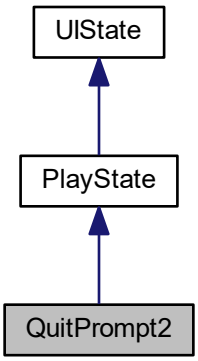
Reimplemented from PlayState.

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

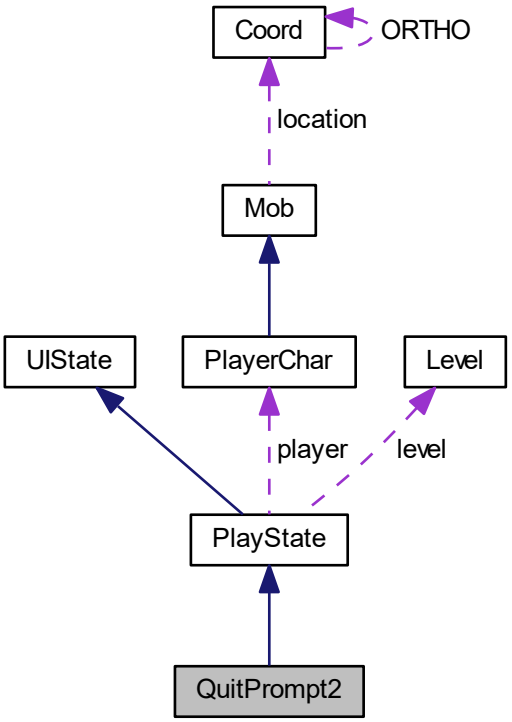
- playstate.cpp

## 4.28 QuitPrompt2 Class Reference

Inheritance diagram for QuitPrompt2:




Collaboration diagram for QuitPrompt2:



### Public Member Functions

- **QuitPrompt2** (PlayerChar *player, Level *level)
- virtual UIState * handleInput (TCOD_key_t key)

  *Handle the various controls.*
- virtual void draw (TCODConsole *con)

  *Render, drawing (in this order), ui, tiles, features, mobs.*

### Additional Inherited Members

## 4.28.1 Member Function Documentation

#### 4.28.1.1 handleInput()

```
virtual UIState* QuitPrompt2::handleInput (
            TCOD_key_t key )  [inline], [virtual]
```

Handle the various controls.

Reimplemented from PlayState.

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

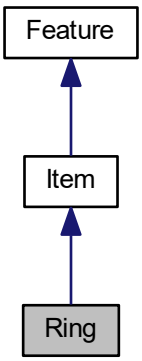
- playstate.cpp

# 4.29 Ring Class Reference

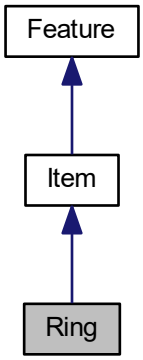
Represents rings.

```
#include <ring.h>
```

Inheritance diagram for Ring:



Collaboration diagram for Ring:




## Public Member Functions

- Ring (Coord)

  *Constructs a Ring instance with a random type.*
- Ring (Coord, Item::Context, int)

  *Constructs a Ring instance.*
- bool activate (Level ∗)

  *Applies the effects derived from equipping this Ring.*

## Additional Inherited Members

### 4.29.1 Detailed Description

Represents rings.

### 4.29.2 Constructor & Destructor Documentation

#### 4.29.2.1 Ring() [1/2]

```
Ring::Ring (
            Coord location )
```

Constructs a Ring instance with a random type.

**Parameters**

| | | |
|---|---|---|
| in | *location* | Ring location |

#### 4.29.2.2 Ring() [2/2]

```
Ring::Ring (
            Coord location,
            Item::Context context,
            int type )
```

Constructs a Ring instance.

**Parameters**

| | | |
|---|---|---|
| in | *location* | Ring location |
| in | *context* | Ring context |
| in | *type* | Ring type |

### 4.29.3 Member Function Documentation

#### 4.29.3.1 activate()

```
bool Ring::activate (
            Level * level )
```

Applies the effects derived from equipping this Ring.

**Parameters**

| | |
|---|---|
| *level* | Reference to the Level instance |

**Returns**

A value reflecting the success of the activation operation.

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

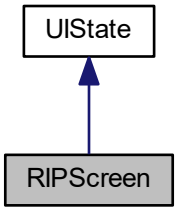
- include/ring.h
- ring.cpp

## 4.30 RIPScreen Class Reference

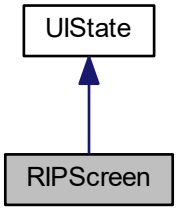
Interface state for post-death/retirement, looking at the high-score table.

```
#include <ripscreen.h>
```

Inheritance diagram for RIPScreen:




Collaboration diagram for RIPScreen:




### Public Member Functions

- RIPScreen (PlayerChar ∗, Level ∗level, std::string cause)

  *Constructor.*
- virtual void draw (TCODConsole ∗)

  *Render.*
- virtual UIState ∗ handleInput (TCOD_key_t)

  *Handle player key input.*

### 4.30.1 Detailed Description

Interface state for post-death/retirement, looking at the high-score table.

Environment variables: input device (e.g., keyboard), monitor, and the file system

### 4.30.2 Constructor & Destructor Documentation

#### 4.30.2.1 RIPScreen()

```
RIPScreen::RIPScreen (
            PlayerChar * player,
            Level * level,
            std::string cause )
```

Constructor.

**Parameters**

| | |
|---|---|
| *cause* | Cause of death/retirement |
| *level* | Level on which player died/retired |

### 4.30.3 Member Function Documentation

#### 4.30.3.1 draw()

```
void RIPScreen::draw (
            TCODConsole * con )  [virtual]
```

Render.

Reimplemented from UIState.

#### 4.30.3.2 handleInput()

```
UIState * RIPScreen::handleInput (
            TCOD_key_t key )  [virtual]
```

Handle player key input.

Reimplemented from UIState.

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

- include/ripscreen.h
- ripscreen.cpp

## 4.31 Room Class Reference

Models a room - a rectangular region of which there are (usually) 9 in any given dungeon level.

```
#include <room.h>
```

### Public Types

- enum **Darkness** { **DARK**, **LIT** }
- enum **Treasure** { **TREASURE**, **WORTHLESS** }
- enum **Hidden** { **HIDDEN**, **VISIBLE** }

### Public Member Functions

- **Room** (Coord, Coord, Darkness, Treasure, Hidden, Coord, bool)
- **Room** (Coord, Coord)
- Coord **operator[ ]** (int)
- void dig (Level &)

  *Clears a passable room in the designated level.*
- Coord **getPosition1** ()
- Coord **getPosition2** ()
- Coord **getRoomSize** ()
- Coord **getRoomIndex** ()
- bool exists ()

  *A non-existent room is one which is a 1x1 tunnel tile.*
- bool touches (Coord)

  *Tells you whether or not the coordinate touches the room.*
- void printInfo (int)

  *A diagnostic tool.*
- bool contains (Coord &, int border=0)

  *Tells you whether or not the coordinate is contained by the room.*
- Darkness **getDark** ()

## 4.31.1 Detailed Description

Models a room - a rectangular region of which there are (usually) 9 in any given dungeon level.

Rooms are connected by tunnels.

**See also**

Tunnel

## 4.31.2 Member Function Documentation

### 4.31.2.1 contains()

```
bool Room::contains (
            Coord & coord,
            int border = 0 )
```

Tells you whether or not the coordinate is contained by the room.

**Parameters**

| | |
|---|---|
| *Coord* | The coordinate to test |

**Returns**

True if the input is within the room, false otherwise.

#### 4.31.2.2 dig()

```
void Room::dig (
            Level & level )
```

Clears a passable room in the designated level.

**Parameters**

| | |
|---|---|
| *Level* | The level in which to dig |

#### 4.31.2.3 exists()

```
bool Room::exists ( )
```

A non-existent room is one which is a 1x1 tunnel tile.

**Returns**

True if the room is real, false if it is simply a tunnel piece.

#### 4.31.2.4 printInfo()

```
void Room::printInfo (
            int numToDisplay )
```

A diagnostic tool.

**Parameters**

| | | |
|---|---|---|
| in | *An* | integer to go along with the info (Used when printing info of multiple rooms). |

#### 4.31.2.5 touches()

```
bool Room::touches (
            Coord c )
```

Tells you whether or not the coordinate touches the room.

**Parameters**

| | | |
|---|---|---|
| in | *Coord* | The location to test |

**Returns**

True if coord can touch or intersect with the room, false otherwise

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

- include/room.h
- room.cpp

## 4.32 ScoreItem Struct Reference

### Public Member Functions

- **ScoreItem** (int gold, int depth, std::string name, std::string death)
- std::string **encode** ()
- bool **operator**< (const ScoreItem &other) const

### Static Public Member Functions

- static ScoreItem **decode** (std::string line)
- static bool **readItem** (std::stringstream &ss, std::string &str)

### Public Attributes

- int **gold**
- int **depth**
- std::string **name**
- std::string **death**

### Static Public Attributes

- static const char **DELIM** = ','

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

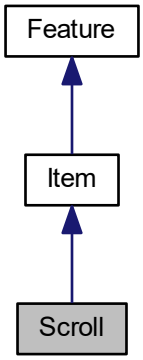
- ripscreen.cpp

## 4.33 Scroll Class Reference

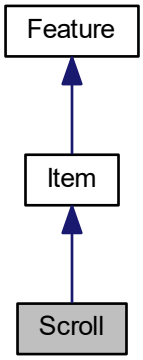
Represents scrolls.

`#include <scroll.h>`

Inheritance diagram for Scroll:




Collaboration diagram for Scroll:




### Public Member Functions

- Scroll (Coord)

  *Constructs a Scroll instance with a random type.*
- Scroll (Coord, Item::Context, int)

  *Constructs a Scroll instance.*
- bool activate (Level *)

  *Applies the effects derived from reading this Scroll.*

### Static Public Member Functions

- static std::vector< std::string > initializeScrollNames ()

  *Initializes the unidentified names of each Scroll.*

### Additional Inherited Members

## 4.33.1 Detailed Description

Represents scrolls.

## 4.33.2 Constructor & Destructor Documentation

### 4.33.2.1 Scroll() [1/2]

```
Scroll::Scroll (
            Coord location )
```

Constructs a Scroll instance with a random type.

**Parameters**

| | | |
|---|---|---|
| in | *location* | Scroll location |

### 4.33.2.2 Scroll() [2/2]

```
Scroll::Scroll (
            Coord location,
            Item::Context context,
            int type )
```

Constructs a Scroll instance.

**Parameters**

| | | |
|---|---|---|
| in | *location* | Scroll location |
| in | *context* | Scroll context |
| in | *type* | Scroll type |

## 4.33.3 Member Function Documentation

### 4.33.3.1 activate()

```
bool Scroll::activate (
            Level * level )
```

Applies the effects derived from reading this Scroll.

**Parameters**

| | |
|---|---|
| *level* | Reference to the Level instance |

**Returns**

A value reflecting the success of the activation operation.

#### 4.33.3.2 initializeScrollNames()

```
std::vector< std::string > Scroll::initializeScrollNames ( )  [static]
```

Initializes the unidentified names of each Scroll.

**Returns**

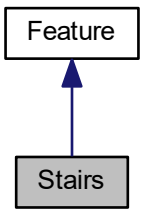
Returns a vector of strings denoting random Scroll names indexed by type.

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

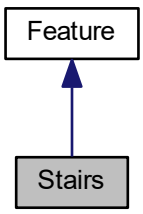
- include/scroll.h
- scroll.cpp

## 4.34 Stairs Class Reference

Inheritance diagram for Stairs:




Collaboration diagram for Stairs:



### Public Member Functions

- **Stairs** (Coord, bool)
- bool **getDirection** ()

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

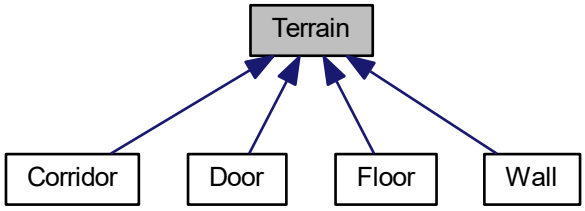
- include/stairs.h
- stairs.cpp

## 4.35 Terrain Class Reference

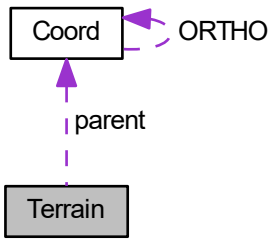
Represents a tile in the dungeon.

`#include <terrain.h>`

Inheritance diagram for Terrain:




Collaboration diagram for Terrain:



### Public Types

- enum Passability { **Blocked**, **Passable** }

  *Tiles can be walk-through-able or not.*
- enum Visibility { **Opaque**, **Corridor**, **Transparent** }

  *Tiles can have full, limited (Corridor), or no visibility.*
- enum Mapped { **Seen**, **UnSeen** }

  *Whether the player has previous seen the tile.*

### Public Member Functions

- Terrain (char, Visibility, Passability)

  *Constructor.*
- char getSymbol ()

  *Getter for character.*
- Passability isPassable ()

  *Getter for passable.*
- Mapped isSeen ()

  *Getter for seen.*
- Visibility getVisibility ()

  *Getter for visible.*
- void setIsSeen (Mapped)

  *Setter for seen.*

### Public Attributes

- bool checked = false

  *Used by other modules for various searches.*
- Coord parent

  *Used by other modules for various searches.*

## 4.35.1 Detailed Description

Represents a tile in the dungeon.

## 4.35.2 Member Enumeration Documentation

### 4.35.2.1 Mapped

`enum Terrain::Mapped`

Whether the player has previous seen the tile.

### 4.35.2.2 Passability

`enum Terrain::Passability`

Tiles can be walk-through-able or not.

### 4.35.3 Constructor & Destructor Documentation

#### 4.35.3.1 Terrain()

```
Terrain::Terrain (
            char character,
            Terrain::Visibility vis,
            Terrain::Passability pass )
```

Constructor.

### 4.35.4 Member Function Documentation

#### 4.35.4.1 getSymbol()

```
char Terrain::getSymbol ( )
```

Getter for character.

**See also**

character

#### 4.35.4.2 getVisibility()

```
Terrain::Visibility Terrain::getVisibility ( )
```

Getter for visible.

**See also**

visible

#### 4.35.4.3 isPassable()

```
Terrain::Passability Terrain::isPassable ( )
```

Getter for passable.

**See also**

passable

#### 4.35.4.4 isSeen()

```
Terrain::Mapped Terrain::isSeen ( )
```

Getter for seen.

**See also**

seen

#### 4.35.4.5 setIsSeen()

```
void Terrain::setIsSeen (
            Terrain::Mapped newState )
```

Setter for seen.

**See also**

seen

### 4.35.5 Member Data Documentation

#### 4.35.5.1 checked

```
bool Terrain::checked = false
```

Used by other modules for various searches.

**See also**

parent

#### 4.35.5.2 parent

```
Coord Terrain::parent
```

Used by other modules for various searches.

**See also**

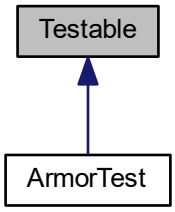
checked

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

- include/terrain.h
- terrain.cpp

## 4.36 Testable Class Reference

Inheritance diagram for Testable:




### Public Member Functions

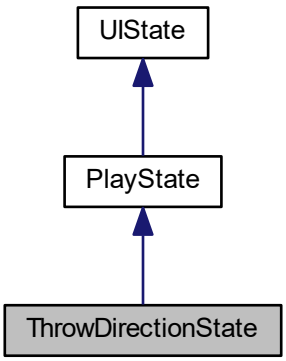
- virtual bool **test** ()=0

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

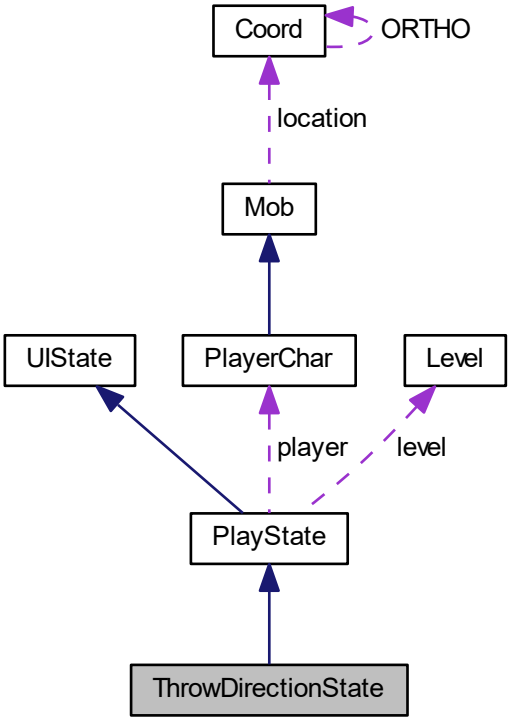
- test/test.testable.cpp

## 4.37 ThrowDirectionState Class Reference

Inheritance diagram for ThrowDirectionState:



Collaboration diagram for ThrowDirectionState:




## Public Member Functions

- **ThrowDirectionState** (PlayerChar *player, Level *level)
- virtual void draw (TCODConsole *con)

  *Render, drawing (in this order), ui, tiles, features, mobs.*
- virtual UIState * handleInput (TCOD_key_t key)

  *Handle the various controls.*

## Additional Inherited Members

### 4.37.1 Member Function Documentation

#### 4.37.1.1 handleInput()

```
virtual UIState* ThrowDirectionState::handleInput (
            TCOD_key_t key )  [inline], [virtual]
```

Handle the various controls.

Reimplemented from PlayState.

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

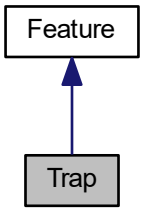
- playstate.cpp

## 4.38 Trap Class Reference

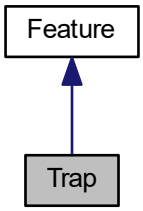
Various hidden traps throughout the dungeon can trigger and endanger the player.

`#include <trap.h>`

Inheritance diagram for Trap:




Collaboration diagram for Trap:




### Public Member Functions

- Trap (Coord location, unsigned char type, bool visible)

  *Constructor.*
- void activate (Mob *)

  *Trigger the trap on the given mob.*

### 4.38.1 Detailed Description

Various hidden traps throughout the dungeon can trigger and endanger the player.

### 4.38.2 Constructor & Destructor Documentation

#### 4.38.2.1 Trap()

```
Trap::Trap (
            Coord location,
            unsigned char type,
            bool visible )
```

Constructor.

**Parameters**

| | |
|---|---|
| *location* | Position of the trap |
| *type* | Type of trap (dart, teleport, pitfall, etc) |
| *visible* | Whether the trap is revealed |

### 4.38.3 Member Function Documentation

#### 4.38.3.1 activate()

```
void Trap::activate (
            Mob * mob )
```

Trigger the trap on the given mob.

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

- include/trap.h
- trap.cpp

## 4.39 Tunnel Class Reference

Tunnels are step-orthogonal paths connecting rooms.

```
#include <tunnel.h>
```

### Public Types

- enum Direction {
  **Up**, **Down**, **Left**, **Right**,
  **None** }

  *An enum to represent step directions.*

### Public Member Functions

- Tunnel (Room *, Room *, Generator)

  *Creates a tunnel between the two rooms.*
- void dig (Level &)

  *Digs the specified tunnel in the given level.*

## 4.39.1 Detailed Description

Tunnels are step-orthogonal paths connecting rooms.

## 4.39.2 Constructor & Destructor Documentation

### 4.39.2.1 Tunnel()

```
Tunnel::Tunnel (
            Room * p,
            Room * q,
            Generator gen )
```

Creates a tunnel between the two rooms.

**Parameters**

| | |
|---|---|
| *Room** | The room to go FROM |
| *Room** | The room to go TO |
| *Generator* | The random generator to use |

## 4.39.3 Member Function Documentation

### 4.39.3.1 dig()

```
void Tunnel::dig (
            Level & level )
```

Digs the specified tunnel in the given level.

**Parameters**

| | |
|---|---|
| *Level&* | The level in which to dig this tunnel |

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

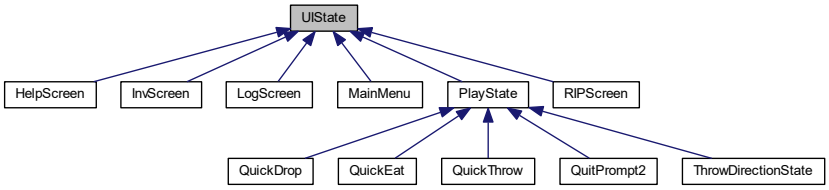
- include/tunnel.h
- tunnel.cpp

## 4.40 UIState Class Reference

Class modeling a state of the game interface.

`#include <uistate.h>`

Inheritance diagram for UIState:




### Public Member Functions

- virtual void draw (TCODConsole ∗)

  *Render the current UI.*
- virtual UIState ∗ handleInput (TCOD_key_t)

  *Do whatever is needed in response to keypresses then return state to transition to (can be self).*
- virtual ∼UIState ()

  *Destructor.*

### 4.40.1 Detailed Description

Class modeling a state of the game interface.

Game transitions between these states like a finite state machine.

Environment variables: input device (e.g., keyboard) and output device (e.g., monitor)

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

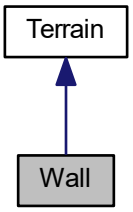
- include/uistate.h
- uistate.cpp

## 4.41 Wall Class Reference

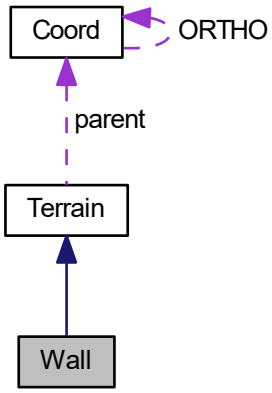
Regular dungeon wall.

`#include <tiles.h>`

Inheritance diagram for Wall:




Collaboration diagram for Wall:




### Additional Inherited Members

### 4.41.1 Detailed Description

Regular dungeon wall.

Has no visiblity or passability.

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

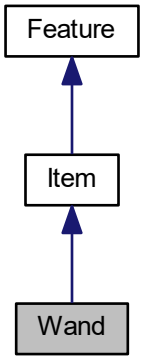
- include/tiles.h
- tiles.cpp

## 4.42 Wand Class Reference

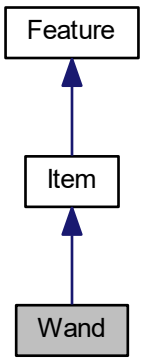
Represents a wand item.

`#include <wand.h>`

Inheritance diagram for Wand:




Collaboration diagram for Wand:




### Public Member Functions

- Wand (Coord)

  *Constructs a Wand instance with a random type.*
- Wand (Coord, Item::Context, int)

*Constructs a Wand instance.*

- bool activate (Level *)

  *Applies the effects derived from using a zap from this Wand.*

- int getCharges ()

  *Gets the charges.*

## Additional Inherited Members

### 4.42.1 Detailed Description

Represents a wand item.

### 4.42.2 Constructor & Destructor Documentation

#### 4.42.2.1 Wand() [1/2]

```
Wand::Wand (
            Coord location )
```

Constructs a Wand instance with a random type.

**Parameters**

| | | |
|---|---|---|
| in | *location* | Wand location |

#### 4.42.2.2 Wand() [2/2]

```
Wand::Wand (
            Coord location,
            Item::Context context,
            int type )
```

Constructs a Wand instance.

**Parameters**

| | | |
|---|---|---|
| in | *location* | Wand location |
| in | *context* | Wand context |
| in | *type* | Wand type |

### 4.42.3 Member Function Documentation

#### 4.42.3.1 activate()

```
bool Wand::activate (
            Level * level )
```

Applies the effects derived from using a zap from this Wand.

**Parameters**

| | |
|---|---|
| *level* | Reference to the Level instance |

**Returns**

A value reflecting the success of the activation operation.

#### 4.42.3.2 getCharges()

```
int Wand::getCharges ( )
```

Gets the charges.

**Returns**

The charges.

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

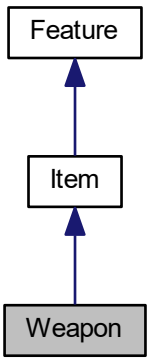
- include/wand.h
- wand.cpp

## 4.43 Weapon Class Reference

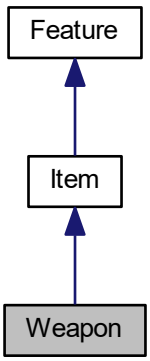
Represents weapons.

```
#include <weapon.h>
```

Inheritance diagram for Weapon:



Collaboration diagram for Weapon:




## Public Member Functions

- Weapon (Coord)

  *Constructs a Weapon instance with a random type.*
- Weapon (Coord, Item::Context, int)

  *Constructs a Weapon instance.*
- int getChance ()

  *Gets the chance of applying a successful hit.*
- std::tuple< int, int, int > getDamage ()

  *Gets the damage triple corresponding to this Weapon.*
- bool isMelee ()

  *Determines if this Weapon is a melee weapon.*
- void setEnchantments (int, int)

  *Sets this Weapon's enchantments.*

## Additional Inherited Members

### 4.43.1 Detailed Description

Represents weapons.

### 4.43.2 Constructor & Destructor Documentation

#### 4.43.2.1 Weapon() [1/2]

```
Weapon::Weapon (
            Coord location )
```

Constructs a Weapon instance with a random type.

**Parameters**

| | | |
|---|---|---|
| in | *location* | Weapon location |

#### 4.43.2.2 Weapon() [2/2]

```
Weapon::Weapon (
            Coord location,
            Item::Context context,
            int type )
```

Constructs a Weapon instance.

**Parameters**

| | | |
|---|---|---|
| in | *location* | Weapon location |
| in | *context* | Weapon context |
| in | *type* | Weapon type |

### 4.43.3 Member Function Documentation

#### 4.43.3.1 getChance()

```
int Weapon::getChance ( )
```

Gets the chance of applying a successful hit.

**Returns**

The chance of applying a successful hit.

#### 4.43.3.2 getDamage()

```
std::tuple< int, int, int > Weapon::getDamage ( )
```

Gets the damage triple corresponding to this Weapon.

**Returns**

The tuple <Dice Rolls, Dice Value, Enchantment>.

#### 4.43.3.3 isMelee()

```
bool Weapon::isMelee ( )
```

Determines if this Weapon is a melee weapon.

**Returns**

True if melee, False otherwise.

#### 4.43.3.4 setEnchantments()

```
void Weapon::setEnchantments (
            int enchantHit,
            int enchantDamage )
```

Sets this Weapon's enchantments.

**Parameters**

| | |
|---|---|
| *enchantHit* | Hit enchantment |
| *enchantDamage* | Damage enchantment |

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

- include/weapon.h
- weapon.cpp

# Chapter 5

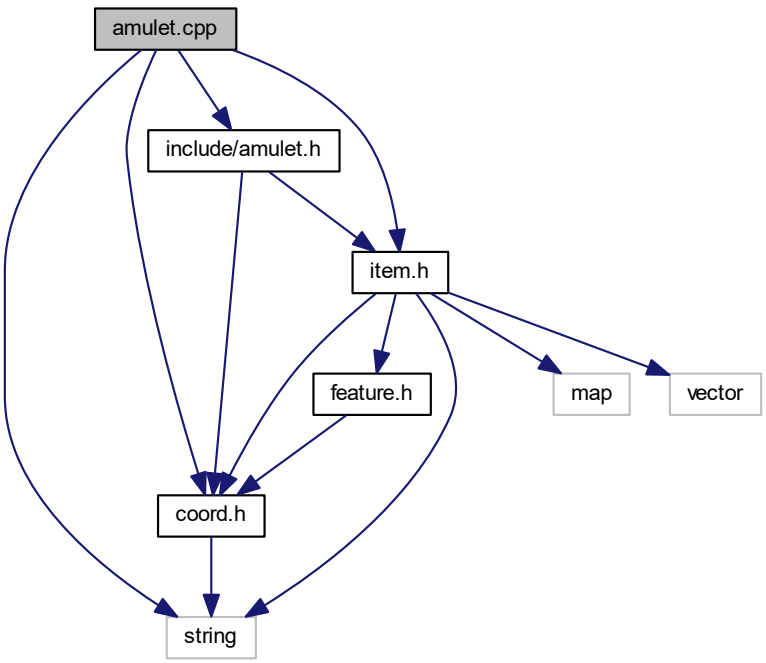
# File Documentation

## 5.1 amulet.cpp File Reference

Member definitions for the Amulet class.

```
#include <string>
#include "include/amulet.h"
#include "include/coord.h"
#include "include/item.h"
```

Include dependency graph for amulet.cpp:



### 5.1.1 Detailed Description

Member definitions for the Amulet class.

**Author**

Team Rogue++

**Date**

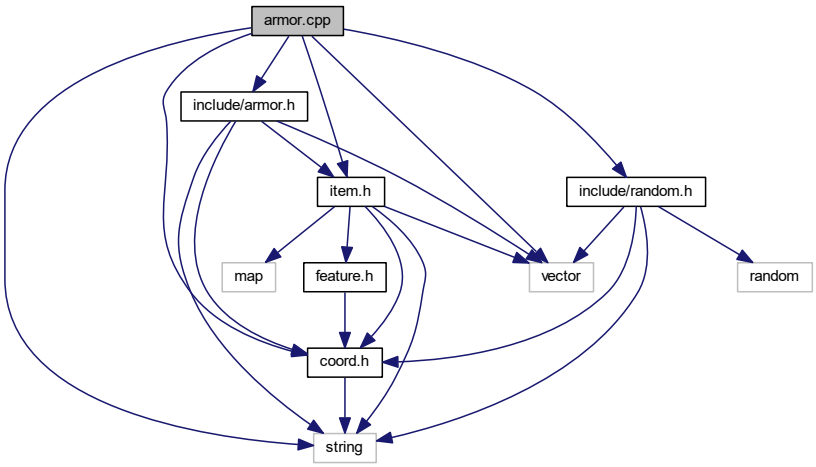
November 13, 2016

## 5.2 armor.cpp File Reference

Member definitions for the Armor class.

```
#include <string>
#include <vector>
#include "include/armor.h"
#include "include/coord.h"
#include "include/item.h"
#include "include/random.h"
```

Include dependency graph for armor.cpp:




### 5.2.1 Detailed Description

Member definitions for the Armor class.

**Author**

Team Rogue++

**Date**

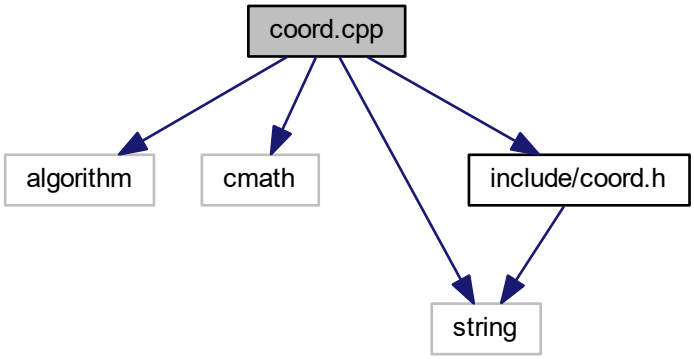
November 13, 2016

## 5.3 coord.cpp File Reference

Member definitions for the Coord class.

```
#include <algorithm>
#include <cmath>
#include <string>
#include "include/coord.h"
```

Include dependency graph for coord.cpp:




### 5.3.1 Detailed Description

Member definitions for the Coord class.

**Author**

Team Rogue++

**Date**

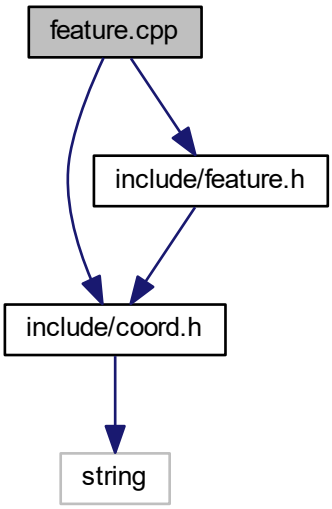
November 13, 2016

## 5.4 feature.cpp File Reference

Member definitions for the Feature class.

```
#include "include/coord.h"
#include "include/feature.h"
```

Include dependency graph for feature.cpp:




### 5.4.1 Detailed Description

Member definitions for the Feature class.

**Author**

Team Rogue++

**Date**

November 13, 2016

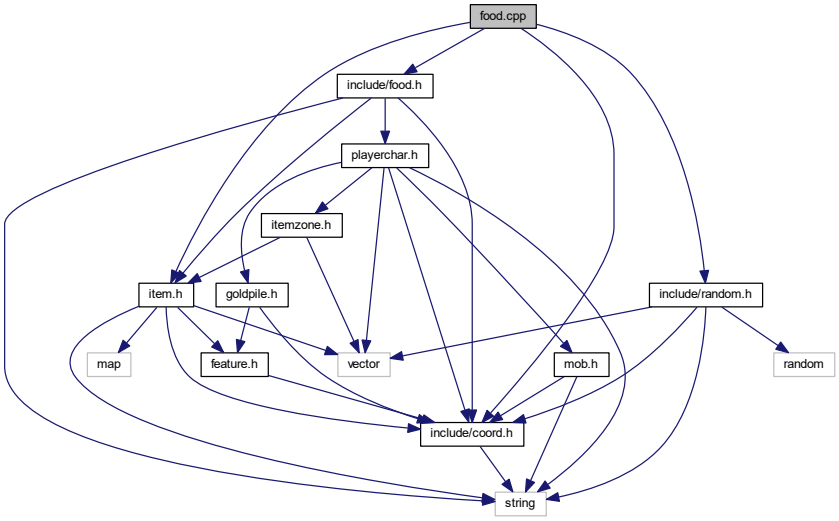
## 5.5 food.cpp File Reference

Member definitions for the Food class.

```
#include "include/coord.h"
#include "include/food.h"
#include "include/item.h"
```

```
#include "include/random.h"
```

Include dependency graph for food.cpp:




### 5.5.1 Detailed Description

Member definitions for the Food class.

**Author**

Team Rogue++

**Date**

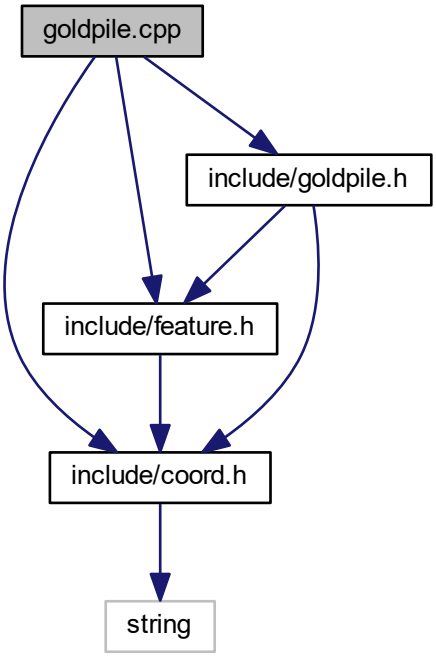
November 13, 2016

## 5.6 goldpile.cpp File Reference

Member definitions for the GoldPile class.

```
#include "include/coord.h"
#include "include/feature.h"
```

```
#include "include/goldpile.h"
```
Include dependency graph for goldpile.cpp:




### 5.6.1 Detailed Description

Member definitions for the GoldPile class.

**Author**

Team Rogue++

**Date**

November 13, 2016

## 5.7 helpscreen.cpp File Reference

Member definitions for the HelpScreen class.

```
#include "include/helpscreen.h"
#include "include/playstate.h"
```
Include dependency graph for helpscreen.cpp:

### 5.7.1 Detailed Description

Member definitions for the HelpScreen class.

**Author**

Team Rogue++

**Date**

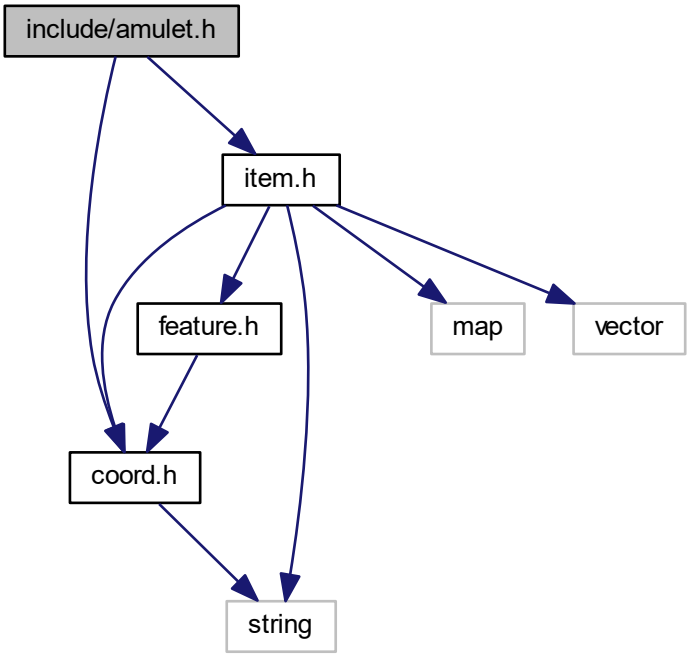
November 13, 2016

## 5.8 include/amulet.h File Reference

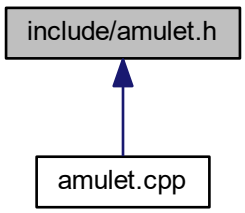
Member declarations for the Amulet class.

```
#include "coord.h"
#include "item.h"
```

Include dependency graph for amulet.h:



This graph shows which files directly or indirectly include this file:




### Classes

- class Amulet

  *Represents the Amulet of Yendor.*

### 5.8.1 Detailed Description

Member declarations for the Amulet class.

**Author**

Team Rogue++

**Date**

November 13, 2016

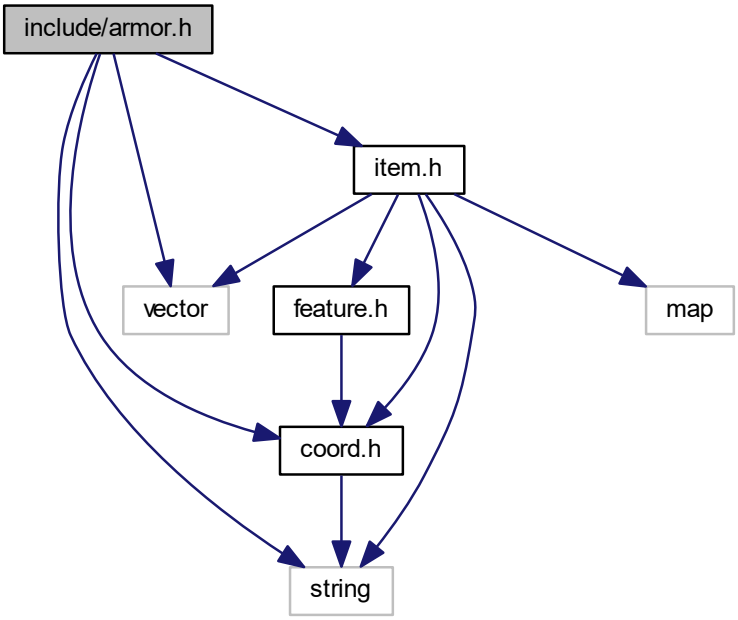
## 5.9 include/armor.h File Reference

Member declarations for the Armor class.

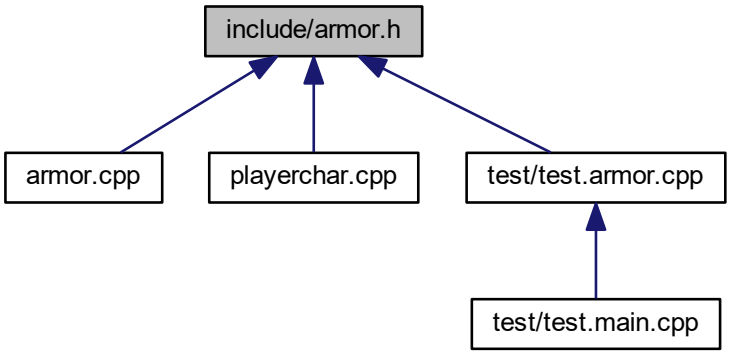
```
#include <string>
#include <vector>
#include "coord.h"
```

```
#include "item.h"
```

Include dependency graph for armor.h:




This graph shows which files directly or indirectly include this file:




## Classes

- class Armor

  *Represents armor.*

### Typedefs

- using ARMOR_TUPLE_TYPE = std::tuple< std::string, int >

  *Tuple representing Armor information (<Name, Rating>)*

### 5.9.1 Detailed Description

Member declarations for the Armor class.

#### Author

Team Rogue++

#### Date

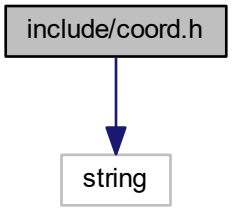
November 13, 2016

## 5.10 include/coord.h File Reference

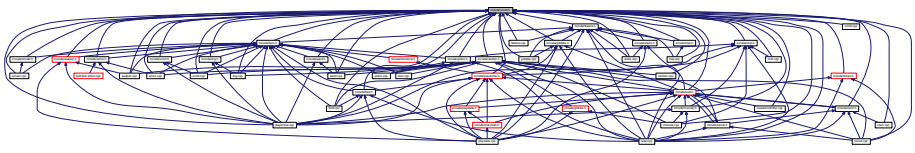
Member declarations for the Coord class.

```
#include <string>
```

Include dependency graph for coord.h:




This graph shows which files directly or indirectly include this file:



### Classes

- class Coord

  *Represents a location within the dungeon or on the screen.*

### 5.10.1 Detailed Description

Member declarations for the Coord class.

**Author**

Team Rogue++

**Date**

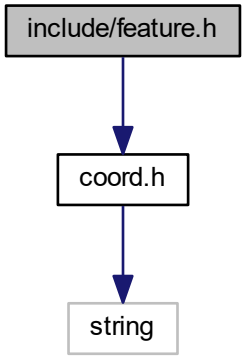
November 13, 2016

## 5.11 include/feature.h File Reference

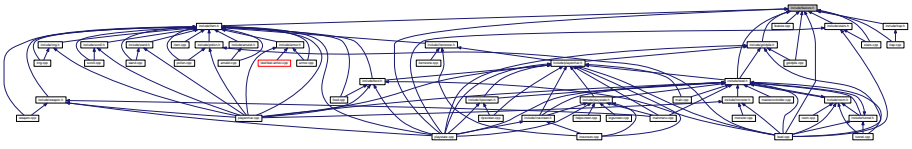
Member declarations for the Feature class.

```
#include "coord.h"
```

Include dependency graph for feature.h:




This graph shows which files directly or indirectly include this file:



### Classes

- class Feature

  *Models a 'thing' in the dungeon that has position and may be visible.*

### 5.11.1 Detailed Description

Member declarations for the Feature class.

**Author**

Team Rogue++

**Date**

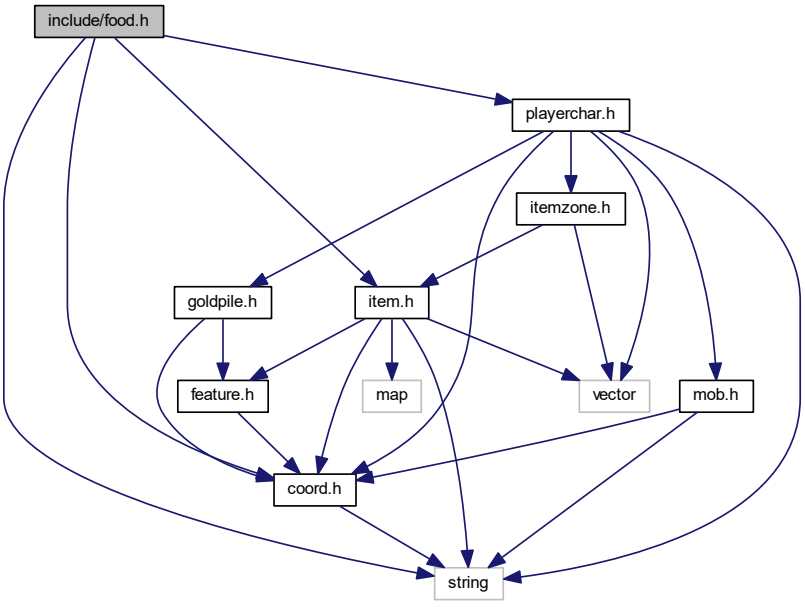
November 13, 2016

## 5.12 include/food.h File Reference

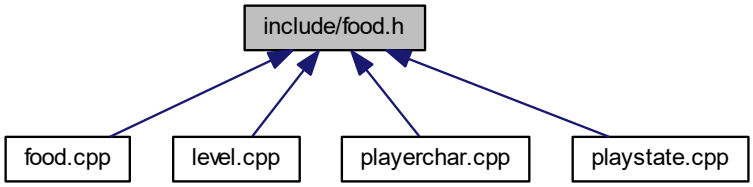
Member declarations for the Food class.

```
#include <string>
#include "coord.h"
#include "item.h"
#include "playerchar.h"
```

Include dependency graph for food.h:



This graph shows which files directly or indirectly include this file:




### Classes

- class Food

  *Represents food.*

### 5.12.1 Detailed Description

Member declarations for the Food class.

**Author**

Team Rogue++

**Date**

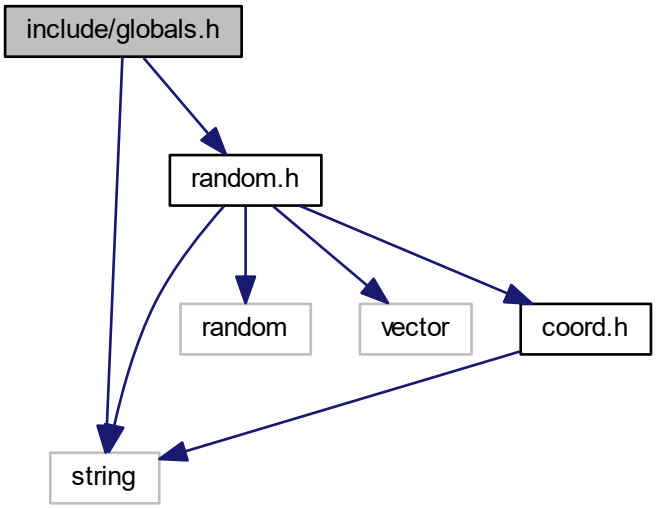
November 13, 2016

## 5.13 include/globals.h File Reference

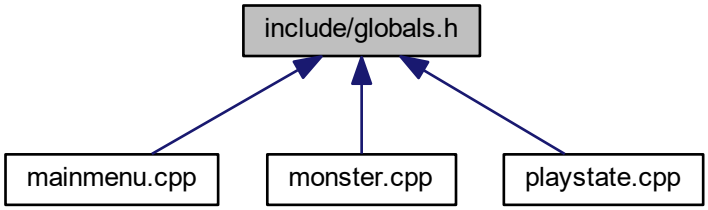
Global members.

```
#include <string>
#include "random.h"
```

Include dependency graph for globals.h:




This graph shows which files directly or indirectly include this file:




### Variables

- const int **NUM_LEVELS** = 26
- const int **NAME_LENGTH** = 10
- const std::string **VALID_NAME** = "abcdefghijklmnopqrstuvwxyz _ABCDEFGHIJKLMNOPQRSTUVWXYZ"
- const int **TURN_TIME** = 50

#### 5.13.1 Detailed Description

Global members.

**Author**

Team Rogue++

**Date**

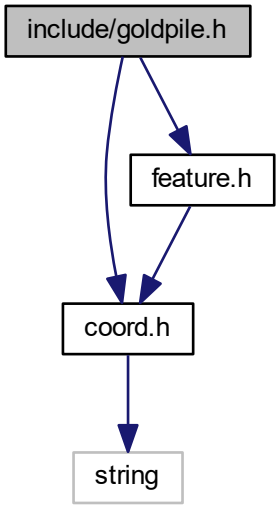
November 13, 2016

## 5.14 include/goldpile.h File Reference

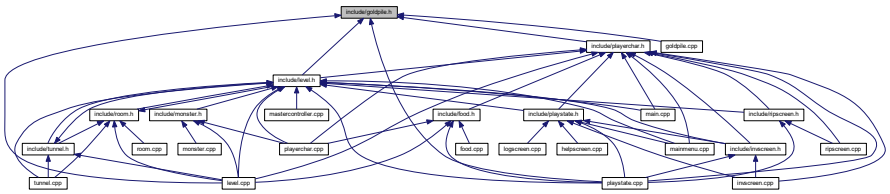
Member declarations for the GoldPile class.

```
#include "coord.h"
#include "feature.h"
```

Include dependency graph for goldpile.h:




This graph shows which files directly or indirectly include this file:



### Classes

- class GoldPile

  *Represents a pile of gold on the ground, which can be picked up by the player to enhance their score.*

### 5.14.1 Detailed Description

Member declarations for the GoldPile class.

**Author**

Team Rogue++

**Date**

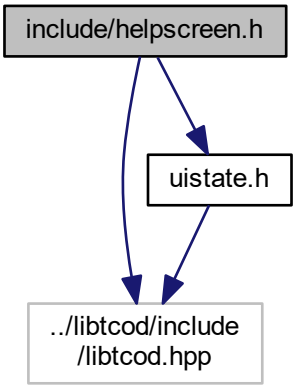
November 13, 2016

## 5.15 include/helpscreen.h File Reference

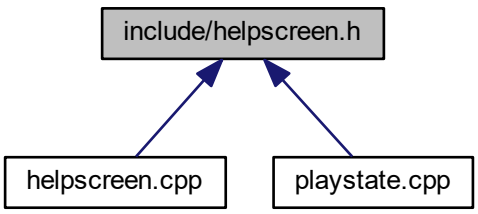
Member declarations for the HelpScreen class.

```
#include "../libtcod/include/libtcod.hpp"
#include "uistate.h"
```

Include dependency graph for helpscreen.h:




This graph shows which files directly or indirectly include this file:



### Classes

- class HelpScreen

  *Interface state that shows the various game controls.*

### 5.15.1 Detailed Description

Member declarations for the HelpScreen class.

#### Author

Team Rogue++

#### Date

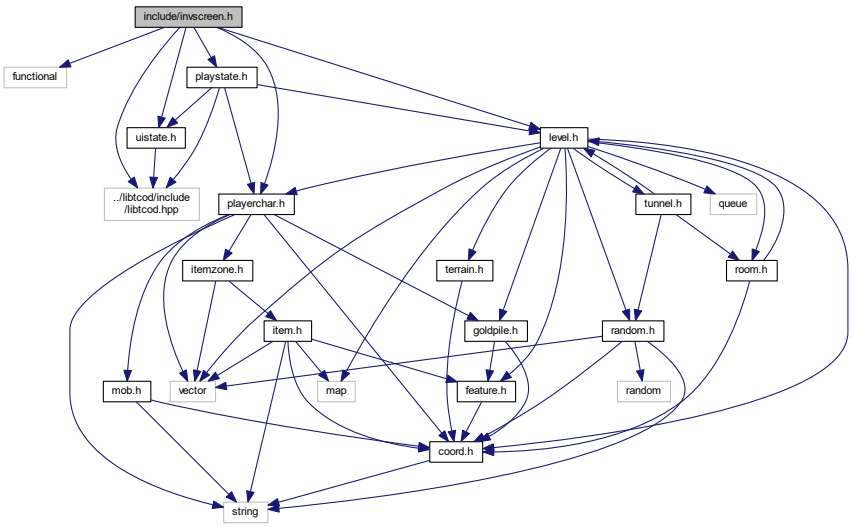
November 13, 2016

## 5.16 include/invscreen.h File Reference

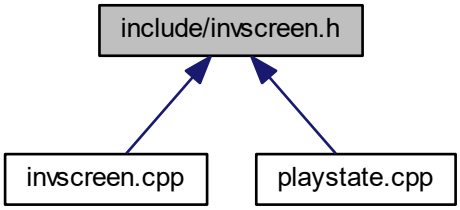
Member declarations for the InvScreen class.

```
#include <functional>
#include "../libtcod/include/libtcod.hpp"
#include "level.h"
#include "playerchar.h"
#include "playstate.h"
#include "uistate.h"
```

Include dependency graph for invscreen.h:



This graph shows which files directly or indirectly include this file:




### Classes

- class InvScreen

  *Interface state for viewing the contents of the player inventory.*

#### 5.16.1 Detailed Description

Member declarations for the InvScreen class.

**Author**

Team Rogue++

**Date**

November 13, 2016

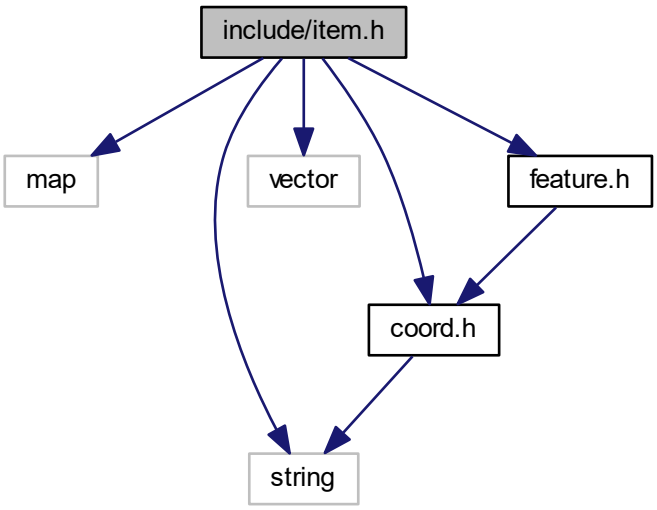
## 5.17 include/item.h File Reference

Member declarations for the Item class.

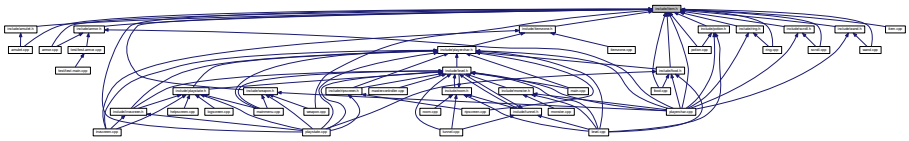
```
#include <map>
#include <string>
#include <vector>
#include "coord.h"
```

```
#include "feature.h"
```

Include dependency graph for item.h:




This graph shows which files directly or indirectly include this file:



## Classes

- class Item

  *Represents a generic item.*

### 5.17.1 Detailed Description

Member declarations for the Item class.

**Author**

Team Rogue++

**Date**

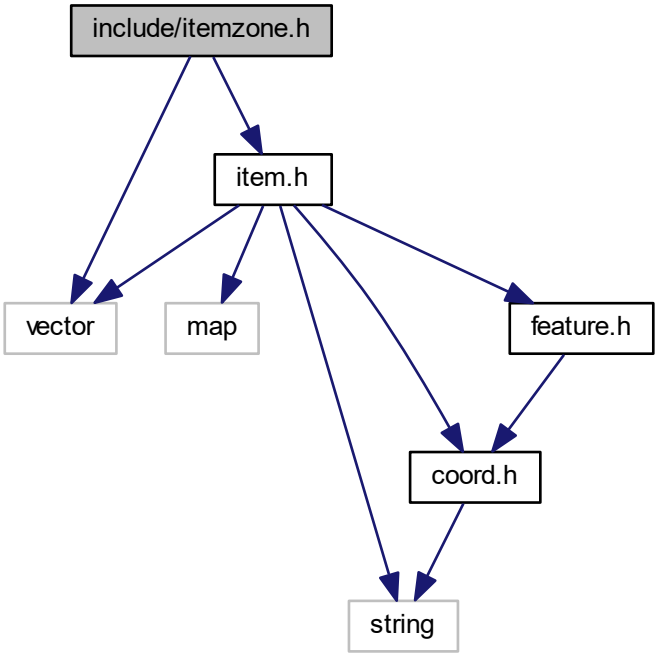
November 13, 2016

## 5.18 include/itemzone.h File Reference

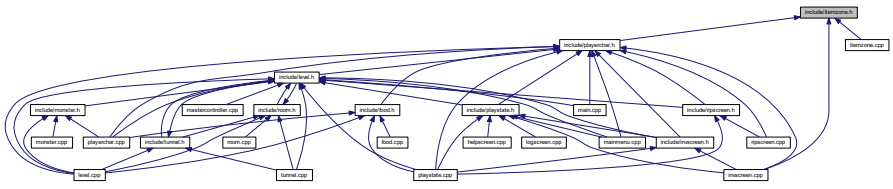
Member declarations for the ItemZone class.

```
#include <vector>
#include "item.h"
```

Include dependency graph for itemzone.h:




This graph shows which files directly or indirectly include this file:



### Classes

- class ItemZone

  *Container for items.*

### 5.18.1 Detailed Description

Member declarations for the ItemZone class.

**Author**

Team Rogue++

**Date**

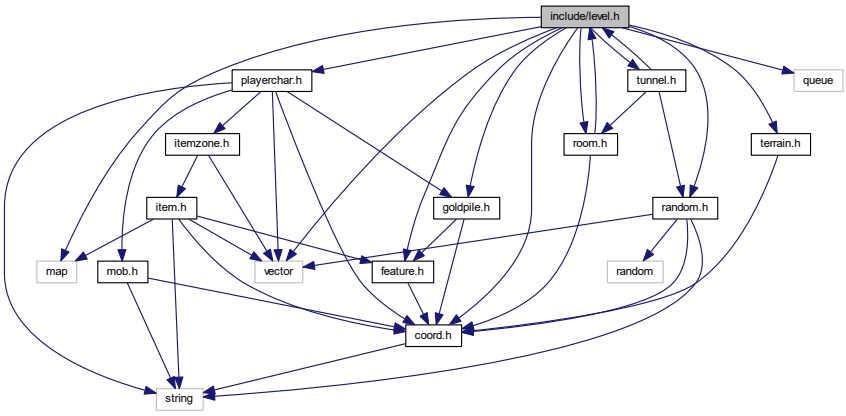
November 13, 2016

## 5.19 include/level.h File Reference

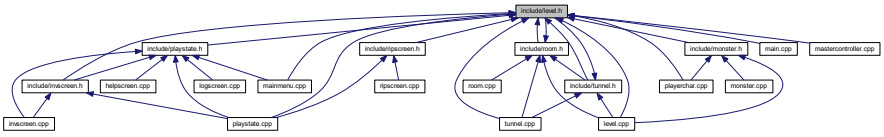
Member declarations for the Level class.

```
#include <map>
#include <queue>
#include <vector>
#include "coord.h"
#include "feature.h"
#include "goldpile.h"
#include "playerchar.h"
#include "random.h"
#include "room.h"
#include "terrain.h"
#include "tunnel.h"
```

Include dependency graph for level.h:




This graph shows which files directly or indirectly include this file:



### Classes

- class Level

### Macros

- #define **MAX_ROOMS_DEF** (9)

#### 5.19.1 Detailed Description

Member declarations for the Level class.

**Author**

Team Rogue++

**Date**

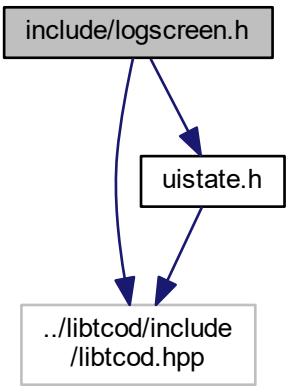
November 13, 2016

## 5.20 include/logscreen.h File Reference

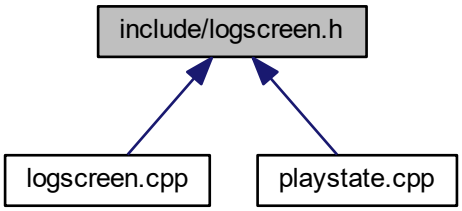
Member declarations for the LogScreen class.

```
#include "../libtcod/include/libtcod.hpp"
#include "uistate.h"
```

Include dependency graph for logscreen.h:



This graph shows which files directly or indirectly include this file:




### Classes

- class LogScreen

  *Controls the display of the event log.*

### 5.20.1 Detailed Description

Member declarations for the LogScreen class.

**Author**

Team Rogue++

**Date**

November 13, 2016

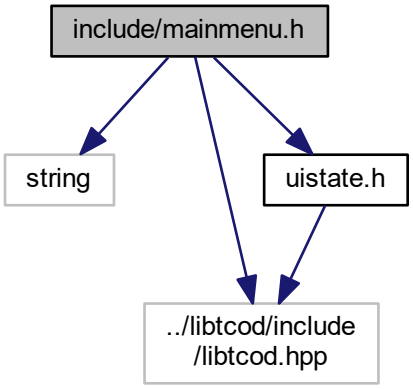
## 5.21 include/mainmenu.h File Reference

Member declarations for the MainMenu class.

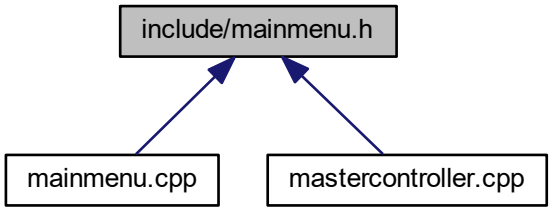
```
#include <string>
#include "../libtcod/include/libtcod.hpp"
```

```
#include "uistate.h"
```

Include dependency graph for mainmenu.h:




This graph shows which files directly or indirectly include this file:




## Classes

- class MainMenu

  *Start screen of the game.*

### 5.21.1 Detailed Description

Member declarations for the MainMenu class.

**Author**

Team Rogue++

**Date**

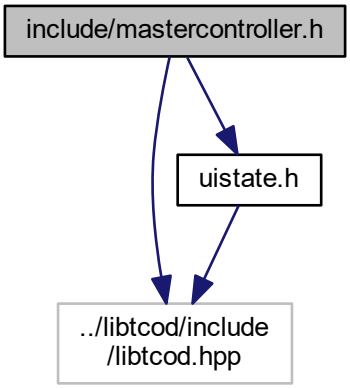
November 13, 2016

## 5.22 include/mastercontroller.h File Reference

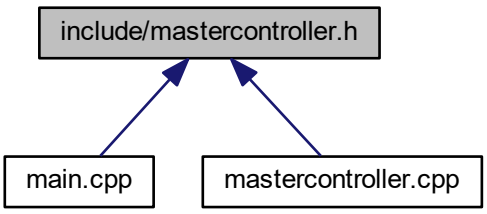
Member declarations for the MasterController class.

```
#include "../libtcod/include/libtcod.hpp"
#include "uistate.h"
```

Include dependency graph for mastercontroller.h:




This graph shows which files directly or indirectly include this file:




### Classes

- class MasterController

  *Controls the top level flow flow of the application and main game loop.*

### 5.22.1 Detailed Description

Member declarations for the MasterController class.

**Author**

Team Rogue++

**Date**

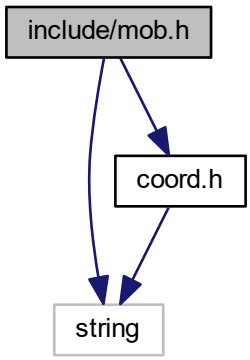
November 13, 2016

## 5.23 include/mob.h File Reference

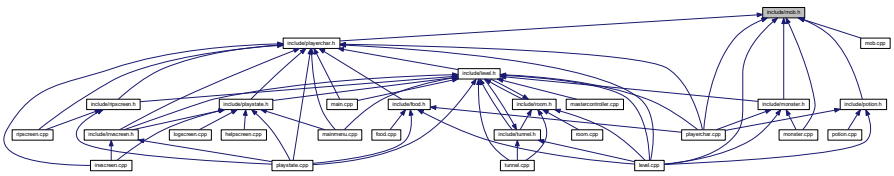
Member declarations for the Mob class.

```
#include <string>
#include "coord.h"
```

Include dependency graph for mob.h:




This graph shows which files directly or indirectly include this file:



### Classes

- class Mob

  *Models a creature in the dungeon, could be the player or a monster.*

### 5.23.1 Detailed Description

Member declarations for the Mob class.

**Author**

Team Rogue++

**Date**

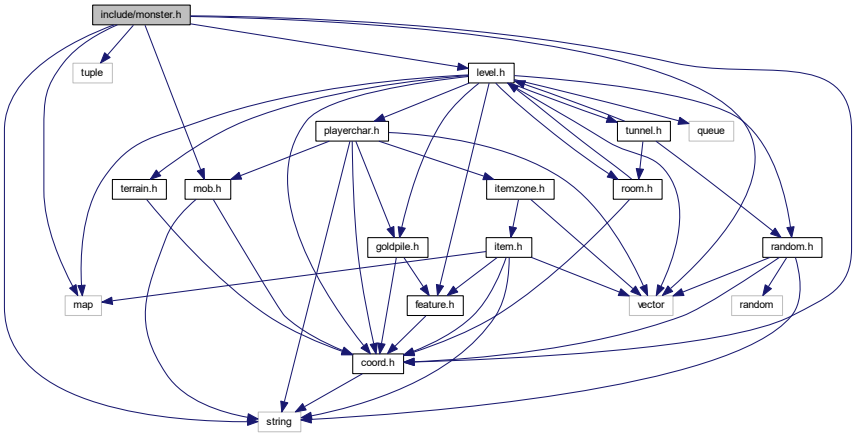
November 13, 2016

## 5.24 include/monster.h File Reference

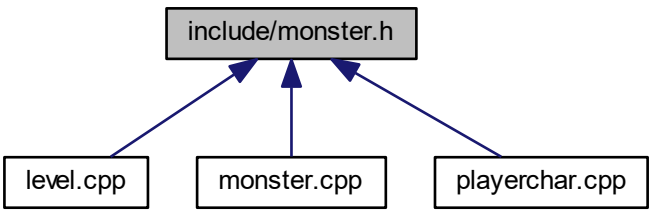
Member declarations for the Monster class.

```
#include <map>
#include <string>
#include <tuple>
#include <vector>
#include "coord.h"
#include "level.h"
#include "mob.h"
```

Include dependency graph for monster.h:




This graph shows which files directly or indirectly include this file:



### Classes

- class Monster

  *Models a monster in the dungeon.*

### Typedefs

- using MONSTER_TUPLE_TYPE = std::tuple< int, int, std::vector< std::pair< int, int > >, int, const char *, int, std::pair< int, int >, std::string, std::pair< int, int > >

  *Tuple representing various Monster types (<Armor, Carry Chance, Attacks, XP, Flags, Monster Level, HP, Name, Dungeon Level Range>)*

#### 5.24.1 Detailed Description

Member declarations for the Monster class.

**Author**

Team Rogue++

**Date**

November 13, 2016

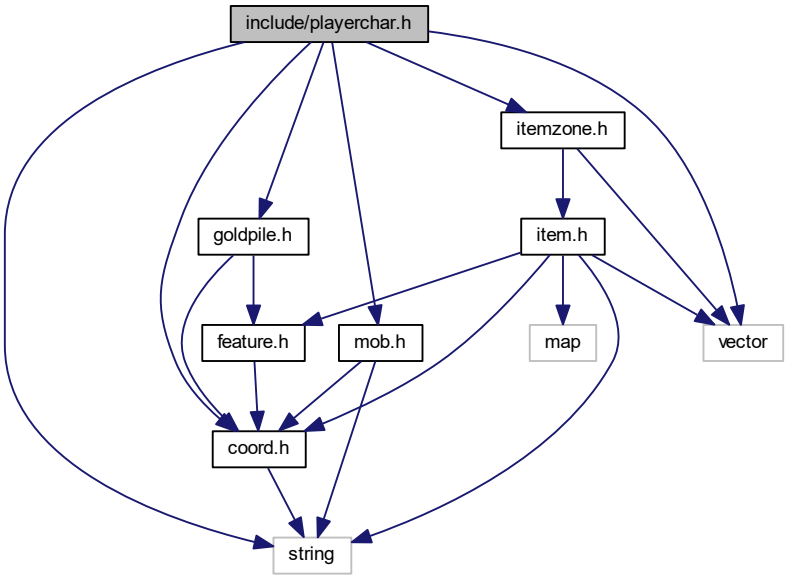
## 5.25 include/playerchar.h File Reference

Member declarations for the PlayerChar class.

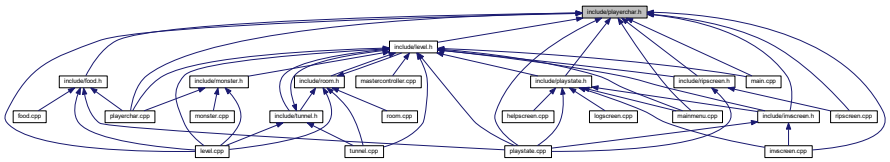
```
#include <string>
#include <vector>
#include "coord.h"
#include "goldpile.h"
#include "itemzone.h"
```

```
#include "mob.h"
```

Include dependency graph for playerchar.h:




This graph shows which files directly or indirectly include this file:



## Classes

- class PlayerChar

  *Models the user-controlled player character.*

### 5.25.1 Detailed Description

Member declarations for the PlayerChar class.

**Author**

Team Rogue++

**Date**

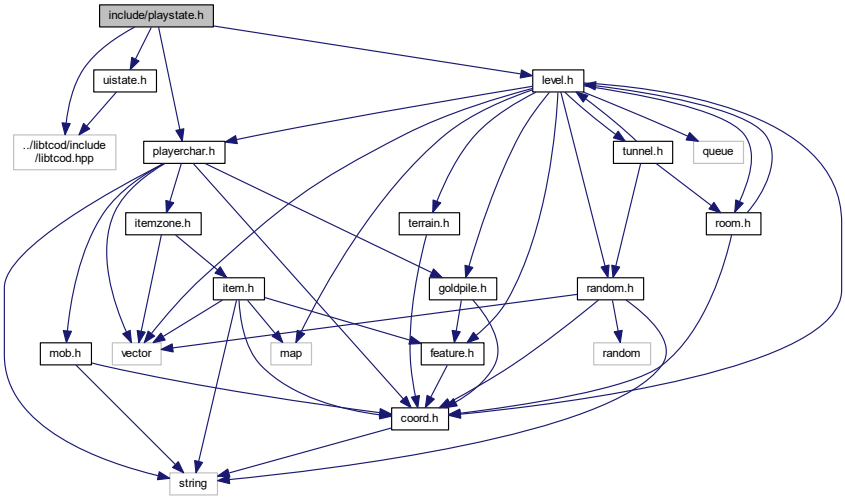
November 13, 2016

## 5.26 include/playstate.h File Reference

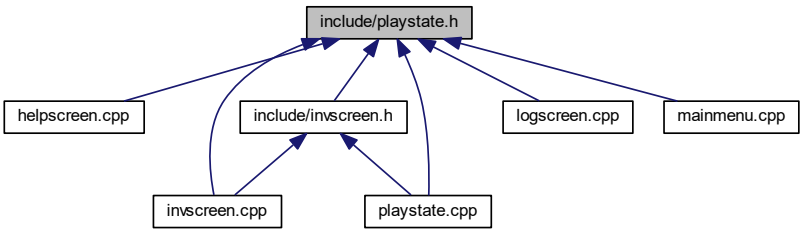
Member declarations for the PlayState class.

```
#include "../libtcod/include/libtcod.hpp"
#include "level.h"
#include "playerchar.h"
#include "uistate.h"
```

Include dependency graph for playstate.h:




This graph shows which files directly or indirectly include this file:




### Classes

- class PlayState

  *Primary interface state, showing level, player, monsters, etc.*

### 5.26.1 Detailed Description

Member declarations for the PlayState class.

**Author**

Team Rogue++

**Date**

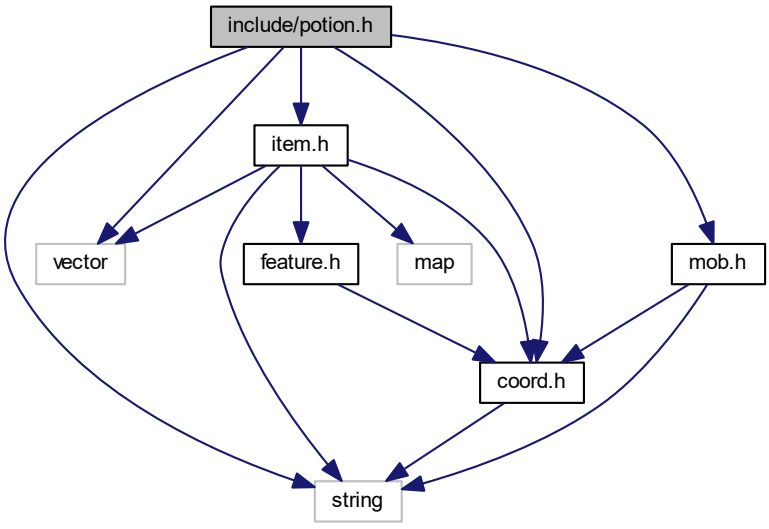
November 13, 2016

## 5.27 include/potion.h File Reference

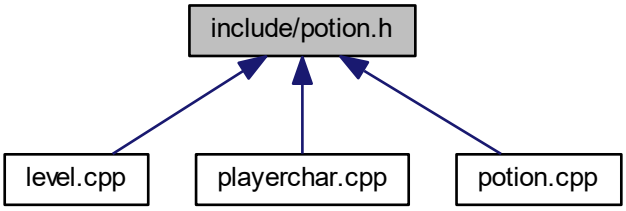
Member declarations for the Potion class.

```
#include <string>
#include <vector>
#include "coord.h"
#include "item.h"
#include "mob.h"
```

Include dependency graph for potion.h:



This graph shows which files directly or indirectly include this file:




### Classes

- class Potion

  *Represents potions.*

### Typedefs

- using POTION_TUPLE_TYPE = std::tuple< std::string >

  *Tuple representing Potion information (<Name>)*

#### 5.27.1 Detailed Description

Member declarations for the Potion class.

**Author**

Team Rogue++

**Date**

November 13, 2016

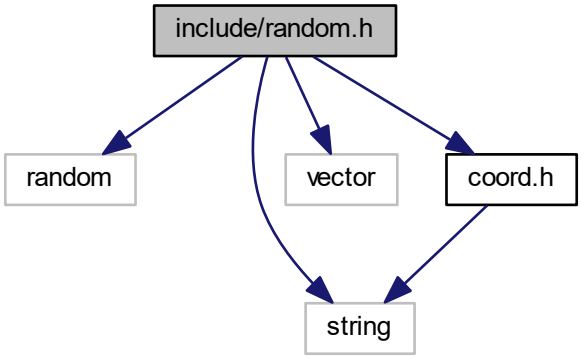
## 5.28 include/random.h File Reference

Member declarations for the Generator class.

```
#include <random>
#include <string>
#include <vector>
```

```
#include "coord.h"
```

Include dependency graph for random.h:




This graph shows which files directly or indirectly include this file:

## Classes

- class Generator

  *Light wrapper around the std library which provides various random generation utilities.*

### 5.28.1 Detailed Description

Member declarations for the Generator class.

**Author**

Team Rogue++

**Date**

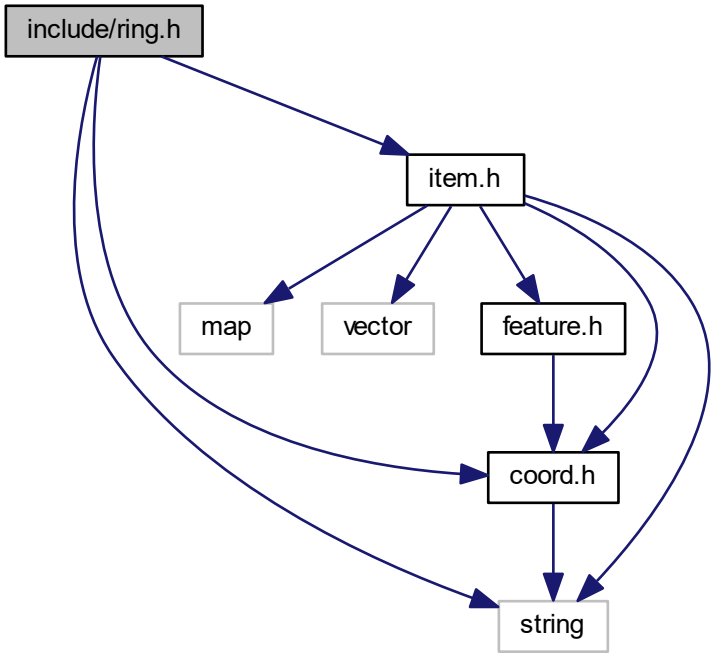
November 13, 2016

## 5.29 include/ring.h File Reference

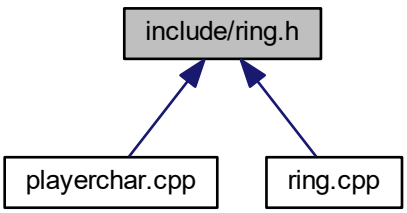
Member declarations for the Ring class.

```
#include <string>
#include "coord.h"
#include "item.h"
```

Include dependency graph for ring.h:




This graph shows which files directly or indirectly include this file:



### Classes

- class Ring

  *Represents rings.*

### Typedefs

- using RING_TUPLE_TYPE = std::tuple< std::string >

  *Tuple representing Ring information (<Name>)*

#### 5.29.1 Detailed Description

Member declarations for the Ring class.

**Author**

Team Rogue++

**Date**

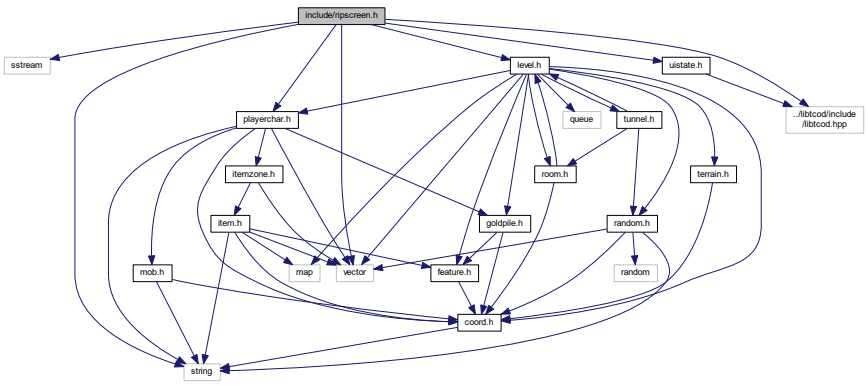
November 13, 2016

## 5.30 include/ripscreen.h File Reference

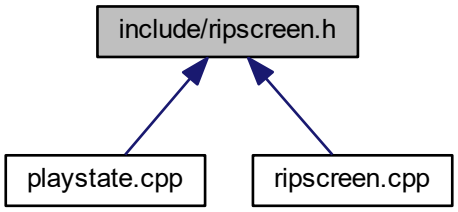
Member declarations for the RIPScreen class.

```
#include <sstream>
#include <string>
#include <vector>
#include "../libtcod/include/libtcod.hpp"
#include "level.h"
#include "playerchar.h"
#include "uistate.h"
```

Include dependency graph for ripscreen.h:



This graph shows which files directly or indirectly include this file:




### Classes

- class RIPScreen

  *Interface state for post-death/retirement, looking at the high-score table.*

#### 5.30.1 Detailed Description

Member declarations for the RIPScreen class.

**Author**

Team Rogue++

**Date**

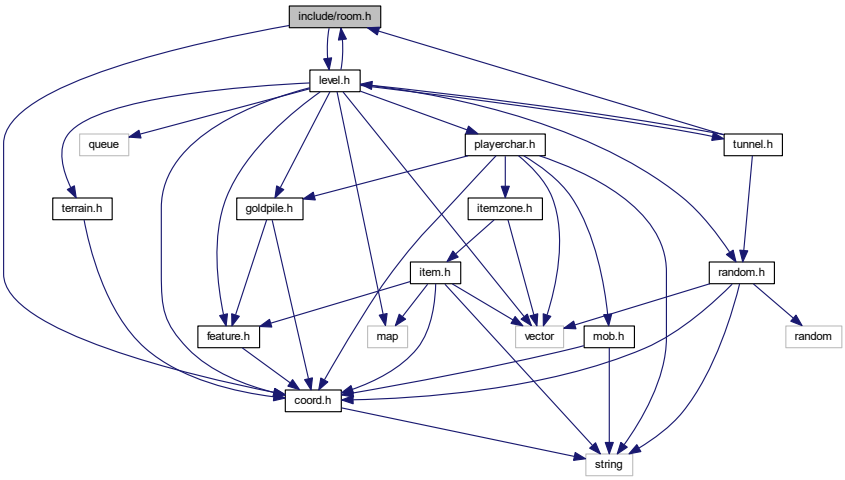
November 13, 2016

## 5.31 include/room.h File Reference

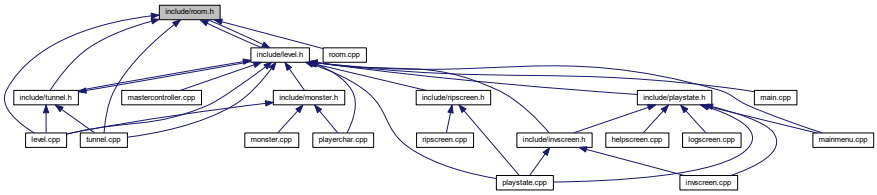
Member declarations for the Room class.

```
#include "coord.h"
#include "level.h"
```

Include dependency graph for room.h:




This graph shows which files directly or indirectly include this file:




## Classes

- class Room

  *Models a room - a rectangular region of which there are (usually) 9 in any given dungeon level.*

### 5.31.1 Detailed Description

Member declarations for the Room class.

**Author**

Team Rogue++

**Date**

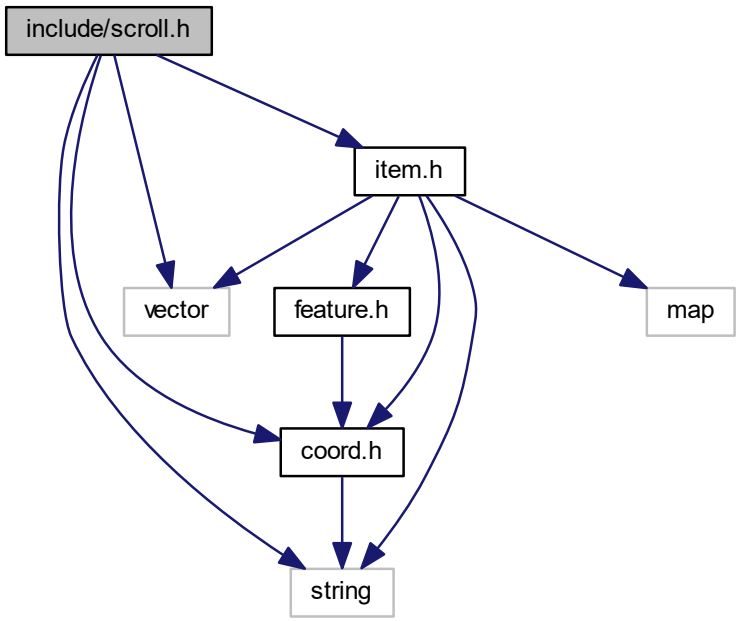
November 13, 2016

## 5.32 include/scroll.h File Reference

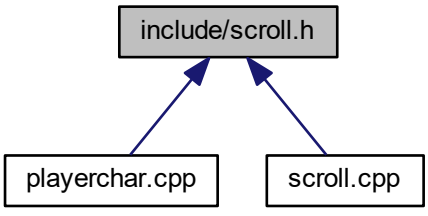
Member declarations for the Scroll class.

```
#include <string>
#include <vector>
#include "coord.h"
#include "item.h"
```

Include dependency graph for scroll.h:




This graph shows which files directly or indirectly include this file:




### Classes

- class Scroll

  *Represents scrolls.*

### Typedefs

- using SCROLL_TUPLE_TYPE = std::tuple< std::string >

  *Tuple representing Scroll information (<Name>)*

### 5.32.1 Detailed Description

Member declarations for the Scroll class.

**Author**

Team Rogue++

**Date**

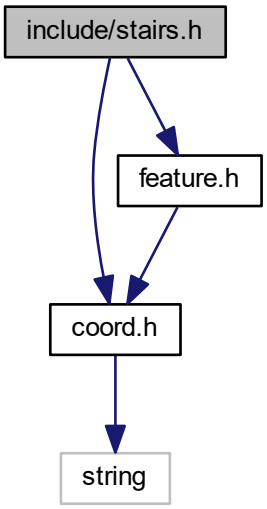
November 13, 2016

## 5.33 include/stairs.h File Reference

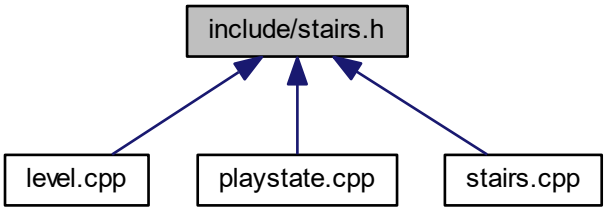
Member declarations for the Stairs class.

```
#include "coord.h"
#include "feature.h"
```

Include dependency graph for stairs.h:



This graph shows which files directly or indirectly include this file:




### Classes

- class Stairs

#### 5.33.1 Detailed Description

Member declarations for the Stairs class.

**Author**

Team Rogue++

**Date**

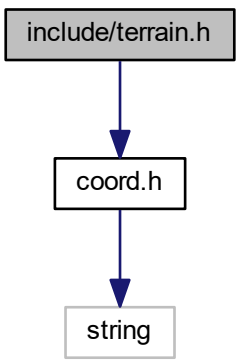
November 13, 2016

## 5.34 include/terrain.h File Reference

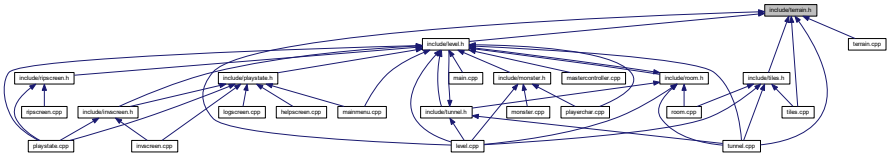
Member declarations for the Terrain class.

```
#include "coord.h"
```

Include dependency graph for terrain.h:



This graph shows which files directly or indirectly include this file:



## Classes

- class Terrain

  *Represents a tile in the dungeon.*

### 5.34.1 Detailed Description

Member declarations for the Terrain class.

**Author**

Team Rogue++

**Date**

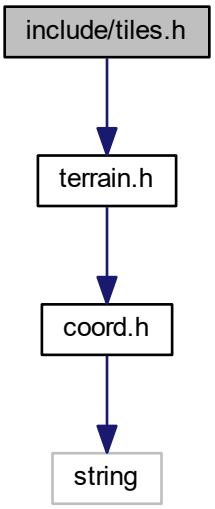
November 13, 2016

## 5.35 include/tiles.h File Reference

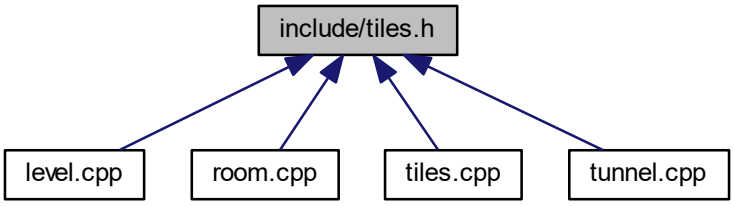
Member declarations for the Corridor, Door, Floor, Wall classes.

```
#include "terrain.h"
```

Include dependency graph for tiles.h:



This graph shows which files directly or indirectly include this file:




### Classes

- class Floor

  *Regular dungeon floor.*
- class Wall

  *Regular dungeon wall.*
- class Corridor

  *Regular corridor tile.*
- class Door

  *Door tile.*

#### 5.35.1 Detailed Description

Member declarations for the Corridor, Door, Floor, Wall classes.

**Author**

Team Rogue++

**Date**

November 13, 2016

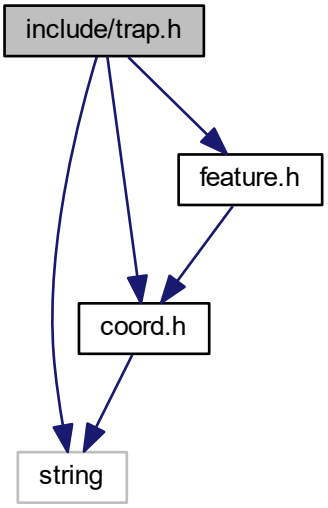
## 5.36 include/trap.h File Reference

Member declarations for the Trap class.

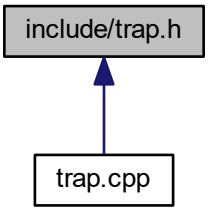
```
#include <string>
#include "coord.h"
```

```
#include "feature.h"
```

Include dependency graph for trap.h:




This graph shows which files directly or indirectly include this file:




## Classes

- class Trap

  *Various hidden traps throughout the dungeon can trigger and endanger the player.*

### 5.36.1 Detailed Description

Member declarations for the Trap class.

**Author**

Team Rogue++

**Date**

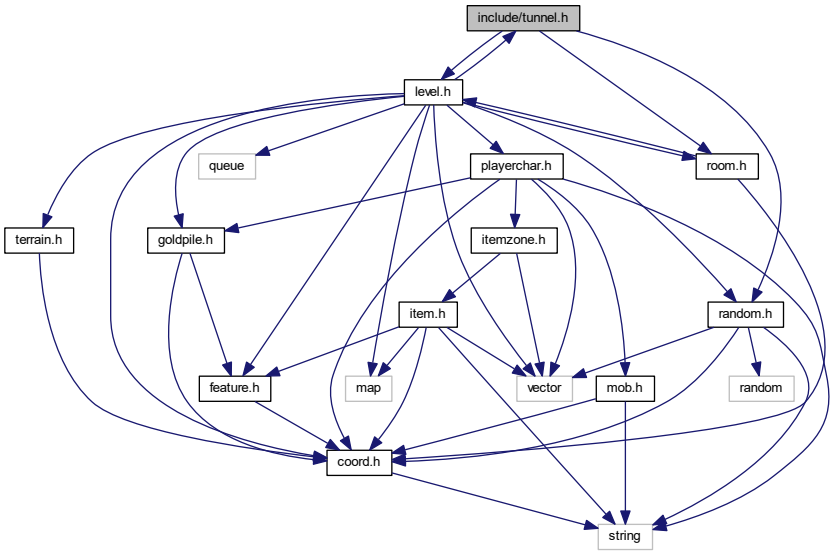
November 13, 2016

## 5.37 include/tunnel.h File Reference

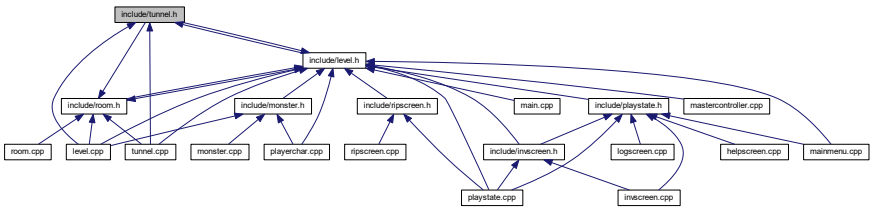
Member declarations for the Tunnel class.

```
#include "level.h"
#include "random.h"
#include "room.h"
```

Include dependency graph for tunnel.h:




This graph shows which files directly or indirectly include this file:




### Classes

- class Tunnel

  *Tunnels are step-orthogonal paths connecting rooms.*

### 5.37.1 Detailed Description

Member declarations for the Tunnel class.

**Author**

Team Rogue++

**Date**

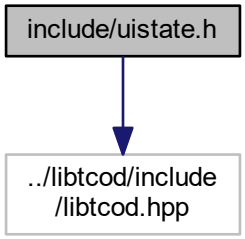
November 13, 2016

## 5.38 include/uistate.h File Reference

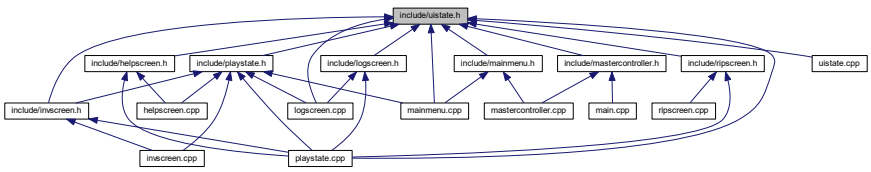
Member declarations for the UIState class.

```
#include "../libtcod/include/libtcod.hpp"
```

Include dependency graph for uistate.h:




This graph shows which files directly or indirectly include this file:




### Classes

- class UIState

  *Class modeling a state of the game interface.*

### 5.38.1 Detailed Description

Member declarations for the UIState class.

**Author**

Team Rogue++

**Date**

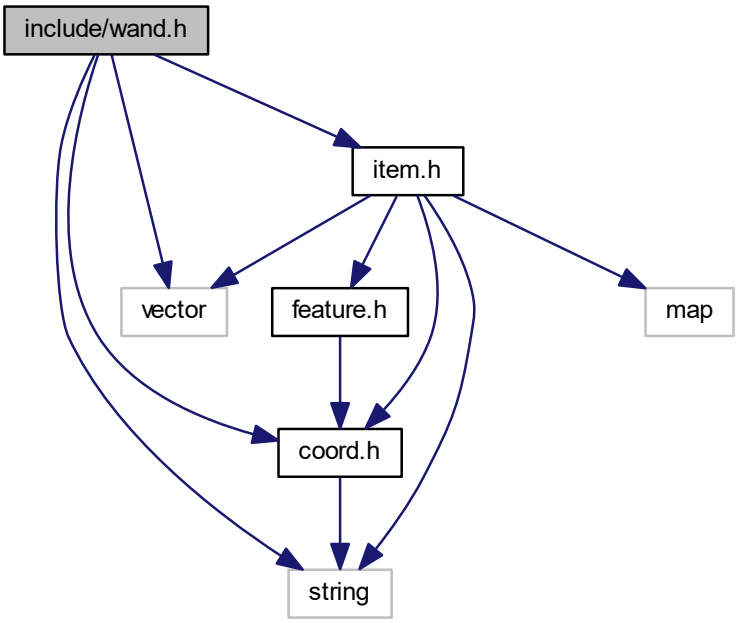
November 13, 2016

## 5.39 include/wand.h File Reference

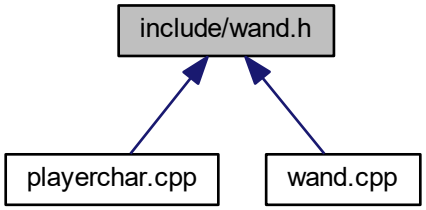
Member declarations for the Wand class.

```
#include <string>
#include <vector>
#include "coord.h"
#include "item.h"
```

Include dependency graph for wand.h:



This graph shows which files directly or indirectly include this file:




### Classes

- class Wand

  *Represents a wand item.*

### Typedefs

- using WAND_TUPLE_TYPE = std::tuple< std::string >

  *Tuple representing Wand information (<Name>)*

#### 5.39.1 Detailed Description

Member declarations for the Wand class.

**Author**

Team Rogue++

**Date**

November 13, 2016

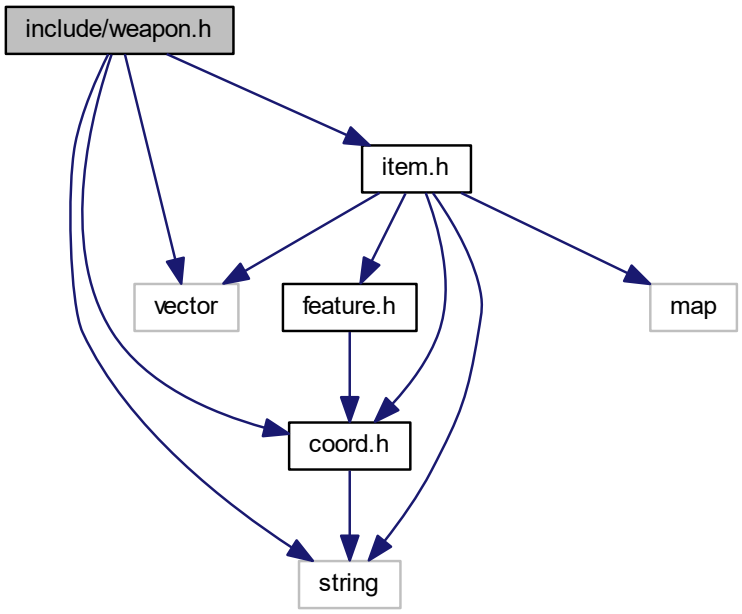
## 5.40 include/weapon.h File Reference

Member declarations for the Weapon class.

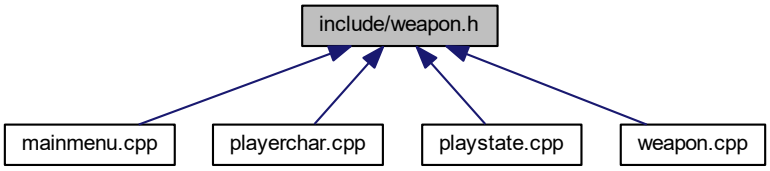
```
#include <string>
#include <vector>
#include "coord.h"
```

```
#include "item.h"
```

Include dependency graph for weapon.h:




This graph shows which files directly or indirectly include this file:




## Classes

- class Weapon

  *Represents weapons.*

## Typedefs

- using WEAPON_TUPLE_TYPE = std::tuple< std::string, std::pair< int, int >, bool, bool >

  *Tuple representing Weapon information (<Name, Damage, Melee, Stackable>)*

### 5.40.1 Detailed Description

Member declarations for the Weapon class.

**Author**

Team Rogue++

**Date**

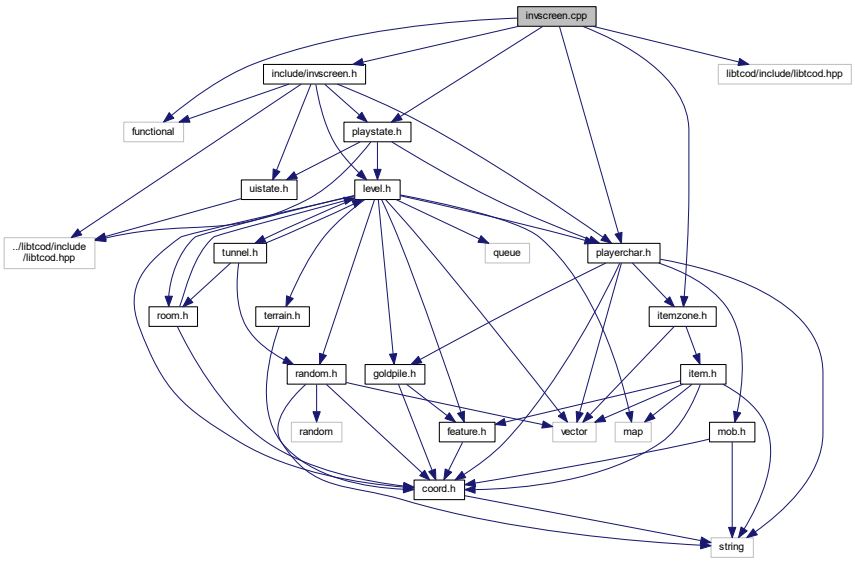
November 13, 2016

## 5.41 invscreen.cpp File Reference

Member definitions for the InvScreen class.

```
#include <functional>
#include "include/invscreen.h"
#include "include/itemzone.h"
#include "include/playerchar.h"
#include "include/playstate.h"
#include "libtcod/include/libtcod.hpp"
```

Include dependency graph for invscreen.cpp:




### 5.41.1 Detailed Description

Member definitions for the InvScreen class.

**Author**

Team Rogue++

**Date**

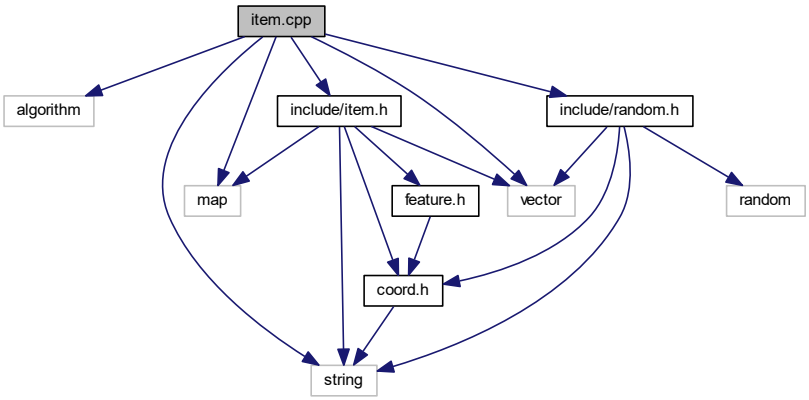
November 13, 2016

## 5.42 item.cpp File Reference

Member definitions for the Item class.

```
#include <algorithm>
#include <map>
#include <string>
#include <vector>
#include "include/item.h"
#include "include/random.h"
```

Include dependency graph for item.cpp:




### 5.42.1 Detailed Description

Member definitions for the Item class.

**Author**

Team Rogue++

**Date**

November 13, 2016

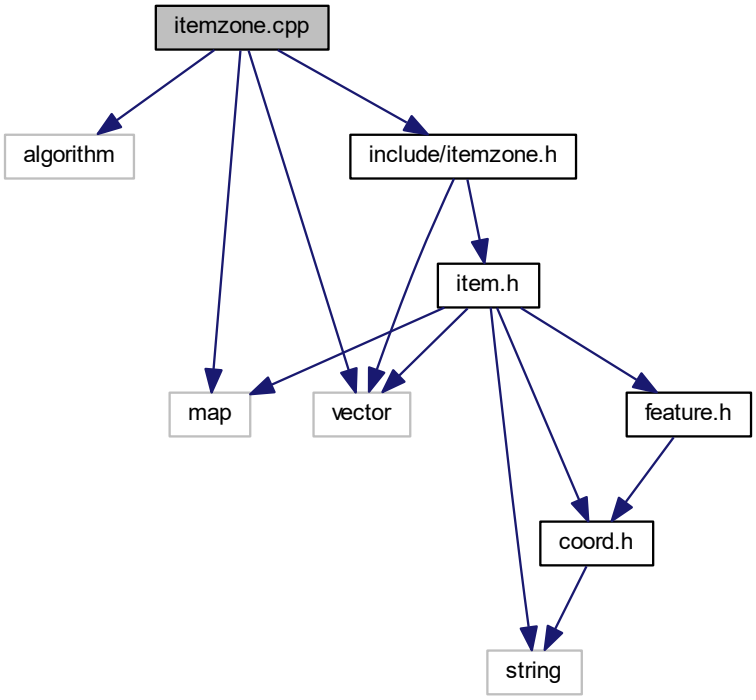
## 5.43 itemzone.cpp File Reference

Member definitions for the ItemZone class.

```
#include <algorithm>
#include <map>
#include <vector>
```

```
#include "include/itemzone.h"
```

Include dependency graph for itemzone.cpp:




### 5.43.1 Detailed Description

Member definitions for the ItemZone class.

**Author**

Team Rogue++

**Date**

November 13, 2016

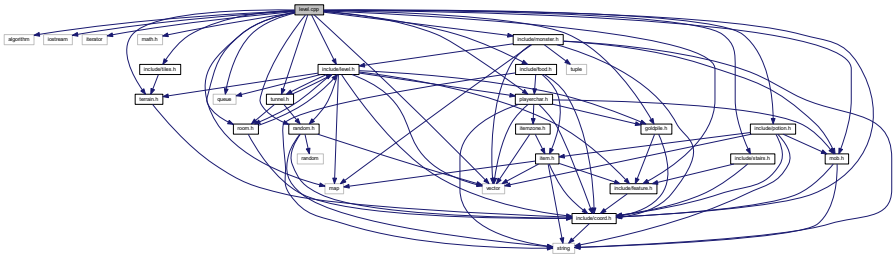
## 5.44 level.cpp File Reference

Member definitions for the Level class.

```
#include <algorithm>
#include <iostream>
#include <iterator>
#include <map>
#include <math.h>
```

```
#include <queue>
#include <vector>
#include "include/coord.h"
#include "include/feature.h"
#include "include/food.h"
#include "include/goldpile.h"
#include "include/level.h"
#include "include/mob.h"
#include "include/monster.h"
#include "include/playerchar.h"
#include "include/potion.h"
#include "include/random.h"
#include "include/room.h"
#include "include/stairs.h"
#include "include/terrain.h"
#include "include/tiles.h"
#include "include/tunnel.h"
```

Include dependency graph for level.cpp:



### 5.44.1 Detailed Description

Member definitions for the Level class.

**Author**

Team Rogue++

**Date**

November 13, 2016

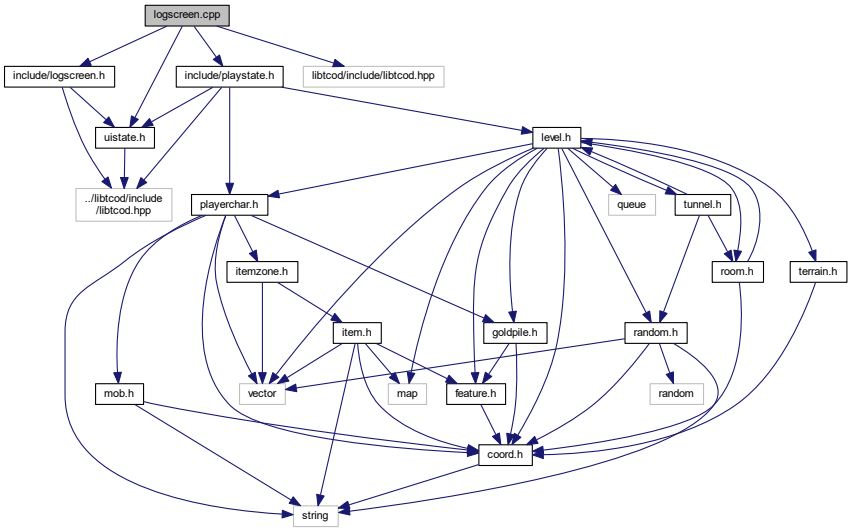
## 5.45 logscreen.cpp File Reference

Member definitions for the LogScreen class.

```
#include "include/logscreen.h"
#include "include/playstate.h"
#include "include/uistate.h"
```

```
#include "libtcod/include/libtcod.hpp"
```

Include dependency graph for logscreen.cpp:




### 5.45.1 Detailed Description

Member definitions for the LogScreen class.

#### Author

Team Rogue++

#### Date

November 13, 2016

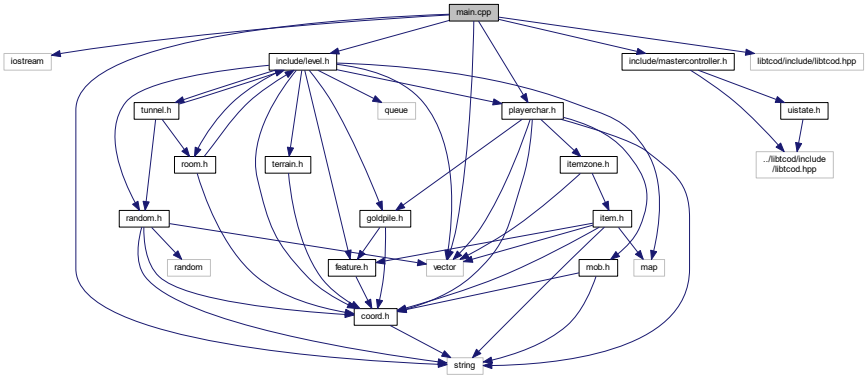
## 5.46 main.cpp File Reference

Global members.

```
#include <iostream>
#include <string>
#include <vector>
#include "include/level.h"
#include "include/mastercontroller.h"
#include "include/playerchar.h"
```

```
#include "libtcod/include/libtcod.hpp"
```

Include dependency graph for main.cpp:




### Typedefs

- using **uint** = unsigned int

### Functions

- void **putString** (int x, int y, std::string text)
- int main (int argv, char **args)

  *Execution starts here.*

#### 5.46.1 Detailed Description

Global members.

**Author**

Team Rogue++

**Date**

November 13, 2016

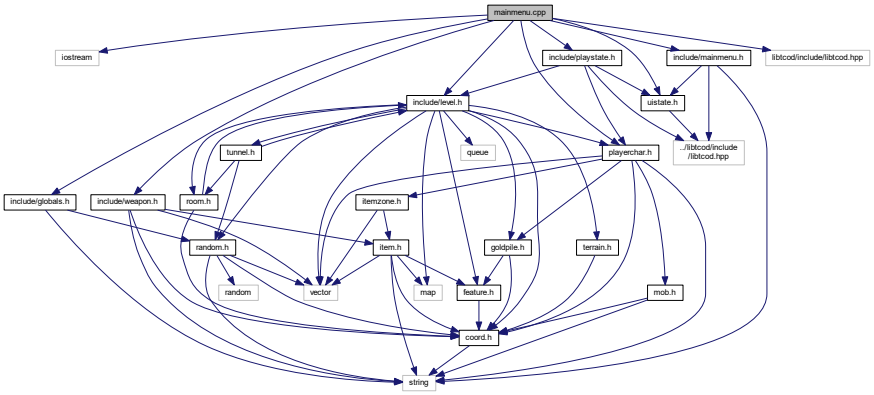
## 5.47 mainmenu.cpp File Reference

Member definitions for the MainMenu class.

```
#include <iostream>
#include "include/globals.h"
#include "include/level.h"
#include "include/mainmenu.h"
#include "include/playerchar.h"
```

```
#include "include/playstate.h"
#include "include/uistate.h"
#include "include/weapon.h"
#include "libtcod/include/libtcod.hpp"
```

Include dependency graph for mainmenu.cpp:




### 5.47.1 Detailed Description

Member definitions for the MainMenu class.

**Author**

Team Rogue++

**Date**

November 13, 2016

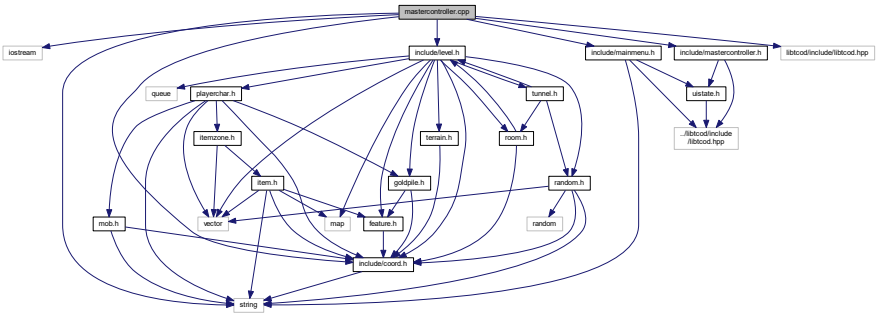
## 5.48 mastercontroller.cpp File Reference

Member definitions for the MasterController class.

```
#include <iostream>
#include <string>
#include "include/coord.h"
#include "include/level.h"
#include "include/mainmenu.h"
#include "include/mastercontroller.h"
```

```
#include "libtcod/include/libtcod.hpp"
```

Include dependency graph for mastercontroller.cpp:




### 5.48.1 Detailed Description

Member definitions for the MasterController class.

**Author**

Team Rogue++

**Date**

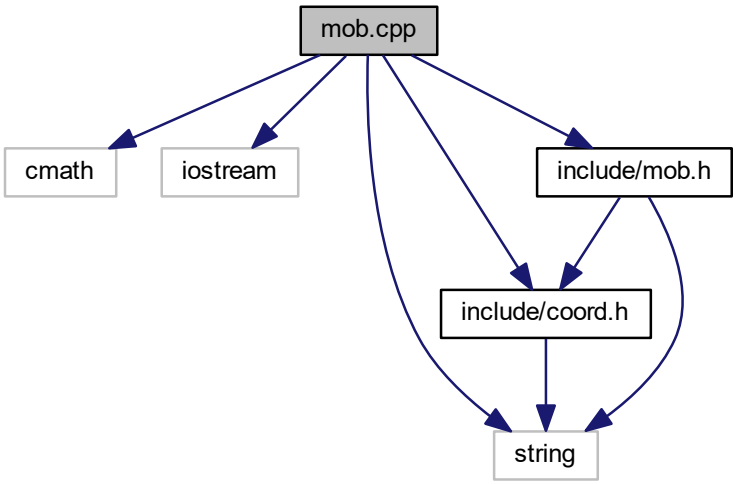
November 13, 2016

## 5.49 mob.cpp File Reference

Member definitions for the Mob class.

```
#include <cmath>
#include <iostream>
#include <string>
#include "include/coord.h"
#include "include/mob.h"
```

Include dependency graph for mob.cpp:




### 5.49.1 Detailed Description

Member definitions for the Mob class.

**Author**

Team Rogue++

**Date**

November 13, 2016

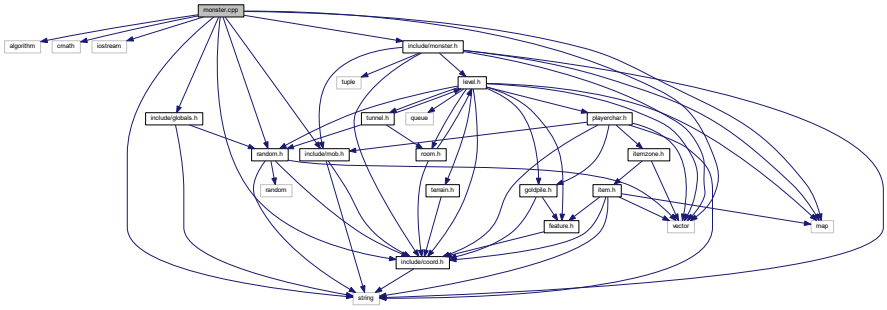
## 5.50 monster.cpp File Reference

Member definitions for the Monster class.

```
#include <algorithm>
#include <cmath>
#include <iostream>
#include <map>
#include <string>
#include <vector>
#include "include/coord.h"
#include "include/globals.h"
#include "include/mob.h"
#include "include/monster.h"
```

```
#include "include/random.h"
```

Include dependency graph for monster.cpp:




### 5.50.1 Detailed Description

Member definitions for the Monster class.

**Author**

Team Rogue++

**Date**

November 13, 2016

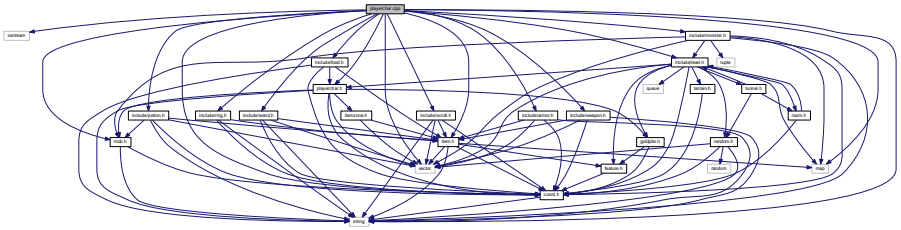
## 5.51 playerchar.cpp File Reference

Member definitions for the PlayerChar class.

```
#include <iostream>
#include <map>
#include <string>
#include <vector>
#include "include/armor.h"
#include "include/coord.h"
#include "include/food.h"
#include "include/item.h"
#include "include/level.h"
#include "include/mob.h"
#include "include/monster.h"
#include "include/playerchar.h"
#include "include/potion.h"
#include "include/ring.h"
#include "include/scroll.h"
#include "include/wand.h"
```

```
#include "include/weapon.h"
```

Include dependency graph for playerchar.cpp:



### 5.51.1 Detailed Description

Member definitions for the PlayerChar class.

**Author**

Team Rogue++

**Date**

November 13, 2016

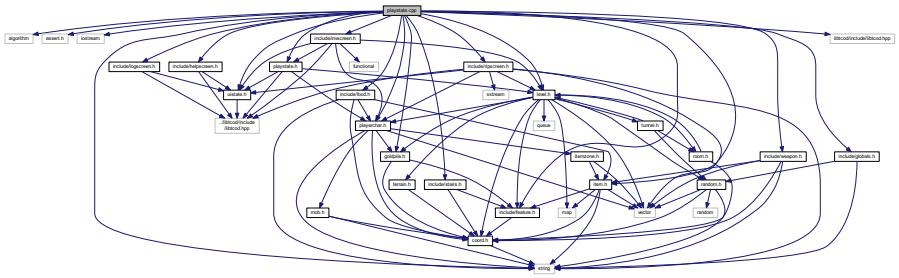
## 5.52 playstate.cpp File Reference

Member definitions for the PlayState class.

```
#include <algorithm>
#include <assert.h>
#include <iostream>
#include <string>
#include "include/feature.h"
#include "include/food.h"
#include "include/globals.h"
#include "include/goldpile.h"
#include "include/helpscreen.h"
#include "include/invscreen.h"
#include "include/item.h"
#include "include/level.h"
#include "include/logscreen.h"
#include "include/playerchar.h"
#include "include/playstate.h"
#include "include/ripscreen.h"
#include "include/stairs.h"
#include "include/uistate.h"
#include "include/weapon.h"
```

```
#include "libtcod/include/libtcod.hpp"
```

Include dependency graph for playstate.cpp:



## Classes

- class QuitPrompt2
- class QuickDrop
- class QuickThrow
- class QuickEat
- class ThrowDirectionState

### 5.52.1 Detailed Description

Member definitions for the PlayState class.

**Author**

Team Rogue++

**Date**

November 13, 2016

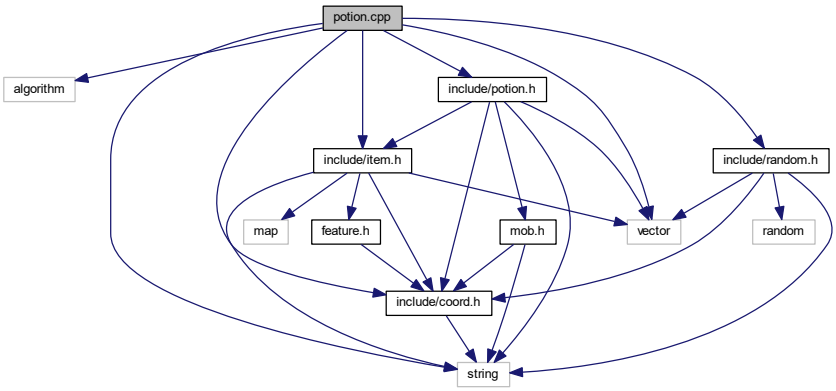
## 5.53 potion.cpp File Reference

Member definitions for the Potion class.

```
#include <algorithm>
#include <string>
#include <vector>
#include "include/coord.h"
#include "include/item.h"
#include "include/potion.h"
```

```
#include "include/random.h"
```

Include dependency graph for potion.cpp:




### 5.53.1 Detailed Description

Member definitions for the Potion class.

**Author**

Team Rogue++

**Date**

November 13, 2016

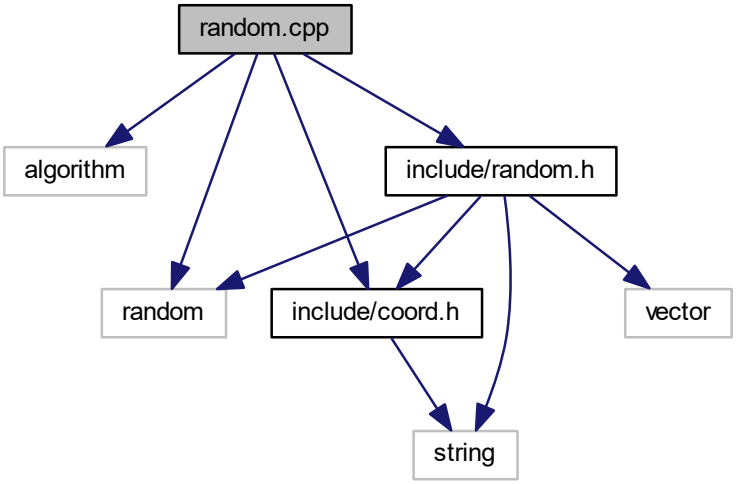
## 5.54 random.cpp File Reference

Global members.

```
#include <algorithm>
#include <random>
#include "include/coord.h"
```

```
#include "include/random.h"
```

Include dependency graph for random.cpp:




### 5.54.1 Detailed Description

Global members.

**Author**

Team Rogue++

**Date**

November 13, 2016

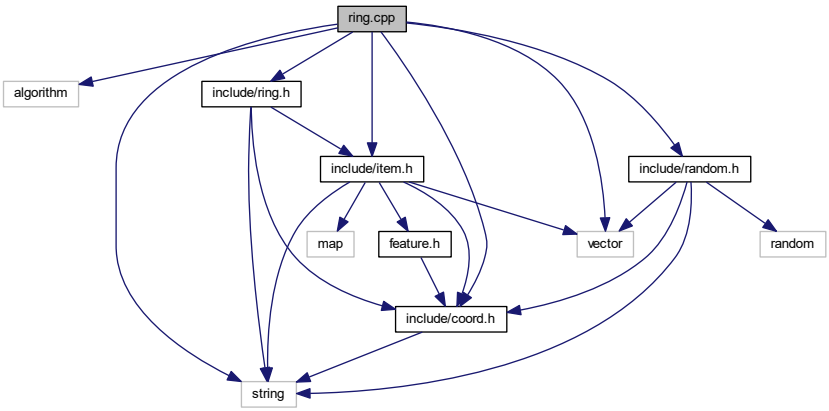
## 5.55 ring.cpp File Reference

Member definitions for the Ring class.

```
#include <algorithm>
#include <string>
#include <vector>
#include "include/coord.h"
#include "include/item.h"
#include "include/random.h"
```

```
#include "include/ring.h"
```

Include dependency graph for ring.cpp:




### 5.55.1 Detailed Description

Member definitions for the Ring class.

**Author**

Team Rogue++

**Date**

November 13, 2016

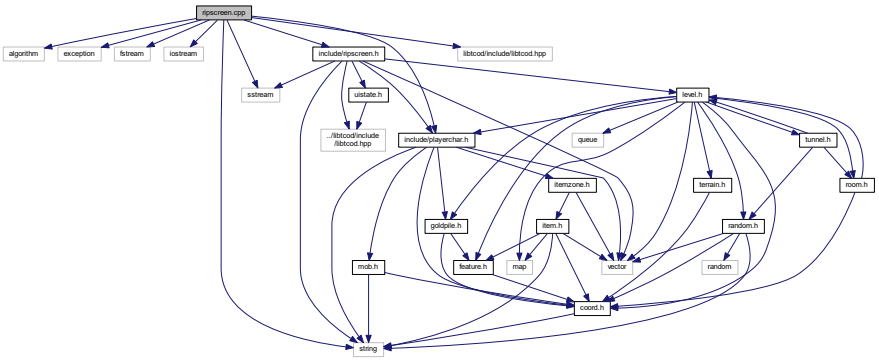
## 5.56 ripscreen.cpp File Reference

Member definitions for the RIPScreen class.

```
#include <algorithm>
#include <exception>
#include <fstream>
#include <iostream>
#include <sstream>
#include <string>
#include "include/playerchar.h"
#include "include/ripscreen.h"
```

```
#include "libtcod/include/libtcod.hpp"
```

Include dependency graph for ripscreen.cpp:




## Classes

- struct ScoreItem

### 5.56.1 Detailed Description

Member definitions for the RIPScreen class.

**Author**

Team Rogue++

**Date**

November 13, 2016

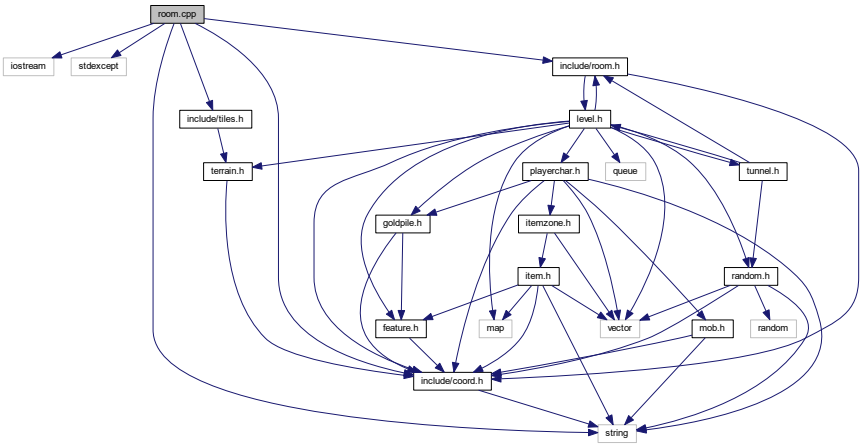
## 5.57 room.cpp File Reference

Member definitions for the Room class.

```
#include <iostream>
#include <stdexcept>
#include <string>
#include "include/coord.h"
#include "include/room.h"
```

```
#include "include/tiles.h"
```

Include dependency graph for room.cpp:




### 5.57.1 Detailed Description

Member definitions for the Room class.

**Author**

Team Rogue++

**Date**

November 13, 2016

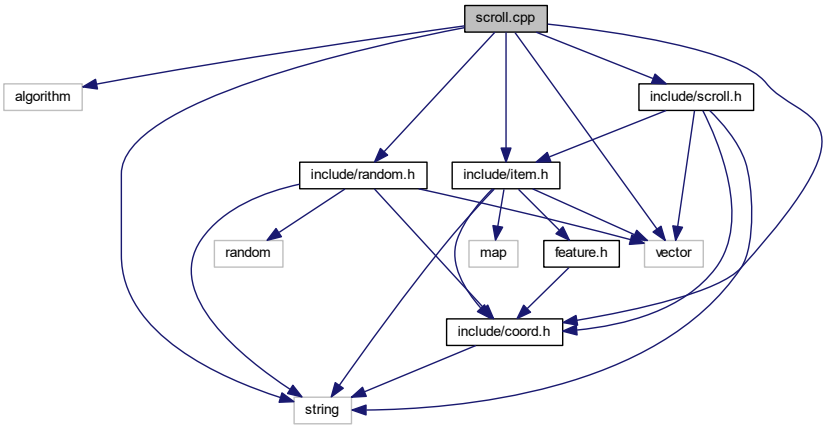
## 5.58 scroll.cpp File Reference

Member definitions for the Scroll class.

```
#include <algorithm>
#include <string>
#include <vector>
#include "include/coord.h"
#include "include/item.h"
#include "include/random.h"
```

```
#include "include/scroll.h"
```

Include dependency graph for scroll.cpp:




### 5.58.1 Detailed Description

Member definitions for the Scroll class.

**Author**

Team Rogue++

**Date**

November 13, 2016

## 5.59 Source_Formatter.py File Reference

Performs several formatting operations over the C++ header and source files.

### Functions

- def Source_Formatter.cleanPragmas (content)

  *Removes all current 'pragma once' lines and define guards of the given C++ file; inserts a 'pragma once' into the file.*
- def Source_Formatter.sortIncludes (content)

  *Sorts the 'include' statements of the given C++ file.*
- def Source_Formatter.trim (content)

  *Trims the given C++ file.*
- def Source_Formatter.addHeader (cppFile, content)

  *Adds a header to the given C++ file.*
- def Source_Formatter.formatContent (cppFile, content)

  *Formats the content of the given C++ source file.*
- def Source_Formatter.formatFiles (cppFiles)

  *Formats all of the given C++ source files.*
- def Source_Formatter.findFiles ()

  *Recursively finds all C++ source files.*
- def Source_Formatter.main ()

  *Execution entry point.*

### Variables

- Source_Formatter.RE_PATH_IGNORE = re.compile(r"libtcod|ParseTest|html")

  *Ignored paths.*
- Source_Formatter.RE_EXTENSION = re.compile(r"\.(cpp|h)")

  *C++ file extensions.*
- Source_Formatter.RE_HEADER_EXTENSION = re.compile(r"\.h$")

  *C++ header file.*
- Source_Formatter.RE_HEADER_CLASS = re.compile(r"class\s+(?P<className>[a-zA-Z]+)\s+(:|{)")

  *C++ header class declaration.*
- Source_Formatter.RE_SRC_CLASS = re.compile(r"^(?P<className>[a-zA-Z]+)::\1")

  *C++ source class declaration.*

## 5.59.1 Detailed Description

Performs several formatting operations over the C++ header and source files.

**Author**

Mikhail Andrenkov

## 5.59.2 Function Documentation

### 5.59.2.1 addHeader()

```
def Source_Formatter.addHeader (
              cppFile,
              content )
```

Adds a header to the given C++ file.

**Parameters**

| | |
|---|---|
| *cppFile* | The name of the C++ file |
| *content* | The content of the C++ file |

**Returns**

A list denoting the formatted contents of the C++ file

### 5.59.2.2 cleanPragmas()

```
def Source_Formatter.cleanPragmas (
              content )
```

Removes all current 'pragma once' lines and define guards of the given C++ file; inserts a 'pragma once' into the file.

**Parameters**

| | |
|---|---|
| *content* | The content of the C++ file |

**Returns**

A list denoting the formatted contents of the C++ file

#### 5.59.2.3 formatContent()

```
def Source_Formatter.formatContent (
              cppFile,
              content )
```

Formats the content of the given C++ source file.

**Parameters**

| | |
|---|---|
| *cppFile* | The name of the C++ file |
| *content* | The content of the C++ file |

**Returns**

A list denoting the formatted contents of the C++ file

#### 5.59.2.4 formatFiles()

```
def Source_Formatter.formatFiles (
              cppFiles )
```

Formats all of the given C++ source files.

**Parameters**

| | |
|---|---|
| *cppFiles* | The C++ source files |

#### 5.59.2.5 sortIncludes()

```
def Source_Formatter.sortIncludes (
              content )
```

Sorts the 'include' statements of the given C++ file.

**Parameters**

| | |
|---|---|
| *content* | The content of the C++ file |

**Returns**

A list denoting the formatted contents of the C++ file

#### 5.59.2.6 trim()

```
def Source_Formatter.trim (
              content )
```

Trims the given C++ file.

**Parameters**

| | |
|---|---|
| *content* | The content of the C++ file |

**Returns**

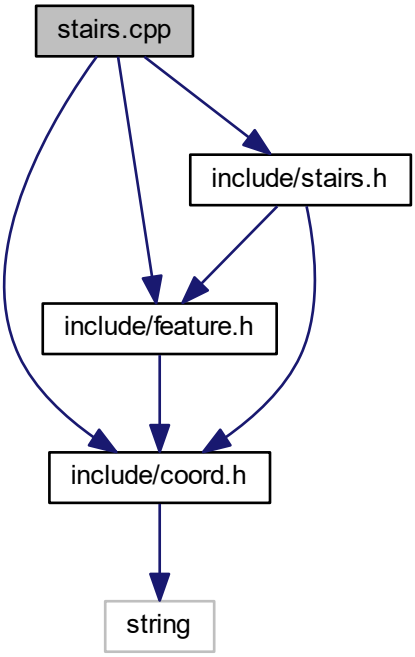
A list denoting the formatted contents of the C++ file

## 5.60 stairs.cpp File Reference

Member definitions for the Stairs class.

```
#include "include/coord.h"
#include "include/feature.h"
#include "include/stairs.h"
```

Include dependency graph for stairs.cpp:



### 5.60.1 Detailed Description

Member definitions for the Stairs class.

**Author**

Team Rogue++

**Date**

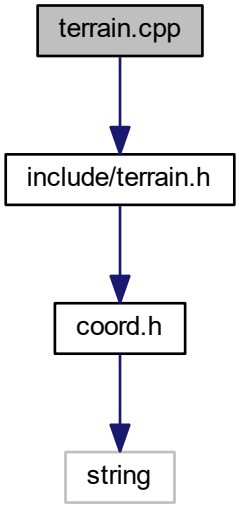
November 13, 2016

## 5.61 terrain.cpp File Reference

Member definitions for the Terrain class.

```
#include "include/terrain.h"
```

Include dependency graph for terrain.cpp:




### 5.61.1 Detailed Description

Member definitions for the Terrain class.

**Author**

Team Rogue++

**Date**

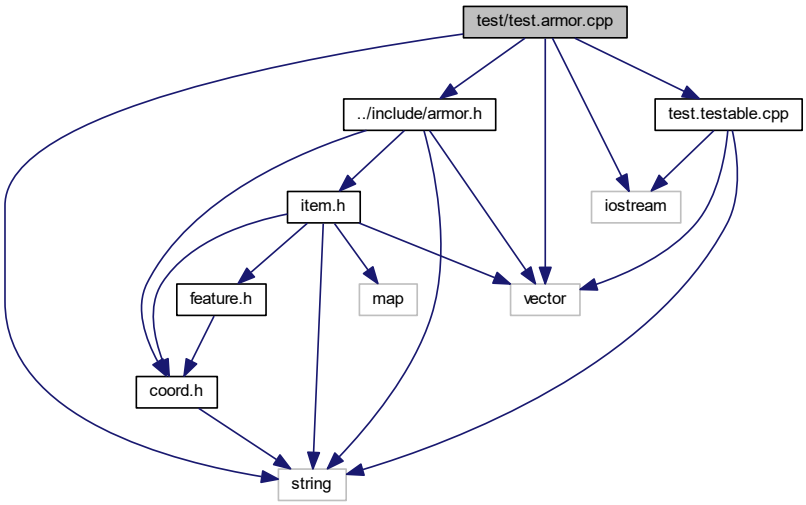
November 13, 2016

## 5.62 test/test.armor.cpp File Reference

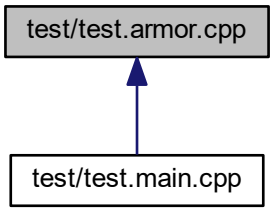
Global members.

```
#include <iostream>
#include <string>
#include <vector>
#include "../include/armor.h"
#include "test.testable.cpp"
```

Include dependency graph for test.armor.cpp:




This graph shows which files directly or indirectly include this file:




### Classes

- class ArmorTest

### 5.62.1 Detailed Description

Global members.

**Author**

Team Rogue++

**Date**

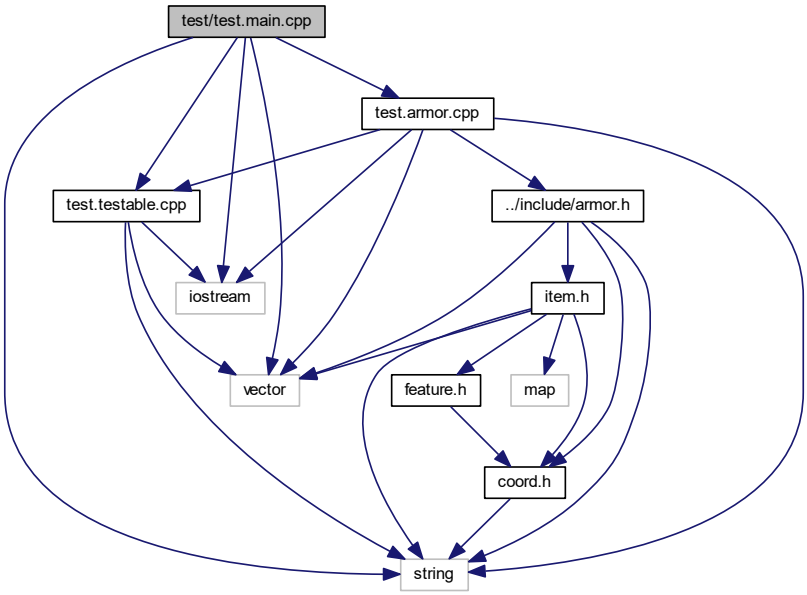
November 13, 2016

## 5.63 test/test.main.cpp File Reference

Global members.

```
#include <iostream>
#include <string>
#include <vector>
#include "test.armor.cpp"
#include "test.testable.cpp"
```

Include dependency graph for test.main.cpp:




## Functions

- int **main** ()

### 5.63.1 Detailed Description

Global members.

**Author**

Team Rogue++

**Date**

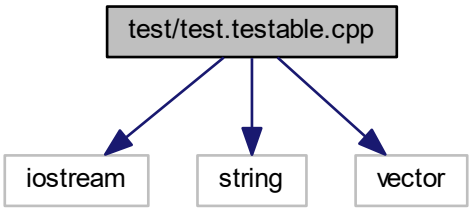
November 13, 2016

## 5.64 test/test.testable.cpp File Reference

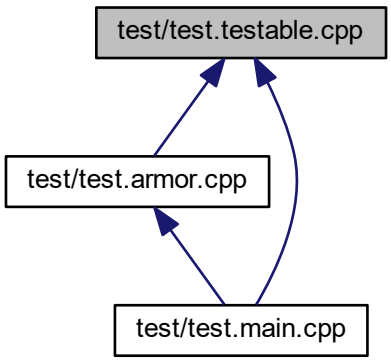
Global members.

```
#include <iostream>
#include <string>
#include <vector>
```

Include dependency graph for test.testable.cpp:




This graph shows which files directly or indirectly include this file:



### Classes

- class Testable

#### 5.64.1 Detailed Description

Global members.

**Author**

Team Rogue++

**Date**

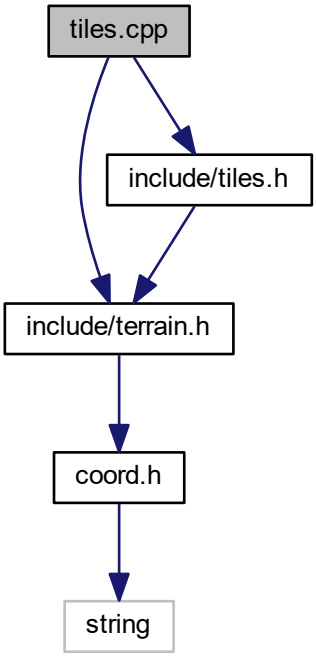
November 13, 2016

## 5.65 tiles.cpp File Reference

Member definitions for the Corridor, Door, Floor, Wall classes.

```
#include "include/terrain.h"
#include "include/tiles.h"
```

Include dependency graph for tiles.cpp:



### 5.65.1 Detailed Description

Member definitions for the Corridor, Door, Floor, Wall classes.

**Author**

Team Rogue++

**Date**

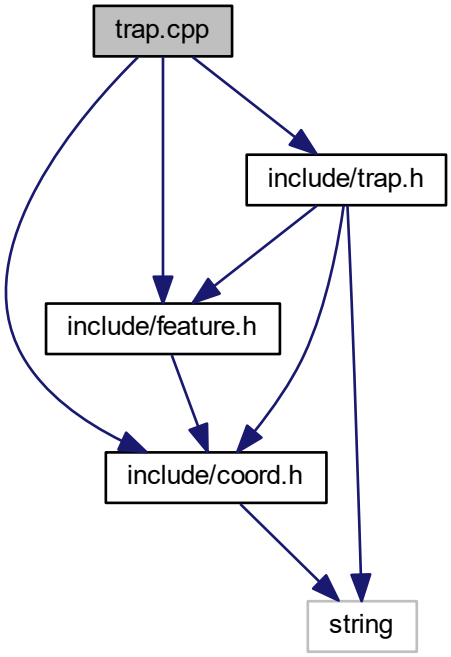
November 13, 2016

## 5.66 trap.cpp File Reference

Member definitions for the Trap class.

```
#include "include/coord.h"
#include "include/feature.h"
#include "include/trap.h"
```

Include dependency graph for trap.cpp:



### 5.66.1 Detailed Description

Member definitions for the Trap class.

**Author**

Team Rogue++

**Date**

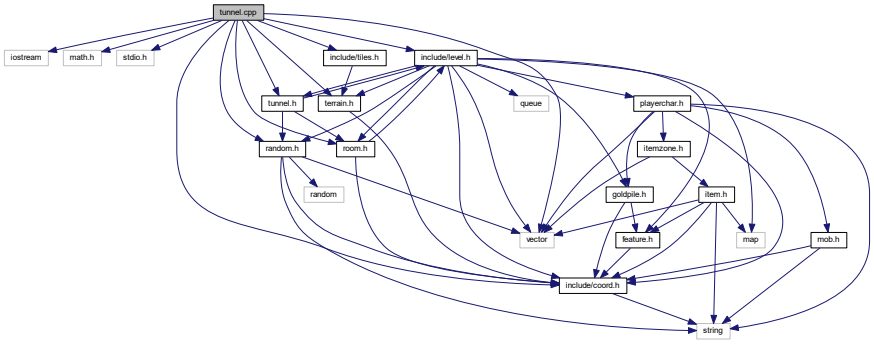
November 13, 2016

## 5.67 tunnel.cpp File Reference

Member definitions for the Tunnel class.

```
#include <iostream>
#include <math.h>
#include <stdio.h>
#include <vector>
#include "include/coord.h"
#include "include/level.h"
#include "include/random.h"
#include "include/room.h"
#include "include/terrain.h"
#include "include/tiles.h"
#include "include/tunnel.h"
```

Include dependency graph for tunnel.cpp:




### 5.67.1 Detailed Description

Member definitions for the Tunnel class.

**Author**

Team Rogue++

**Date**

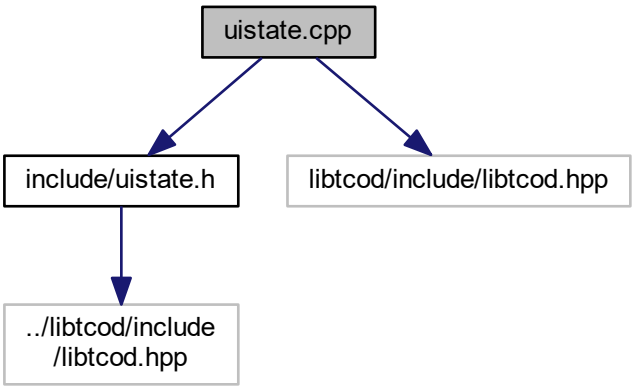
November 13, 2016

## 5.68 uistate.cpp File Reference

Member definitions for the UIState class.

```
#include "include/uistate.h"
#include "libtcod/include/libtcod.hpp"
```

Include dependency graph for uistate.cpp:




### 5.68.1 Detailed Description

Member definitions for the UIState class.

**Author**

Team Rogue++

**Date**

November 13, 2016

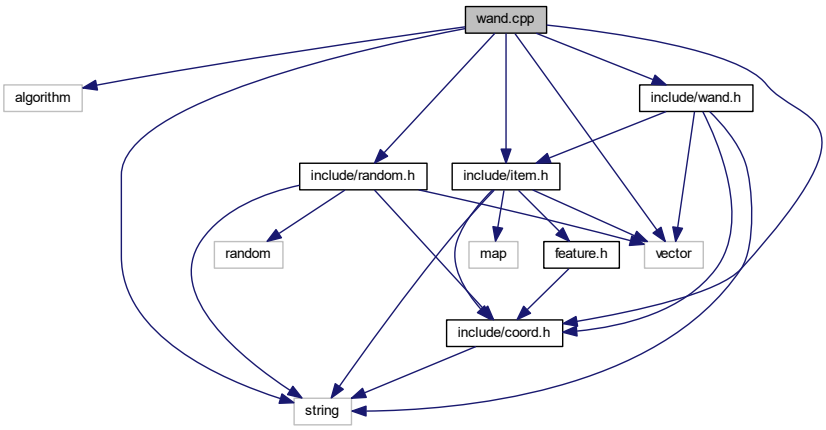
## 5.69 wand.cpp File Reference

Member definitions for the Wand class.

```
#include <algorithm>
#include <string>
#include <vector>
#include "include/coord.h"
#include "include/item.h"
#include "include/random.h"
```

```
#include "include/wand.h"
```

Include dependency graph for wand.cpp:




### 5.69.1 Detailed Description

Member definitions for the Wand class.

**Author**

Team Rogue++

**Date**

November 13, 2016

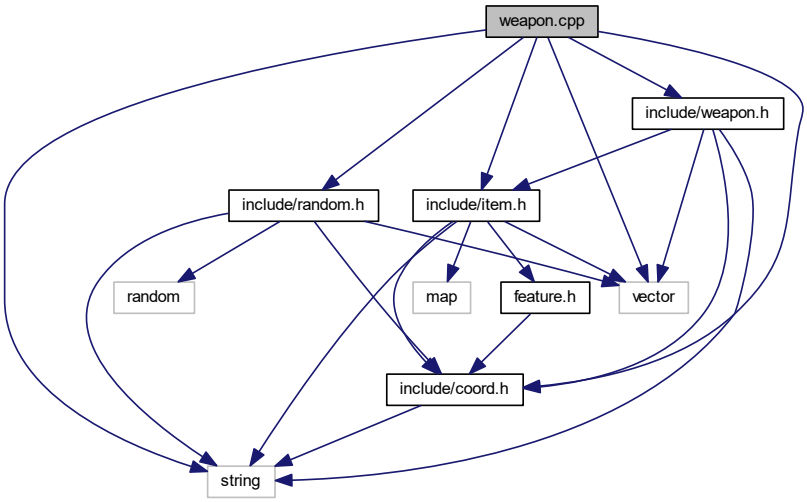
## 5.70 weapon.cpp File Reference

Member definitions for the Weapon class.

```
#include <string>
#include <vector>
#include "include/coord.h"
#include "include/item.h"
#include "include/random.h"
```

```
#include "include/weapon.h"
```

Include dependency graph for weapon.cpp:




### 5.70.1 Detailed Description

Member definitions for the Weapon class.

**Author**

Team Rogue++

**Date**

November 13, 2016