

# Rogue Reborn

Generated by Doxygen 1.8.12



# Contents

<b>1</b>	<b>Hierarchical Index</b>	<b>1</b>
1.1	Class Hierarchy	1
<b>2</b>	<b>Class Index</b>	<b>3</b>
2.1	Class List	3
<b>3</b>	<b>File Index</b>	<b>7</b>
3.1	File List	7
<b>4</b>	<b>Class Documentation</b>	<b>13</b>
4.1	Amulet Class Reference	13
4.1.1	Detailed Description	14
4.1.2	Constructor & Destructor Documentation	14
4.1.2.1	Amulet()	14
4.2	AmuletTest Class Reference	15
4.3	Armor Class Reference	16
4.3.1	Detailed Description	17
4.3.2	Constructor & Destructor Documentation	17
4.3.2.1	Armor() [1/2]	17
4.3.2.2	Armor() [2/2]	17
4.3.3	Member Function Documentation	18
4.3.3.1	getEnchantment()	18
4.3.3.2	getRating()	18
4.3.3.3	setEnchantment()	18
4.4	ArmorTest Class Reference	19

4.5	Coord Class Reference . . . . .	19
4.5.1	Detailed Description . . . . .	21
4.5.2	Constructor & Destructor Documentation . . . . .	21
4.5.2.1	Coord() [1/2] . . . . .	21
4.5.2.2	Coord() [2/2] . . . . .	21
4.5.3	Member Function Documentation . . . . .	21
4.5.3.1	asScreen() . . . . .	21
4.5.3.2	copy() . . . . .	21
4.5.3.3	isAdjacentTo() . . . . .	21
4.5.3.4	operator"!==" . . . . .	22
4.5.3.5	operator*() . . . . .	22
4.5.3.6	operator*==() . . . . .	22
4.5.3.7	operator+() . . . . .	22
4.5.3.8	operator+=() . . . . .	22
4.5.3.9	operator-() . . . . .	22
4.5.3.10	operator-=() . . . . .	22
4.5.3.11	operator<() . . . . .	23
4.5.3.12	operator==() . . . . .	23
4.5.3.13	operator[]() . . . . .	23
4.5.3.14	toString() . . . . .	23
4.5.4	Member Data Documentation . . . . .	23
4.5.4.1	ORTHO . . . . .	23
4.6	CoordTest Class Reference . . . . .	24
4.7	Corridor Class Reference . . . . .	24
4.7.1	Detailed Description . . . . .	25
4.8	DirectionPrompt Class Reference . . . . .	26
4.8.1	Member Function Documentation . . . . .	27
4.8.1.1	handleInput() . . . . .	27
4.9	Door Class Reference . . . . .	27
4.9.1	Detailed Description . . . . .	28

4.10 Feature Class Reference . . . . .	28
4.10.1 Detailed Description . . . . .	29
4.10.2 Constructor & Destructor Documentation . . . . .	29
4.10.2.1 Feature() . . . . .	29
4.10.2.2 ~Feature() . . . . .	30
4.10.3 Member Function Documentation . . . . .	30
4.10.3.1 getFColor() . . . . .	30
4.10.3.2 getLocation() . . . . .	30
4.10.3.3 getSymbol() . . . . .	30
4.10.3.4 getVisible() . . . . .	31
4.10.3.5 setLocation() . . . . .	31
4.10.3.6 setVisible() . . . . .	31
4.10.4 Member Data Documentation . . . . .	31
4.10.4.1 fcolor . . . . .	31
4.10.4.2 possibleColors . . . . .	31
4.10.4.3 visible . . . . .	32
4.11 FeatureTest Class Reference . . . . .	32
4.12 Floor Class Reference . . . . .	33
4.12.1 Detailed Description . . . . .	33
4.13 Food Class Reference . . . . .	34
4.13.1 Detailed Description . . . . .	35
4.13.2 Constructor & Destructor Documentation . . . . .	35
4.13.2.1 Food() . . . . .	35
4.13.3 Member Function Documentation . . . . .	35
4.13.3.1 activate() . . . . .	35
4.14 FoodTest Class Reference . . . . .	36
4.15 Generator Class Reference . . . . .	36
4.15.1 Detailed Description . . . . .	37
4.15.2 Member Function Documentation . . . . .	37
4.15.2.1 intFromRange() . . . . .	37

4.15.2.2	nDx()	37
4.15.2.3	rand()	37
4.15.2.4	randBool()	38
4.16	GoldPile Class Reference	38
4.16.1	Detailed Description	39
4.16.2	Constructor & Destructor Documentation	39
4.16.2.1	GoldPile()	39
4.16.3	Member Function Documentation	39
4.16.3.1	getQuantity()	39
4.17	GoldPileTest Class Reference	40
4.18	HelpScreen Class Reference	40
4.18.1	Detailed Description	41
4.18.2	Constructor & Destructor Documentation	42
4.18.2.1	HelpScreen()	42
4.18.3	Member Function Documentation	42
4.18.3.1	draw()	42
4.18.3.2	handleInput()	42
4.19	InvScreen Class Reference	42
4.19.1	Detailed Description	43
4.19.2	Constructor & Destructor Documentation	44
4.19.2.1	InvScreen()	44
4.19.3	Member Function Documentation	44
4.19.3.1	draw()	44
4.19.3.2	handleInput()	44
4.20	Item Class Reference	45
4.20.1	Detailed Description	47
4.20.2	Constructor & Destructor Documentation	47
4.20.2.1	Item() [1/2]	47
4.20.2.2	Item() [2/2]	48
4.20.3	Member Function Documentation	48

---

4.20.3.1 applyEffect() . . . . .	48
4.20.3.2 getClassName() . . . . .	49
4.20.3.3 getContext() . . . . .	49
4.20.3.4 getDisplayName() . . . . .	49
4.20.3.5 getName() . . . . .	49
4.20.3.6 getType() . . . . .	50
4.20.3.7 getWeight() . . . . .	50
4.20.3.8 hasEffect() . . . . .	50
4.20.3.9 isCursed() . . . . .	50
4.20.3.10 isIdentified() . . . . .	51
4.20.3.11 isStackable() . . . . .	51
4.20.3.12 isThrowable() . . . . .	51
4.20.3.13 operator<() . . . . .	51
4.20.3.14 operator==() . . . . .	52
4.20.3.15 removeEffect() . . . . .	52
4.20.3.16 setContext() . . . . .	52
4.20.3.17 setIdentified() . . . . .	52
4.20.3.18 shuffleNameVector() . . . . .	53
4.21 ItemTest Class Reference . . . . .	53
4.22 ItemZone Class Reference . . . . .	54
4.22.1 Detailed Description . . . . .	54
4.22.2 Constructor & Destructor Documentation . . . . .	55
4.22.2.1 ItemZone() . . . . .	55
4.22.3 Member Function Documentation . . . . .	55
4.22.3.1 add() . . . . .	55
4.22.3.2 contains() [1/2] . . . . .	55
4.22.3.3 contains() [2/2] . . . . .	55
4.22.3.4 getContents() . . . . .	55
4.22.3.5 getCurrWeight() . . . . .	55
4.22.3.6 getItem() . . . . .	56

4.22.3.7	getMaxWeight()	56
4.22.3.8	getSize()	56
4.23	ItemZoneTest Class Reference	56
4.24	Level Class Reference	57
4.24.1	Member Function Documentation	58
4.24.1.1	addFeature()	58
4.24.1.2	bfsDiag()	58
4.24.1.3	bfsPerp()	59
4.24.1.4	canSee()	59
4.24.1.5	getAdjPassable()	59
4.24.1.6	getFeatures()	60
4.24.1.7	getMobs()	60
4.24.1.8	getNearestGold()	60
4.24.1.9	getRandomEmptyPosition()	61
4.24.1.10	getRooms()	61
4.24.1.11	monsterAt()	61
4.24.1.12	popTurnClock()	61
4.24.1.13	pushMob()	62
4.24.1.14	registerMob()	63
4.24.1.15	removeFeature()	63
4.24.1.16	removeMob()	63
4.24.1.17	throwLocation()	63
4.25	LevelGenTest Class Reference	64
4.26	LevelTest Class Reference	65
4.27	LogScreen Class Reference	66
4.27.1	Detailed Description	67
4.27.2	Member Function Documentation	67
4.27.2.1	draw()	67
4.27.2.2	handleInput()	67
4.28	MainMenu Class Reference	68

4.28.1 Detailed Description . . . . .	68
4.28.2 Constructor & Destructor Documentation . . . . .	69
4.28.2.1 MainMenu() . . . . .	69
4.28.3 Member Function Documentation . . . . .	69
4.28.3.1 draw() . . . . .	69
4.28.3.2 handleInput() . . . . .	69
4.29 MasterController Class Reference . . . . .	69
4.29.1 Detailed Description . . . . .	70
4.30 Mob Class Reference . . . . .	70
4.30.1 Detailed Description . . . . .	72
4.30.2 Constructor & Destructor Documentation . . . . .	72
4.30.2.1 Mob() [1/2] . . . . .	72
4.30.2.2 Mob() [2/2] . . . . .	72
4.30.2.3 ~Mob() . . . . .	73
4.30.3 Member Function Documentation . . . . .	73
4.30.3.1 changeArmor() . . . . .	73
4.30.3.2 getArmorRating() . . . . .	73
4.30.3.3 getDelay() . . . . .	73
4.30.3.4 getExperience() . . . . .	73
4.30.3.5 getFColor() . . . . .	74
4.30.3.6 getHP() . . . . .	74
4.30.3.7 getLevel() . . . . .	74
4.30.3.8 getLocation() . . . . .	74
4.30.3.9 getMaxHP() . . . . .	74
4.30.3.10 getName() . . . . .	75
4.30.3.11 getSymbol() . . . . .	75
4.30.3.12 hit() . . . . .	75
4.30.3.13 isDead() . . . . .	75
4.30.3.14 moveLocation() . . . . .	76
4.30.3.15 setCurrentHP() . . . . .	76

4.30.3.16 setFColor() . . . . .	76
4.30.3.17 setLocation() . . . . .	76
4.30.3.18 setMaxHP() . . . . .	77
4.30.4 Member Data Documentation . . . . .	77
4.30.4.1 armor . . . . .	77
4.30.4.2 exp . . . . .	77
4.30.4.3 fcolor . . . . .	77
4.30.4.4 level . . . . .	77
4.30.4.5 location . . . . .	77
4.30.4.6 maxHP . . . . .	77
4.30.4.7 name . . . . .	78
4.31 MobTest Class Reference . . . . .	78
4.32 Monster Class Reference . . . . .	79
4.32.1 Detailed Description . . . . .	81
4.32.2 Constructor & Destructor Documentation . . . . .	81
4.32.2.1 Monster() . . . . .	81
4.32.3 Member Function Documentation . . . . .	81
4.32.3.1 addFlag() . . . . .	81
4.32.3.2 addFrozenTurns() . . . . .	83
4.32.3.3 attack() . . . . .	83
4.32.3.4 attackConfuse() . . . . .	83
4.32.3.5 attackDrainLife() . . . . .	83
4.32.3.6 attackDropLevel() . . . . .	84
4.32.3.7 attackFreeze() . . . . .	84
4.32.3.8 attackRust() . . . . .	84
4.32.3.9 attackSteal() . . . . .	84
4.32.3.10 attackSting() . . . . .	85
4.32.3.11 calculateDamage() . . . . .	85
4.32.3.12 calculateHitChance() . . . . .	85
4.32.3.13 getArmorRating() . . . . .	85

4.32.3.14 getCarryChance() . . . . .	86
4.32.3.15 getDelay() . . . . .	86
4.32.3.16 getName() [1/2] . . . . .	86
4.32.3.17 getName() [2/2] . . . . .	86
4.32.3.18 getSymbolsForLevel() . . . . .	86
4.32.3.19 getSymbolsForTreasure() . . . . .	87
4.32.3.20 hit() . . . . .	87
4.32.3.21 isAwake() . . . . .	87
4.32.3.22 isVisible() . . . . .	88
4.32.3.23 randomMonster() . . . . .	88
4.32.3.24 removeFlag() . . . . .	88
4.32.3.25 setAwake() . . . . .	88
4.32.3.26 setVisible() . . . . .	88
4.32.3.27 turn() . . . . .	89
4.33 MonsterTest Class Reference . . . . .	89
4.34 PlayerChar Class Reference . . . . .	90
4.34.1 Detailed Description . . . . .	94
4.34.2 Constructor & Destructor Documentation . . . . .	94
4.34.2.1 PlayerChar() . . . . .	94
4.34.3 Member Function Documentation . . . . .	94
4.34.3.1 activateItem() . . . . .	94
4.34.3.2 addExp() . . . . .	94
4.34.3.3 appendLog() . . . . .	95
4.34.3.4 applyCondition() . . . . .	95
4.34.3.5 attack() . . . . .	95
4.34.3.6 calculateDamage() . . . . .	95
4.34.3.7 calculateHitChance() . . . . .	96
4.34.3.8 changeCurrentHP() . . . . .	96
4.34.3.9 changeCurrentStrength() . . . . .	96
4.34.3.10 changeFoodLife() . . . . .	96

4.34.3.11 changeMaxStrength()	97
4.34.3.12 collectGold()	97
4.34.3.13 dropItem()	97
4.34.3.14 eat()	97
4.34.3.15 equipArmor()	98
4.34.3.16 equipRingLeft()	98
4.34.3.17 equipRingRight()	98
4.34.3.18 equipWeapon()	98
4.34.3.19 getArmor()	99
4.34.3.20 getDelay()	99
4.34.3.21 getDexterity()	99
4.34.3.22 getFoodLife()	99
4.34.3.23 getFoodStatus()	100
4.34.3.24 getGold()	100
4.34.3.25 getInventory()	100
4.34.3.26 getLevel()	100
4.34.3.27 getLog()	100
4.34.3.28 getMaxStrength()	101
4.34.3.29 getRings()	101
4.34.3.30 getSaveFlag()	101
4.34.3.31 getSearchRadius()	101
4.34.3.32 getSightRadius()	102
4.34.3.33 getStrength()	102
4.34.3.34 getWeapon()	102
4.34.3.35 hasAmulet()	102
4.34.3.36 hasCondition()	102
4.34.3.37 hit()	103
4.34.3.38 move()	103
4.34.3.39 pickupItem()	103
4.34.3.40 quaff()	104

4.34.3.41 removeArmor() . . . . .	104
4.34.3.42 removeCondition() . . . . .	104
4.34.3.43 removeRingLeft() . . . . .	104
4.34.3.44 removeRingRight() . . . . .	105
4.34.3.45 removeWeapon() . . . . .	105
4.34.3.46 setDexterity() . . . . .	105
4.34.3.47 setFoodLife() . . . . .	105
4.34.3.48 setGold() . . . . .	105
4.34.3.49 setSaveFlag() . . . . .	106
4.34.3.50 setStrength() . . . . .	106
4.34.3.51 update() . . . . .	106
4.35 PlayerCharTest Class Reference . . . . .	107
4.36 PlayState Class Reference . . . . .	108
4.36.1 Detailed Description . . . . .	110
4.36.2 Constructor & Destructor Documentation . . . . .	110
4.36.2.1 PlayState() . . . . .	110
4.36.2.2 ~PlayState() . . . . .	110
4.36.3 Member Function Documentation . . . . .	110
4.36.3.1 handleInput() . . . . .	110
4.36.4 Member Data Documentation . . . . .	110
4.36.4.1 level . . . . .	110
4.36.4.2 player . . . . .	111
4.37 Potion Class Reference . . . . .	111
4.37.1 Detailed Description . . . . .	112
4.37.2 Constructor & Destructor Documentation . . . . .	112
4.37.2.1 Potion() [1/2] . . . . .	112
4.37.2.2 Potion() [2/2] . . . . .	112
4.37.3 Member Function Documentation . . . . .	112
4.37.3.1 activate() . . . . .	113
4.38 PotionTest Class Reference . . . . .	113

4.39 QuickThrow Class Reference . . . . .	114
4.39.1 Member Function Documentation . . . . .	115
4.39.1.1 handleInput() . . . . .	115
4.40 QuickUse< T > Class Template Reference . . . . .	116
4.40.1 Member Function Documentation . . . . .	117
4.40.1.1 handleInput() . . . . .	117
4.41 QuickZap Class Reference . . . . .	117
4.41.1 Member Function Documentation . . . . .	118
4.41.1.1 handleInput() . . . . .	118
4.42 QuitPrompt2 Class Reference . . . . .	119
4.42.1 Member Function Documentation . . . . .	120
4.42.1.1 handleInput() . . . . .	120
4.43 RandomTest Class Reference . . . . .	120
4.44 Ring Class Reference . . . . .	121
4.44.1 Detailed Description . . . . .	122
4.44.2 Constructor & Destructor Documentation . . . . .	122
4.44.2.1 Ring() [1/2] . . . . .	122
4.44.2.2 Ring() [2/2] . . . . .	123
4.44.3 Member Function Documentation . . . . .	123
4.44.3.1 activate() . . . . .	123
4.44.3.2 deactivate() . . . . .	123
4.45 RingRemovePrompt Class Reference . . . . .	124
4.45.1 Member Function Documentation . . . . .	125
4.45.1.1 handleInput() . . . . .	125
4.46 RingTest Class Reference . . . . .	126
4.47 RIPScreen Class Reference . . . . .	126
4.47.1 Detailed Description . . . . .	127
4.47.2 Constructor & Destructor Documentation . . . . .	128
4.47.2.1 RIPScreen() . . . . .	128
4.47.3 Member Function Documentation . . . . .	128

4.47.3.1 draw() . . . . .	128
4.47.3.2 handleInput() . . . . .	128
4.48 Room Class Reference . . . . .	128
4.48.1 Detailed Description . . . . .	129
4.48.2 Member Function Documentation . . . . .	129
4.48.2.1 contains() . . . . .	129
4.48.2.2 dig() . . . . .	130
4.48.2.3 exists() . . . . .	130
4.48.2.4 printInfo() . . . . .	130
4.48.2.5 touches() . . . . .	130
4.49 RoomTest Class Reference . . . . .	131
4.50 SaveScreen Class Reference . . . . .	132
4.51 ScoreItem Struct Reference . . . . .	133
4.52 Scroll Class Reference . . . . .	134
4.52.1 Detailed Description . . . . .	135
4.52.2 Constructor & Destructor Documentation . . . . .	135
4.52.2.1 Scroll() [1/2] . . . . .	135
4.52.2.2 Scroll() [2/2] . . . . .	135
4.52.3 Member Function Documentation . . . . .	136
4.52.3.1 activate() . . . . .	136
4.52.3.2 initializeScrollNames() . . . . .	136
4.53 ScrollTest Class Reference . . . . .	137
4.54 Stairs Class Reference . . . . .	138
4.55 StairsTest Class Reference . . . . .	139
4.56 StatusScreen Class Reference . . . . .	139
4.56.1 Detailed Description . . . . .	140
4.56.2 Constructor & Destructor Documentation . . . . .	140
4.56.2.1 StatusScreen() . . . . .	140
4.56.3 Member Function Documentation . . . . .	141
4.56.3.1 draw() . . . . .	141

4.56.3.2 handleInput() . . . . .	141
4.57 SymbolScreen Class Reference . . . . .	141
4.57.1 Constructor & Destructor Documentation . . . . .	142
4.57.1.1 SymbolScreen() . . . . .	142
4.57.2 Member Function Documentation . . . . .	142
4.57.2.1 draw() . . . . .	142
4.57.2.2 handleInput() . . . . .	142
4.58 Terrain Class Reference . . . . .	143
4.58.1 Detailed Description . . . . .	144
4.58.2 Member Enumeration Documentation . . . . .	144
4.58.2.1 Mapped . . . . .	144
4.58.2.2 Passability . . . . .	145
4.58.3 Constructor & Destructor Documentation . . . . .	145
4.58.3.1 Terrain() [1/2] . . . . .	145
4.58.3.2 Terrain() [2/2] . . . . .	145
4.58.3.3 ~Terrain() . . . . .	145
4.58.4 Member Function Documentation . . . . .	145
4.58.4.1 getColor() . . . . .	145
4.58.4.2 getSymbol() . . . . .	146
4.58.4.3 getVisibility() . . . . .	146
4.58.4.4 isPassable() . . . . .	146
4.58.4.5 isSeen() . . . . .	146
4.58.4.6 setIsSeen() . . . . .	146
4.58.4.7 setPassable() . . . . .	147
4.58.4.8 setSymbol() . . . . .	147
4.58.5 Member Data Documentation . . . . .	147
4.58.5.1 character . . . . .	147
4.58.5.2 checked . . . . .	147
4.58.5.3 parent . . . . .	147
4.58.5.4 passable . . . . .	148

4.58.5.5	visible	148
4.59	TerrainTest Class Reference	148
4.60	Testable Class Reference	150
4.61	Trap Class Reference	151
4.61.1	Detailed Description	152
4.61.2	Constructor & Destructor Documentation	152
4.61.2.1	Trap()	152
4.61.3	Member Function Documentation	152
4.61.3.1	activate()	152
4.61.3.2	randomTrap()	153
4.61.4	Member Data Documentation	153
4.61.4.1	MAX_TYPE	153
4.62	TrapTest Class Reference	153
4.63	Tunnel Class Reference	154
4.63.1	Detailed Description	154
4.63.2	Constructor & Destructor Documentation	154
4.63.2.1	Tunnel()	154
4.63.3	Member Function Documentation	155
4.63.3.1	dig()	155
4.64	TunnelTest Class Reference	155
4.65	UIState Class Reference	156
4.65.1	Detailed Description	157
4.66	UIStateTest Class Reference	157
4.67	Wall Class Reference	158
4.67.1	Detailed Description	159
4.68	Wand Class Reference	159
4.68.1	Detailed Description	160
4.68.2	Constructor & Destructor Documentation	160
4.68.2.1	Wand() [1/2]	160
4.68.2.2	Wand() [2/2]	161

4.68.3 Member Function Documentation . . . . .	161
4.68.3.1 activate() . . . . .	161
4.68.3.2 getCharges() . . . . .	161
4.69 WandTest Class Reference . . . . .	162
4.70 Weapon Class Reference . . . . .	162
4.70.1 Detailed Description . . . . .	164
4.70.2 Constructor & Destructor Documentation . . . . .	164
4.70.2.1 Weapon() [1/2] . . . . .	164
4.70.2.2 Weapon() [2/2] . . . . .	164
4.70.3 Member Function Documentation . . . . .	165
4.70.3.1 getChance() . . . . .	165
4.70.3.2 getDamage() . . . . .	165
4.70.3.3 getEnchantments() . . . . .	165
4.70.3.4 isMelee() . . . . .	165
4.70.3.5 setEnchantments() . . . . .	165
4.71 WeaponTest Class Reference . . . . .	166
<b>5 File Documentation</b> . . . . .	<b>167</b>
5.1 amulet.cpp File Reference . . . . .	167
5.1.1 Detailed Description . . . . .	167
5.2 armor.cpp File Reference . . . . .	167
5.2.1 Detailed Description . . . . .	168
5.3 coord.cpp File Reference . . . . .	168
5.3.1 Detailed Description . . . . .	169
5.4 feature.cpp File Reference . . . . .	169
5.4.1 Detailed Description . . . . .	170
5.5 food.cpp File Reference . . . . .	170
5.5.1 Detailed Description . . . . .	171
5.6 goldpile.cpp File Reference . . . . .	171
5.6.1 Detailed Description . . . . .	172
5.7 helpscreen.cpp File Reference . . . . .	172

5.7.1	Detailed Description	173
5.8	include/amulet.h File Reference	173
5.8.1	Detailed Description	175
5.9	include/armor.h File Reference	175
5.9.1	Detailed Description	176
5.10	include/controls.h File Reference	176
5.10.1	Detailed Description	177
5.11	include/coord.h File Reference	178
5.11.1	Detailed Description	178
5.12	include/debug.h File Reference	179
5.12.1	Detailed Description	179
5.13	include/feature.h File Reference	179
5.13.1	Detailed Description	180
5.14	include/food.h File Reference	180
5.14.1	Detailed Description	181
5.15	include/globals.h File Reference	181
5.15.1	Detailed Description	183
5.16	include/goldpile.h File Reference	183
5.16.1	Detailed Description	184
5.17	include/helpscreen.h File Reference	184
5.17.1	Detailed Description	185
5.18	include/invscreen.h File Reference	185
5.18.1	Detailed Description	186
5.19	include/item.h File Reference	187
5.19.1	Detailed Description	187
5.20	include/itemzone.h File Reference	188
5.20.1	Detailed Description	188
5.21	include/level.h File Reference	189
5.21.1	Detailed Description	190
5.22	include/logscren.h File Reference	190

---

5.22.1 Detailed Description . . . . .	191
5.23 include/mainmenu.h File Reference . . . . .	191
5.23.1 Detailed Description . . . . .	192
5.24 include/mastercontroller.h File Reference . . . . .	193
5.24.1 Detailed Description . . . . .	194
5.25 include/mob.h File Reference . . . . .	194
5.25.1 Detailed Description . . . . .	195
5.26 include/monster.h File Reference . . . . .	195
5.26.1 Detailed Description . . . . .	196
5.27 include/playerchar.h File Reference . . . . .	196
5.27.1 Detailed Description . . . . .	197
5.28 include/playstate.h File Reference . . . . .	197
5.28.1 Detailed Description . . . . .	198
5.29 include/potion.h File Reference . . . . .	198
5.29.1 Detailed Description . . . . .	199
5.30 include/random.h File Reference . . . . .	199
5.30.1 Detailed Description . . . . .	200
5.31 include/ring.h File Reference . . . . .	201
5.31.1 Detailed Description . . . . .	202
5.32 include/ripscreen.h File Reference . . . . .	202
5.32.1 Detailed Description . . . . .	203
5.33 include/room.h File Reference . . . . .	203
5.33.1 Detailed Description . . . . .	204
5.34 include/savescreen.h File Reference . . . . .	205
5.34.1 Detailed Description . . . . .	206
5.35 include/saving.h File Reference . . . . .	206
5.35.1 Detailed Description . . . . .	207
5.36 include/scroll.h File Reference . . . . .	207
5.36.1 Detailed Description . . . . .	208
5.37 include/stairs.h File Reference . . . . .	208

---

5.37.1 Detailed Description . . . . .	209
5.38 include/statusscreen.h File Reference . . . . .	210
5.38.1 Detailed Description . . . . .	211
5.39 include/symbolscreen.h File Reference . . . . .	211
5.39.1 Detailed Description . . . . .	212
5.40 include/terrain.h File Reference . . . . .	212
5.40.1 Detailed Description . . . . .	213
5.41 include/tiles.h File Reference . . . . .	214
5.41.1 Detailed Description . . . . .	215
5.42 include/trap.h File Reference . . . . .	215
5.42.1 Detailed Description . . . . .	216
5.43 include/tunnel.h File Reference . . . . .	216
5.43.1 Detailed Description . . . . .	217
5.44 include/uistate.h File Reference . . . . .	217
5.44.1 Detailed Description . . . . .	218
5.45 include/wand.h File Reference . . . . .	218
5.45.1 Detailed Description . . . . .	219
5.46 include/weapon.h File Reference . . . . .	219
5.46.1 Detailed Description . . . . .	220
5.47 include/wizard.h File Reference . . . . .	221
5.47.1 Detailed Description . . . . .	221
5.48 invscreen.cpp File Reference . . . . .	221
5.48.1 Detailed Description . . . . .	222
5.49 item.cpp File Reference . . . . .	222
5.49.1 Detailed Description . . . . .	223
5.50 level.cpp File Reference . . . . .	223
5.50.1 Detailed Description . . . . .	224
5.51 logscreen.cpp File Reference . . . . .	224
5.51.1 Detailed Description . . . . .	225
5.52 main.cpp File Reference . . . . .	225

---

5.52.1 Detailed Description . . . . .	226
5.53 mainmenu.cpp File Reference . . . . .	226
5.53.1 Detailed Description . . . . .	227
5.54 mastercontroller.cpp File Reference . . . . .	227
5.54.1 Detailed Description . . . . .	227
5.55 misc/codeformatter.py File Reference . . . . .	228
5.55.1 Detailed Description . . . . .	228
5.55.2 Function Documentation . . . . .	228
5.55.2.1 addHeader() . . . . .	228
5.55.2.2 cleanPragmas() . . . . .	229
5.55.2.3 formatContent() . . . . .	229
5.55.2.4 formatFiles() . . . . .	229
5.55.2.5 sortIncludes() . . . . .	230
5.55.2.6 trim() . . . . .	230
5.56 misc/stringfinder.py File Reference . . . . .	230
5.56.1 Detailed Description . . . . .	231
5.57 mob.cpp File Reference . . . . .	231
5.57.1 Detailed Description . . . . .	232
5.58 monster.cpp File Reference . . . . .	232
5.58.1 Detailed Description . . . . .	232
5.59 playerchar.cpp File Reference . . . . .	233
5.59.1 Detailed Description . . . . .	233
5.60 playstate.cpp File Reference . . . . .	233
5.60.1 Detailed Description . . . . .	234
5.61 potion.cpp File Reference . . . . .	235
5.61.1 Detailed Description . . . . .	235
5.62 random.cpp File Reference . . . . .	236
5.62.1 Detailed Description . . . . .	236
5.63 ring.cpp File Reference . . . . .	236
5.63.1 Detailed Description . . . . .	237

5.64 ripscreen.cpp File Reference . . . . .	237
5.64.1 Detailed Description . . . . .	238
5.65 room.cpp File Reference . . . . .	238
5.65.1 Detailed Description . . . . .	239
5.66 savescreen.cpp File Reference . . . . .	239
5.66.1 Detailed Description . . . . .	239
5.67 saving.cpp File Reference . . . . .	240
5.67.1 Detailed Description . . . . .	240
5.68 scroll.cpp File Reference . . . . .	241
5.68.1 Detailed Description . . . . .	241
5.69 stairs.cpp File Reference . . . . .	241
5.69.1 Detailed Description . . . . .	242
5.70 statusscreen.cpp File Reference . . . . .	242
5.70.1 Detailed Description . . . . .	243
5.71 symbolscreen.cpp File Reference . . . . .	243
5.71.1 Detailed Description . . . . .	244
5.72 terrain.cpp File Reference . . . . .	244
5.72.1 Detailed Description . . . . .	244
5.73 test.amulet.cpp File Reference . . . . .	245
5.73.1 Detailed Description . . . . .	246
5.74 test.armor.cpp File Reference . . . . .	246
5.74.1 Detailed Description . . . . .	247
5.75 test.coord.cpp File Reference . . . . .	247
5.75.1 Detailed Description . . . . .	248
5.76 test.feature.cpp File Reference . . . . .	248
5.76.1 Detailed Description . . . . .	249
5.77 test.food.cpp File Reference . . . . .	250
5.77.1 Detailed Description . . . . .	251
5.78 test.goldpile.cpp File Reference . . . . .	251
5.78.1 Detailed Description . . . . .	252

5.79 test.item.cpp File Reference . . . . .	252
5.79.1 Detailed Description . . . . .	253
5.80 test.itemzone.cpp File Reference . . . . .	254
5.80.1 Detailed Description . . . . .	254
5.81 test.level.cpp File Reference . . . . .	255
5.81.1 Detailed Description . . . . .	256
5.82 test.levelgen.cpp File Reference . . . . .	256
5.82.1 Detailed Description . . . . .	257
5.83 test.main.cpp File Reference . . . . .	257
5.83.1 Detailed Description . . . . .	258
5.84 test.mob.cpp File Reference . . . . .	258
5.84.1 Detailed Description . . . . .	259
5.85 test.monster.cpp File Reference . . . . .	259
5.85.1 Detailed Description . . . . .	260
5.86 test.playerchar.cpp File Reference . . . . .	260
5.86.1 Detailed Description . . . . .	261
5.87 test.potion.cpp File Reference . . . . .	261
5.87.1 Detailed Description . . . . .	262
5.88 test.random.cpp File Reference . . . . .	262
5.88.1 Detailed Description . . . . .	263
5.89 test.ring.cpp File Reference . . . . .	263
5.89.1 Detailed Description . . . . .	264
5.90 test.room.cpp File Reference . . . . .	265
5.90.1 Detailed Description . . . . .	265
5.91 test.scroll.cpp File Reference . . . . .	266
5.91.1 Detailed Description . . . . .	267
5.92 test.stairs.cpp File Reference . . . . .	267
5.92.1 Detailed Description . . . . .	268
5.93 test.terrain.cpp File Reference . . . . .	268
5.93.1 Detailed Description . . . . .	269

---

5.94 test.testable.cpp File Reference . . . . .	270
5.94.1 Detailed Description . . . . .	270
5.95 test.testable.h File Reference . . . . .	271
5.95.1 Detailed Description . . . . .	271
5.96 test.trap.cpp File Reference . . . . .	272
5.96.1 Detailed Description . . . . .	272
5.97 test.tunnel.cpp File Reference . . . . .	273
5.97.1 Detailed Description . . . . .	274
5.98 test.uistate.cpp File Reference . . . . .	274
5.98.1 Detailed Description . . . . .	275
5.99 test.wand.cpp File Reference . . . . .	275
5.99.1 Detailed Description . . . . .	276
5.100 test.weapon.cpp File Reference . . . . .	276
5.100.1 Detailed Description . . . . .	277
5.101 tiles.cpp File Reference . . . . .	277
5.101.1 Detailed Description . . . . .	278
5.102 trap.cpp File Reference . . . . .	278
5.102.1 Detailed Description . . . . .	279
5.103 tunnel.cpp File Reference . . . . .	279
5.103.1 Detailed Description . . . . .	280
5.104 uistate.cpp File Reference . . . . .	280
5.104.1 Detailed Description . . . . .	280
5.105 wand.cpp File Reference . . . . .	281
5.105.1 Detailed Description . . . . .	281
5.106 weapon.cpp File Reference . . . . .	281
5.106.1 Detailed Description . . . . .	282
<b>Index</b>	<b>283</b>



# Chapter 1

## Hierarchical Index

### 1.1 Class Hierarchy

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

Coord . . . . .	19
Feature . . . . .	28
GoldPile . . . . .	38
Item . . . . .	45
Amulet . . . . .	13
Armor . . . . .	16
Food . . . . .	34
Potion . . . . .	111
Ring . . . . .	121
Scroll . . . . .	134
Wand . . . . .	159
Weapon . . . . .	162
Stairs . . . . .	138
Trap . . . . .	151
Generator . . . . .	36
ItemZone . . . . .	54
Level . . . . .	57
MasterController . . . . .	69
Mob . . . . .	70
Monster . . . . .	79
PlayerChar . . . . .	90
Room . . . . .	128
ScoreItem . . . . .	133
Terrain . . . . .	143
Corridor . . . . .	24
Door . . . . .	27
Floor . . . . .	33
Wall . . . . .	158
Testable . . . . .	150
AmuletTest . . . . .	15
ArmorTest . . . . .	19
CoordTest . . . . .	24
FeatureTest . . . . .	32
FoodTest . . . . .	36

GoldPileTest . . . . .	40
ItemTest . . . . .	53
ItemZoneTest . . . . .	56
LevelGenTest . . . . .	64
LevelTest . . . . .	65
MobTest . . . . .	78
MonsterTest . . . . .	89
PlayerCharTest . . . . .	107
PotionTest . . . . .	113
RandomTest . . . . .	120
RingTest . . . . .	126
RoomTest . . . . .	131
ScrollTest . . . . .	137
StairsTest . . . . .	139
TerrainTest . . . . .	148
TrapTest . . . . .	153
TunnelTest . . . . .	155
UIStateTest . . . . .	157
WandTest . . . . .	162
WeaponTest . . . . .	166
Tunnel . . . . .	154
UIState . . . . .	156
HelpScreen . . . . .	40
InvScreen . . . . .	42
LogScreen . . . . .	66
MainMenu . . . . .	68
PlayState . . . . .	108
DirectionPrompt . . . . .	26
QuickThrow . . . . .	114
QuickUse< T > . . . . .	116
QuickZap . . . . .	117
QuitPrompt2 . . . . .	119
RingRemovePrompt . . . . .	124
RIPScreen . . . . .	126
SaveScreen . . . . .	132
StatusScreen . . . . .	139
SymbolScreen . . . . .	141

# Chapter 2

## Class Index

### 2.1 Class List

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

<a href="#">Amulet</a>	Represents the <a href="#">Amulet</a> of Yendor . . . . .	13
<a href="#">AmuletTest</a>	. . . . .	15
<a href="#">Armor</a>	Represents armor . . . . .	16
<a href="#">ArmorTest</a>	. . . . .	19
<a href="#">Coord</a>	Represents a location within the dungeon or on the screen . . . . .	19
<a href="#">CoordTest</a>	. . . . .	24
<a href="#">Corridor</a>	Regular corridor tile . . . . .	24
<a href="#">DirectionPrompt</a>	. . . . .	26
<a href="#">Door</a>	Door tile . . . . .	27
<a href="#">Feature</a>	Models a 'thing' in the dungeon that has position and may be visible . . . . .	28
<a href="#">FeatureTest</a>	. . . . .	32
<a href="#">Floor</a>	Regular dungeon floor . . . . .	33
<a href="#">Food</a>	Represents food . . . . .	34
<a href="#">FoodTest</a>	. . . . .	36
<a href="#">Generator</a>	Light wrapper around the std library which provides various random generation utilities . . . . .	36
<a href="#">GoldPile</a>	Represents a pile of gold on the ground, which can be picked up by the player to enhance their score . . . . .	38
<a href="#">GoldPileTest</a>	. . . . .	40
<a href="#">HelpScreen</a>	Interface state that shows the various game controls . . . . .	40
<a href="#">InvScreen</a>	Interface state for viewing the contents of the player inventory . . . . .	42
<a href="#">Item</a>	Represents a generic item . . . . .	45
<a href="#">ItemTest</a>	. . . . .	53

<b>ItemZone</b>	Container for items . . . . .	54
<b>ItemZoneTest</b>	. . . . .	56
<b>Level</b>	. . . . .	57
<b>LevelGenTest</b>	. . . . .	64
<b>LevelTest</b>	. . . . .	65
<b>LogScreen</b>	Controls the display of the event log . . . . .	66
<b>MainMenu</b>	Start screen of the game . . . . .	68
<b>MasterController</b>	Controls the top level flow flow of the application and main game loop . . . . .	69
<b>Mob</b>	Models a creature in the dungeon, could be the player or a monster . . . . .	70
<b>MobTest</b>	. . . . .	78
<b>Monster</b>	Models a monster in the dungeon . . . . .	79
<b>MonsterTest</b>	. . . . .	89
<b>PlayerChar</b>	Models the user-controlled player character . . . . .	90
<b>PlayerCharTest</b>	. . . . .	107
<b>PlayState</b>	Primary interface state, showing level, player, monsters, etc . . . . .	108
<b>Potion</b>	Represents potions . . . . .	111
<b>PotionTest</b>	. . . . .	113
<b>QuickThrow</b>	. . . . .	114
<b>QuickUse&lt; T &gt;</b>	. . . . .	116
<b>QuickZap</b>	. . . . .	117
<b>QuitPrompt2</b>	. . . . .	119
<b>RandomTest</b>	. . . . .	120
<b>Ring</b>	Represents rings . . . . .	121
<b>RingRemovePrompt</b>	. . . . .	124
<b>RingTest</b>	. . . . .	126
<b>RIPScreen</b>	Interface state for post-death/retirement, looking at the high-score table . . . . .	126
<b>Room</b>	Models a room - a rectangular region of which there are (usually) 9 in any given dungeon level	128
<b>RoomTest</b>	. . . . .	131
<b>SaveScreen</b>	. . . . .	132
<b>ScoreItem</b>	. . . . .	133
<b>Scroll</b>	Represents scrolls . . . . .	134
<b>ScrollTest</b>	. . . . .	137
<b>Stairs</b>	. . . . .	138
<b>StairsTest</b>	. . . . .	139
<b>StatusScreen</b>	Screen which shows detailed player statistics . . . . .	139
<b>SymbolScreen</b>	. . . . .	141
<b>Terrain</b>	Represents a tile in the dungeon . . . . .	143
<b>TerrainTest</b>	. . . . .	148
<b>Testable</b>	. . . . .	150
<b>Trap</b>	Various hidden traps throughout the dungeon can trigger and endanger the player . . . . .	151
<b>TrapTest</b>	. . . . .	153

<a href="#">Tunnel</a>	Tunnels are step-orthogonal paths connecting rooms . . . . .	154
<a href="#">TunnelTest</a>	. . . . .	155
<a href="#">UIState</a>	Class modeling a state of the game interface . . . . .	156
<a href="#">UIStateTest</a>	. . . . .	157
<a href="#">Wall</a>	Regular dungeon wall . . . . .	158
<a href="#">Wand</a>	Represents a wand item . . . . .	159
<a href="#">WandTest</a>	. . . . .	162
<a href="#">Weapon</a>	Represents weapons . . . . .	162
<a href="#">WeaponTest</a>	. . . . .	166



# Chapter 3

## File Index

### 3.1 File List

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

amulet.cpp	Member definitions for the <a href="#">Amulet</a> class . . . . .	167
armor.cpp	Member definitions for the <a href="#">Armor</a> class . . . . .	167
coord.cpp	Member definitions for the <a href="#">Coord</a> class . . . . .	168
feature.cpp	Member definitions for the <a href="#">Feature</a> class . . . . .	169
food.cpp	Member definitions for the <a href="#">Food</a> class . . . . .	170
goldpile.cpp	Member definitions for the <a href="#">GoldPile</a> class . . . . .	171
helpscreen.cpp	Member definitions for the <a href="#">HelpScreen</a> class . . . . .	172
invscreen.cpp	Member definitions for the <a href="#">InvScreen</a> class . . . . .	221
item.cpp	Member definitions for the <a href="#">Item</a> class . . . . .	222
level.cpp	Member definitions for the <a href="#">Level</a> class . . . . .	223
logscreen.cpp	Member definitions for the <a href="#">LogScreen</a> class . . . . .	224
main.cpp	Global members . . . . .	225
mainmenu.cpp	Member definitions for the <a href="#">MainMenu</a> class . . . . .	226
mastercontroller.cpp	Member definitions for the <a href="#">MasterController</a> class . . . . .	227
mob.cpp	Member definitions for the <a href="#">Mob</a> class . . . . .	231
monster.cpp	Member definitions for the <a href="#">Monster</a> class . . . . .	232
playerchar.cpp	Member definitions for the <a href="#">PlayerChar</a> class . . . . .	233
playstate.cpp	Member definitions for the <a href="#">PlayState</a> class . . . . .	233

<a href="#">potion.cpp</a>	
Member definitions for the <a href="#">Potion</a> class . . . . .	235
<a href="#">random.cpp</a>	
Global members . . . . .	236
<a href="#">ring.cpp</a>	
Member definitions for the <a href="#">Ring</a> class . . . . .	236
<a href="#">ripscreen.cpp</a>	
Member definitions for the <a href="#">RIPScreen</a> class . . . . .	237
<a href="#">room.cpp</a>	
Member definitions for the <a href="#">Room</a> class . . . . .	238
<a href="#">savescreen.cpp</a>	
Member definitions for the <a href="#">SaveScreen</a> class . . . . .	239
<a href="#">saving.cpp</a>	
Global members . . . . .	240
<a href="#">scroll.cpp</a>	
Member definitions for the <a href="#">Scroll</a> class . . . . .	241
<a href="#">stairs.cpp</a>	
Member definitions for the <a href="#">Stairs</a> class . . . . .	241
<a href="#">statusscreen.cpp</a>	
Member definitions for the <a href="#">StatusScreen</a> class . . . . .	242
<a href="#">symbolscreen.cpp</a>	
Member definitions for the <a href="#">SymbolScreen</a> class . . . . .	243
<a href="#">terrain.cpp</a>	
Member definitions for the <a href="#">Terrain</a> class . . . . .	244
<a href="#">test.amulet.cpp</a>	
Global members . . . . .	245
<a href="#">test.armor.cpp</a>	
Global members . . . . .	246
<a href="#">test.coord.cpp</a>	
Global members . . . . .	247
<a href="#">test.feature.cpp</a>	
Global members . . . . .	248
<a href="#">test.food.cpp</a>	
Global members . . . . .	250
<a href="#">test.goldpile.cpp</a>	
Global members . . . . .	251
<a href="#">test.item.cpp</a>	
Global members . . . . .	252
<a href="#">test.itemzone.cpp</a>	
Global members . . . . .	254
<a href="#">test.level.cpp</a>	
Global members . . . . .	255
<a href="#">test.levelgen.cpp</a>	
Global members . . . . .	256
<a href="#">test.main.cpp</a>	
Global members . . . . .	257
<a href="#">test.mob.cpp</a>	
Global members . . . . .	258
<a href="#">test.monster.cpp</a>	
Global members . . . . .	259
<a href="#">test.playerchar.cpp</a>	
Global members . . . . .	260
<a href="#">test.potion.cpp</a>	
Global members . . . . .	261
<a href="#">test.random.cpp</a>	
Global members . . . . .	262
<a href="#">test.ring.cpp</a>	
Global members . . . . .	263

test.room.cpp	Global members . . . . .	265
test.scroll.cpp	Global members . . . . .	266
test.stairs.cpp	Global members . . . . .	267
test.terrain.cpp	Global members . . . . .	268
test.testable.cpp	Global members . . . . .	270
test.testable.h	Member declarations for the <a href="#">Testable</a> class . . . . .	271
test.trap.cpp	Global members . . . . .	272
test.tunnel.cpp	Global members . . . . .	273
test.uistate.cpp	Global members . . . . .	274
test.wand.cpp	Global members . . . . .	275
test.weapon.cpp	Global members . . . . .	276
tiles.cpp	Member definitions for the <a href="#">Corridor</a> , <a href="#">Door</a> , <a href="#">Floor</a> , <a href="#">Wall</a> classes . . . . .	277
trap.cpp	Member definitions for the <a href="#">Trap</a> class . . . . .	278
tunnel.cpp	Member definitions for the <a href="#">Tunnel</a> class . . . . .	279
uistate.cpp	Member definitions for the <a href="#">UIState</a> class . . . . .	280
wand.cpp	Member definitions for the <a href="#">Wand</a> class . . . . .	281
weapon.cpp	Member definitions for the <a href="#">Weapon</a> class . . . . .	281
include/amulet.h	Member declarations for the <a href="#">Amulet</a> class . . . . .	173
include/armor.h	Member declarations for the <a href="#">Armor</a> class . . . . .	175
include/controls.h	Global members . . . . .	176
include/coord.h	Member declarations for the <a href="#">Coord</a> class . . . . .	178
include/debug.h	Global members . . . . .	179
include/feature.h	Member declarations for the <a href="#">Feature</a> class . . . . .	179
include/food.h	Member declarations for the <a href="#">Food</a> class . . . . .	180
include/globals.h	Global members . . . . .	181
include/goldpile.h	Member declarations for the <a href="#">GoldPile</a> class . . . . .	183
include/helpscreen.h	Member declarations for the <a href="#">HelpScreen</a> class . . . . .	184
include/invscreen.h	Member declarations for the <a href="#">InvScreen</a> class . . . . .	185
include/item.h	Member declarations for the <a href="#">Item</a> class . . . . .	187

include/itemzone.h	
Member declarations for the <code>ItemZone</code> class . . . . .	188
include/level.h	
Member declarations for the <code>Level</code> class . . . . .	189
include/logscren.h	
Member declarations for the <code>LogScreen</code> class . . . . .	190
include/mainmenu.h	
Member declarations for the <code>MainMenu</code> class . . . . .	191
include/mastercontroller.h	
Member declarations for the <code>MasterController</code> class . . . . .	193
include/mob.h	
Member declarations for the <code>Mob</code> class . . . . .	194
include/monster.h	
Member declarations for the <code>Monster</code> class . . . . .	195
include/playerchar.h	
Member declarations for the <code>PlayerChar</code> class . . . . .	196
include/playstate.h	
Member declarations for the <code>PlayState</code> class . . . . .	197
include/potion.h	
Member declarations for the <code>Potion</code> class . . . . .	198
include/random.h	
Member declarations for the <code>Generator</code> class . . . . .	199
include/ring.h	
Member declarations for the <code>Ring</code> class . . . . .	201
include/ripscreen.h	
Member declarations for the <code>RIPScreen</code> class . . . . .	202
include/room.h	
Member declarations for the <code>Room</code> class . . . . .	203
include/savescreen.h	
Member declarations for the <code>SaveScreen</code> class . . . . .	205
include/saving.h	
Global members . . . . .	206
include/scroll.h	
Member declarations for the <code>Scroll</code> class . . . . .	207
include/stairs.h	
Member declarations for the <code>Stairs</code> class . . . . .	208
include/statusscreen.h	
Member declarations for the <code>StatusScreen</code> class . . . . .	210
include/symbolscreen.h	
Member declarations for the <code>SymbolScreen</code> class . . . . .	211
include/terrain.h	
Member declarations for the <code>Terrain</code> class . . . . .	212
include/tiles.h	
Member declarations for the <code>Corridor, Door, Floor, Wall</code> classes . . . . .	214
include/trap.h	
Member declarations for the <code>Trap</code> class . . . . .	215
include/tunnel.h	
Member declarations for the <code>Tunnel</code> class . . . . .	216
include/uistate.h	
Member declarations for the <code>UIState</code> class . . . . .	217
include/wand.h	
Member declarations for the <code>Wand</code> class . . . . .	218
include/weapon.h	
Member declarations for the <code>Weapon</code> class . . . . .	219
include/wizard.h	
Global members . . . . .	221
misc/codeformatter.py	
Performs several formatting operations over the C++ header and source files . . . . .	228

misc/ <a href="#">stringfinder.py</a>	Accumulates all strings throughout the C++ files . . . . .	230
---------------------------------------	--	-----



## Chapter 4

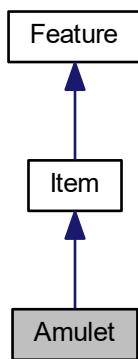
# Class Documentation

### 4.1 Amulet Class Reference

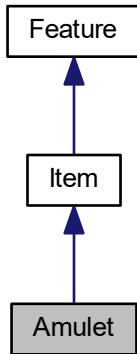
Represents the [Amulet](#) of Yendor.

```
#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

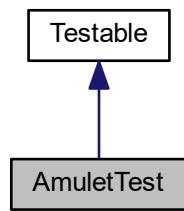
in	<i>location</i>	<a href="#">Amulet</a> location
in	<i>context</i>	<a href="#">Amulet</a> context

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

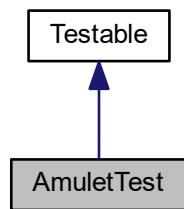
- [include/amulet.h](#)
- [amulet.cpp](#)

## 4.2 AmuletTest Class Reference

Inheritance diagram for AmuletTest:



Collaboration diagram for AmuletTest:



### Public Member Functions

- [void test \(\)](#)

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

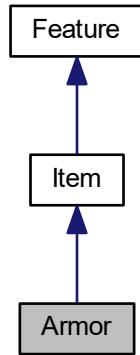
- [test.amulet.cpp](#)

## 4.3 Armor Class Reference

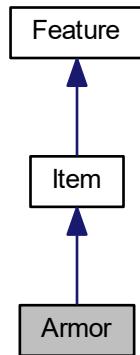
Represents armor.

```
#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 `getEnchantment ()`  
Gets the enchantment.
- int `getRating ()`  
Gets the rating.
- void `setEnchantment (int)`  
Sets this `Armor`'s enchantment.

## Friends

- std::string `encode (PlayerChar *, Level *)`
- std::tuple< `PlayerChar *`, `Level *` > `decode (std::string)`

## Additional Inherited Members

### 4.3.1 Detailed Description

Represents armor.

### 4.3.2 Constructor & Destructor Documentation

#### 4.3.2.1 `Armor()` [1/2]

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

Constructs an `Armor` instance with a random type.

##### Parameters

in	<i>location</i>	<code>Armor</code> location
----	-----------------	-----------------------------

#### 4.3.2.2 `Armor()` [2/2]

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

Constructs an `Armor` instance.

##### Parameters

in	<i>location</i>	<code>Armor</code> location
in	<i>context</i>	<code>Armor</code> context
in	<i>type</i>	<code>Armor</code> type

### 4.3.3 Member Function Documentation

#### 4.3.3.1 getEnchantment()

```
int Armor::getEnchantment ( )
```

Gets the enchantment.

##### Returns

The enchantment.

#### 4.3.3.2 getRating()

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

Gets the rating.

##### Returns

The rating.

#### 4.3.3.3 setEnchantment()

```
void Armor::setEnchantment ( int enchantProtection )
```

Sets this [Armor](#)'s enchantment.

##### Parameters

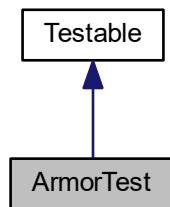
in	<i>enchantment</i>	New enchantment for this <a href="#">Armor</a> .
----	--------------------	--

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

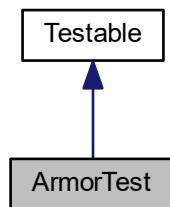
- [include/armor.h](#)
- [armor.cpp](#)

## 4.4 ArmorTest Class Reference

Inheritance diagram for ArmorTest:



Collaboration diagram for ArmorTest:



### Public Member Functions

- void **test ()**

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

- [test.armor.cpp](#)

## 4.5 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 &, bool taxicab=true) const**  
*Maximum distance in either dimension.*

## Static Public Attributes

- static `Coord ORTHO [4]`

*Set of unit vectors.*

### 4.5.1 Detailed Description

Represents a location within the dungeon or on the screen.

### 4.5.2 Constructor & Destructor Documentation

#### 4.5.2.1 `Coord() [1/2]`

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

(x,y) constructor.

#### 4.5.2.2 `Coord() [2/2]`

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

(0,0) constructor.

### 4.5.3 Member Function Documentation

#### 4.5.3.1 `asScreen()`

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

Convert position in level to position in screen.

#### 4.5.3.2 `copy()`

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

Return a copy of this coord.

#### 4.5.3.3 `isAdjacentTo()`

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

Return distance(taxicab)  $\leq 1$ .

#### 4.5.3.4 operator"!=()

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

Inverse of == operator.

#### 4.5.3.5 operator\*()

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

Multiply all vector items by scalar.

#### 4.5.3.6 operator\*=(())

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

Multiply two coords (item by item).

#### 4.5.3.7 operator+()

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

Add two coords together.

#### 4.5.3.8 operator+=(())

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

Augmented assignment for addition.

#### 4.5.3.9 operator-()

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

Subtract two coords.

#### 4.5.3.10 operator-=(())

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

Augmented assignment for subtraction.

#### 4.5.3.11 operator<()

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

Order coords by overall magnitude.

#### 4.5.3.12 operator==( )

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

True if all vector items equal.

#### 4.5.3.13 operator[]( )

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

Access param dimension magnitude.

#### 4.5.3.14 toString()

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

Format as x, y.

### 4.5.4 Member Data Documentation

#### 4.5.4.1 ORTHO

`Coord Coord::ORTHO [static]`

**Initial value:**

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

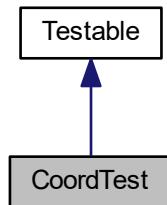
Set of unit vectors.

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

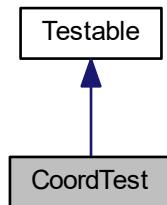
- include/coord.h
- coord.cpp

## 4.6 CoordTest Class Reference

Inheritance diagram for CoordTest:



Collaboration diagram for CoordTest:



### Public Member Functions

- void **test ()**

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

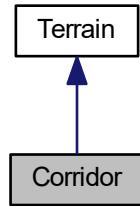
- [test.coord.cpp](#)

## 4.7 Corridor Class Reference

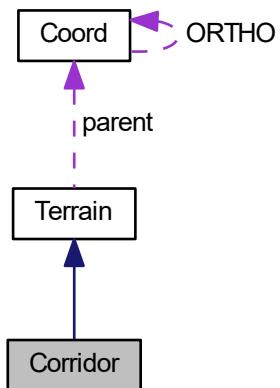
Regular corridor tile.

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

Inheritance diagram for Corridor:



Collaboration diagram for Corridor:



## Additional Inherited Members

### 4.7.1 Detailed Description

Regular corridor tile.

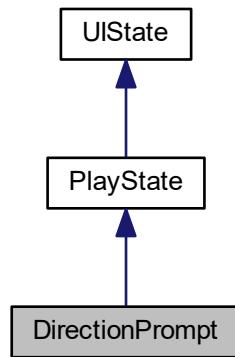
Has limited visibility and full passability

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

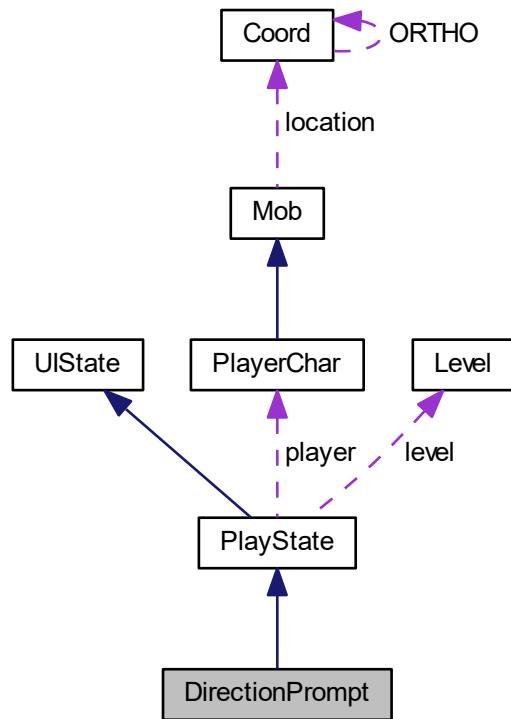
- [include/tiles.h](#)
- [tiles.cpp](#)

## 4.8 DirectionPrompt Class Reference

Inheritance diagram for DirectionPrompt:



Collaboration diagram for DirectionPrompt:



## Public Member Functions

- **DirectionPrompt** (`PlayerChar *player, Level *level, std::function< UIState *(Coord)> makeUseOf)`
- 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

### 4.8.1 Member Function Documentation

#### 4.8.1.1 `handleInput()`

```
virtual UIState* DirectionPrompt::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:

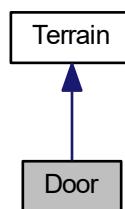
- [playstate.cpp](#)

## 4.9 Door Class Reference

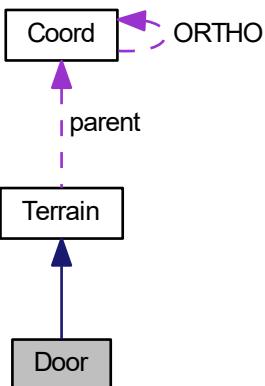
[Door](#) tile.

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

Inheritance diagram for Door:



Collaboration diagram for Door:



## Additional Inherited Members

### 4.9.1 Detailed Description

[Door](#) tile.

Only cosmetically different from corridor tile.

See also

[Corridor](#)

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

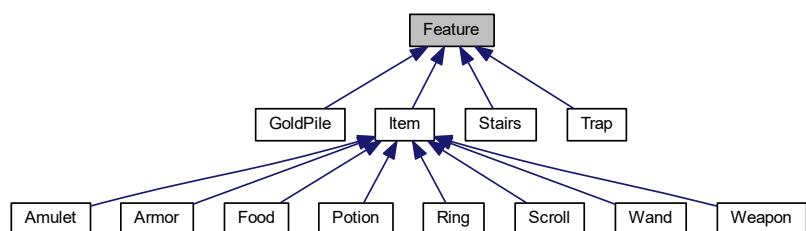
- [include/tiles.h](#)
- [tiles.cpp](#)

## 4.10 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 *symbol*, **Coord** *location*, bool *visible*=true, TCODColor *fcolor*=TCODColor::white)  
*Constructor for symbol, location.*
- virtual char **getSymbol** ()  
*Getter for symbol.*
- **Coord getLocation** ()  
*Getter for location.*
- bool **getVisible** ()  
*Setter for location.*
- void **setVisible** (bool)  
*Setter for visibility.*
- void **setLocation** (**Coord**)  
*Setter for location.*
- TCODColor **getFColor** ()  
*Getter for the foreground color.*
- virtual ~**Feature** ()  
*Destructor.*

## Protected Attributes

- bool **visible**  
*Whether the feature is visible to the player.*
- TCODColor **fcolor**  
*Foreground color the feature.*

## Static Protected Attributes

- static const std::vector< TCODColor > **possibleColors**  
*Set of possible colors for randomly colored features.*

### 4.10.1 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

### 4.10.2 Constructor & Destructor Documentation

#### 4.10.2.1 Feature()

```
Feature::Feature (
    char symbol,
    Coord location,
    bool visible = true,
    TCODColor fcolor = TCODColor::white )
```

Constructor for symbol, location.

#### 4.10.2.2 ~Feature()

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

Destructor.

### 4.10.3 Member Function Documentation

#### 4.10.3.1 getFColor()

```
TCODColor Feature::getFColor ( )
```

Getter for the foreground color.

#### See also

[fcolor](#)

#### Returns

Foreground color of the feature.

#### 4.10.3.2 getLocation()

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

Getter for location.

#### See also

[location](#)

#### 4.10.3.3 getSymbol()

```
char Feature::getSymbol ( ) [virtual]
```

Getter for symbol.

#### See also

[symbol](#)

#### 4.10.3.4 getVisible()

```
bool Feature::getVisible ( )
```

Setter for location.

##### See also

[location](#) Getter for visibility.  
[visible](#)

#### 4.10.3.5 setLocation()

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

Setter for location.

##### See also

[location](#)

#### 4.10.3.6 setVisible()

```
void Feature::setVisible (  
    bool newvis )
```

Setter for visibility.

##### See also

[visible](#)

### 4.10.4 Member Data Documentation

#### 4.10.4.1 fcolor

```
TCODColor Feature::fcolor [protected]
```

Foreground color the feature.

#### 4.10.4.2 possibleColors

```
const std::vector< TCODColor > Feature::possibleColors [static], [protected]
```

##### Initial value:

```
= {TCODColor::lightBlue, TCODColor::red, TCODColor::orange,  
   TCODColor::green, TCODColor::purple, TCODColor::yellow}
```

Set of possible colors for randomly colored features.

---

#### 4.10.4.3 visible

```
bool Feature::visible [protected]
```

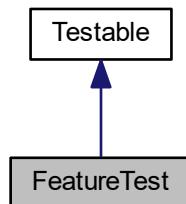
Whether the feature is visible to the player.

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

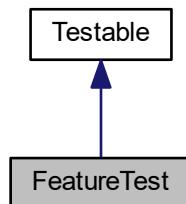
- [include/feature.h](#)
- [feature.cpp](#)

## 4.11 FeatureTest Class Reference

Inheritance diagram for FeatureTest:



Collaboration diagram for FeatureTest:



### Public Member Functions

- [void test \(\)](#)

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

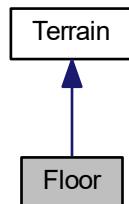
- [test.feature.cpp](#)

## 4.12 Floor Class Reference

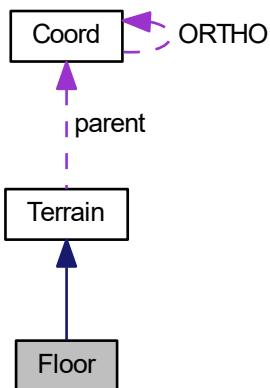
Regular dungeon floor.

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

Inheritance diagram for Floor:



Collaboration diagram for Floor:



### Additional Inherited Members

#### 4.12.1 Detailed Description

Regular dungeon floor.

Has full visibility and passability.

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

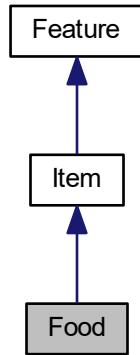
- include/tiles.h
- tiles.cpp

## 4.13 Food Class Reference

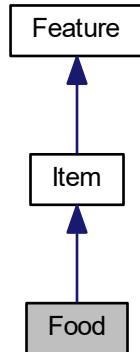
Represents food.

```
#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

### 4.13.1 Detailed Description

Represents food.

### 4.13.2 Constructor & Destructor Documentation

#### 4.13.2.1 Food()

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

Constructs a [Food](#) instance.

##### Parameters

in	<i>location</i>	<a href="#">Food</a> location
in	<i>context</i>	<a href="#">Food</a> context

### 4.13.3 Member Function Documentation

#### 4.13.3.1 activate()

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

Applies the effects derived from eating this [Food](#).

##### Parameters

<i>player</i>	Reference to the PlayerCharacter instance
---------------	---

##### Returns

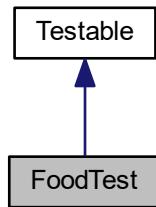
A value reflecting the success of the activation operation.

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

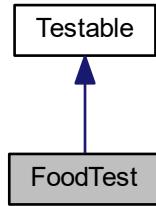
- [include/food.h](#)
- [food.cpp](#)

## 4.14 FoodTest Class Reference

Inheritance diagram for FoodTest:



Collaboration diagram for FoodTest:



### Public Member Functions

- void **test ()**

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

- [test.food.cpp](#)

## 4.15 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 int [randPercent](#) ()  
*Random percent from 0 to 100.*
- static [Coord randPosition](#) ([Coord](#), [Coord](#))  
*Random coord in box delineated by topleft, bottomright.*
- static int [nDx](#) (int numDice, int numFaces)  
*Rolls the designated dice and returns sum.*

### 4.15.1 Detailed Description

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

### 4.15.2 Member Function Documentation

#### 4.15.2.1 [intFromRange\(\)](#)

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

Random integer from range (inclusive).

#### 4.15.2.2 [nDx\(\)](#)

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

Rolls the designated dice and returns sum.

#### 4.15.2.3 [rand\(\)](#)

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

Random double between 0 and 1 (inclusive).

#### 4.15.2.4 randBool()

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

Random boolean.

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

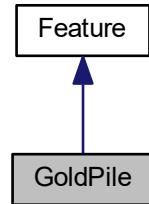
- [include/random.h](#)
- [random.cpp](#)

## 4.16 GoldPile Class Reference

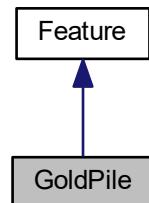
Represents a pile of gold on the ground, which can be picked up by the player to enhance their score.

```
#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.*

## Additional Inherited Members

### 4.16.1 Detailed Description

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

### 4.16.2 Constructor & Destructor Documentation

#### 4.16.2.1 GoldPile()

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

Constructor of location, quantity.

### 4.16.3 Member Function Documentation

#### 4.16.3.1 getQuantity()

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

Getter for quantity.

#### See also

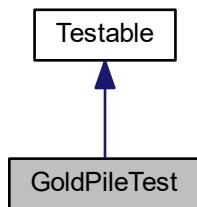
[quantity](#)

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

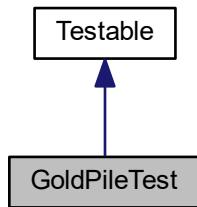
- [include/goldpile.h](#)
- [goldpile.cpp](#)

## 4.17 GoldPileTest Class Reference

Inheritance diagram for GoldPileTest:



Collaboration diagram for GoldPileTest:



### Public Member Functions

- void **test ()**

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

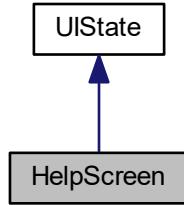
- [test.goldpile.cpp](#)

## 4.18 HelpScreen Class Reference

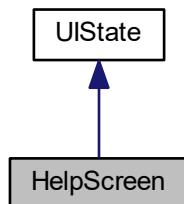
Interface state that shows the various game controls.

```
#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).*

### 4.18.1 Detailed Description

Interface state that shows the various game controls.

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

## 4.18.2 Constructor & Destructor Documentation

### 4.18.2.1 HelpScreen()

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

Constructor.

## 4.18.3 Member Function Documentation

### 4.18.3.1 draw()

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

Render the controls.

Reimplemented from [UIState](#).

### 4.18.3.2 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:

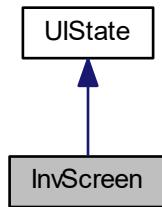
- [include/helpscreen.h](#)
- [helpscreen.cpp](#)

## 4.19 InvScreen Class Reference

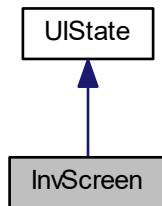
Interface state for viewing the contents of the player inventory.

```
#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, std::string prompt="")`  
*Constructor.*
- `void draw (TCODConsole *)`  
*Draw the inventory.*
- `UIState * handleInput (TCOD_key_t)`  
*Handle input (just the quit key).*

### 4.19.1 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)

## 4.19.2 Constructor & Destructor Documentation

### 4.19.2.1 InvScreen()

```
InvScreen::InvScreen (
    PlayerChar * player,
    Level * level,
    filtFunc filter,
    transFunc trans,
    bool escapeable,
    std::string prompt = "" )
```

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.

## 4.19.3 Member Function Documentation

### 4.19.3.1 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](#).

### 4.19.3.2 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:

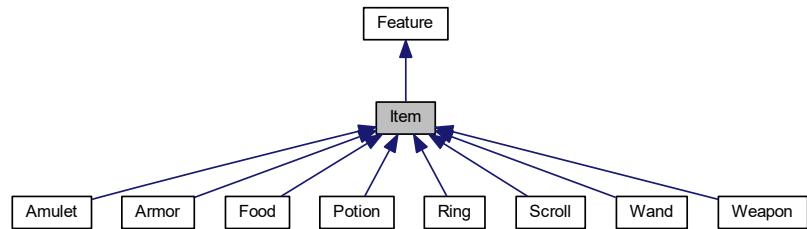
- [include/invscreen.h](#)
- [invscreen.cpp](#)

## 4.20 Item Class Reference

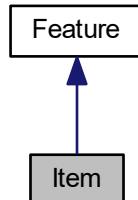
Represents a generic item.

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

Inheritance diagram for Item:



Collaboration diagram for Item:



### Public Types

- enum `Context` { **FLOOR**, **INVENTORY** }  
*Placement context of this `Item`.*
- enum `Effect` { **CURSED**, **PROTECTED** }

### Public Member Functions

- `Item` (char, `Coord`, `Context`, std::string, std::string, int, bool, bool, int)  
*Constructs an `Item` instance.*
- `Item` (char, `Coord`, `Context`, std::string, std::string, std::string, int, bool, bool, int)  
*Constructs an `Item` instance.*
- bool `operator==` (const `Item` &) const  
*`Item` equality definition.*

- bool `operator<` (const `Item` &) const  
*Item 'less than' comparison definition.*
- virtual void `applyEffect` (`Effect`)  
*Applies the given effect to the `Item`.*
- `Context getContext ()`  
*Gets the context.*
- int `getWeight ()`  
*Gets the item's weight.*
- std::string `getClassName ()`  
*Gets the subclass name.*
- bool `hasEffect` (`Effect`)  
*Determines if it the `Item` has the provided effect.*
- bool `isCursed ()`  
*Determines if this `Item` is cursed.*
- virtual void `removeEffect` (`Effect`)  
*Removes the given effect from the `Item`.*
- 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** = 3

## Protected Attributes

- std::set< Effect > **effects**  
*Effects applied to the [Item](#).*
- 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](#).*
- std::string **name**  
*Name of this [Item](#).*
- std::string **pseudoName**  
*Name of the unidentified version of this [Item](#).*
- int **type**  
*Type of this [Item](#).*
- int **weight**  
*How much the item weighs (unit unspecified)*

## Static Protected Attributes

- static std::map< std::string, std::map< int, bool > > **identified**  
*Identification map of the following form: {Class Name : {Type : Status}}.*

### 4.20.1 Detailed Description

Represents a generic item.

### 4.20.2 Constructor & Destructor Documentation

#### 4.20.2.1 [Item\(\)](#) [1/2]

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

Constructs an [Item](#) instance.

### Parameters

in	<i>symbol</i>	Character denoting this <a href="#">Item</a>
in	<i>location</i>	<a href="#">Item</a> location
in	<i>context</i>	<a href="#">Item</a> context
in	<i>className</i>	Name of <a href="#">Item</a> subclass using this constructor
in	<i>name</i>	<a href="#">Item</a> name
in	<i>type</i>	<a href="#">Item</a> type
in	<i>canStack</i>	Denotes whether or not this <a href="#">Item</a> can be stacked in the inventory
in	<i>canThrow</i>	Denotes whether or not this <a href="#">Item</a> can be thrown
in	<i>weight</i>	How much the item weighs

#### 4.20.2.2 [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,
    int weight )
```

Constructs an [Item](#) instance.

### Parameters

in	<i>symbol</i>	Character denoting this <a href="#">Item</a>
in	<i>location</i>	<a href="#">Item</a> location
in	<i>context</i>	<a href="#">Item</a> context
in	<i>className</i>	Name of <a href="#">Item</a> subclass using this constructor
in	<i>name</i>	<a href="#">Item</a> name
in	<i>pseudoName</i>	Unidentified <a href="#">Item</a> name
in	<i>type</i>	<a href="#">Item</a> type
in	<i>canStack</i>	Denotes whether or not this <a href="#">Item</a> can be stacked in the inventory
in	<i>canThrow</i>	Denotes whether or not this <a href="#">Item</a> can be thrown
in	<i>weight</i>	How much the item weighs

### 4.20.3 Member Function Documentation

#### 4.20.3.1 [applyEffect\(\)](#)

```
void Item::applyEffect (
    Item::Effect effect ) [virtual]
```

Applies the given effect to the [Item](#).

**Parameters**

in	<i>effect</i>	Effect to be applied
----	---------------	----------------------

**4.20.3.2 `getClassName()`**

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

Gets the subclass name.

**Returns**

The subclass name.

**4.20.3.3 `getContext()`**

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

Gets the context.

**Returns**

The context.

**4.20.3.4 `getDisplayName()`**

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

Gets the display name.

**Returns**

The display name.

**4.20.3.5 `getName()`**

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

Gets the name.

**Returns**

The name.

#### 4.20.3.6 `getType()`

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

Gets the type.

##### Returns

The type.

#### 4.20.3.7 `getWeight()`

```
int Item::getWeight ( )
```

Gets the item's weight.

##### Returns

The weight

#### 4.20.3.8 `hasEffect()`

```
bool Item::hasEffect (   
    Item::Effect effect )
```

Determines if it the [Item](#) has the provided effect.

##### Parameters

in	<i>effect</i>	Effect to be checked
----	---------------	----------------------

##### Returns

True if the [Item](#) has the given effect, False otherwise.

#### 4.20.3.9 `isCursed()`

```
bool Item::isCursed ( )
```

Determines if this [Item](#) is cursed.

##### Returns

True if cursed, False otherwise.

#### 4.20.3.10 isIdentified()

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

Determines if the [Item](#) is identified.

##### Returns

True if identified, False otherwise.

#### 4.20.3.11 isStackable()

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

Determines if the [Item](#) is stackable.

##### Returns

True if stackable, False otherwise.

#### 4.20.3.12 isThrowable()

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

Determines if [Item](#) is throwable.

##### Returns

True if throwable, False otherwise.

#### 4.20.3.13 operator<()

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

[Item](#) 'less than' comparison definition.

##### Parameters

in	<i>item</i>	Other comparison operand
----	-------------	--------------------------

##### Returns

True if this [Item](#) is less than the given [Item](#), False otherwise

#### 4.20.3.14 operator==( )

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

[Item](#) equality definition.

##### Parameters

in	<i>item</i>	Other equality operand
----	-------------	------------------------

##### Returns

True if this [Item](#) is equivalent to the given [Item](#), False otherwise

#### 4.20.3.15 removeEffect( )

```
void Item::removeEffect (
    Item::Effect effect ) [virtual]
```

Removes the given effect from the [Item](#).

param[in] effect Effect to be removed

#### 4.20.3.16 setContext( )

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

Sets the context.

##### Parameters

in	<i>context</i>	New <a href="#">Item</a> context
----	----------------	----------------------------------

#### 4.20.3.17 setIdentified( )

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

Sets the identified status of this [Item](#) type.

##### Parameters

in	<i>newValue</i>	New identified status of this <a href="#">Item</a> type.
----	-----------------	--

#### 4.20.3.18 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

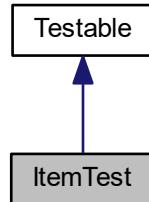
The shuffled copy of the provided vector of names.

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

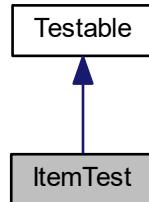
- [include/item.h](#)
- [item.cpp](#)

## 4.21 ItemTest Class Reference

Inheritance diagram for ItemTest:



Collaboration diagram for ItemTest:



## Public Member Functions

- void **test** ()

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

- [test.item.cpp](#)

## 4.22 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.*
- [\*\*bool 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.*
- [\*\*int getCurrWeight\*\* \(\)](#)  
*Return the sum the weight of all inventory contents.*
- [\*\*int getMaxWeight\*\* \(\)](#)  
*Return the maximum inventory capacity.*

### 4.22.1 Detailed Description

Container for items.

#### See also

[Item](#) Tracks stackability and how it relates to capacity, provides utility functions, and tracks persistent hotkeys.

## 4.22.2 Constructor & Destructor Documentation

### 4.22.2.1 ItemZone()

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

Constructor for empty container.

## 4.22.3 Member Function Documentation

### 4.22.3.1 add()

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

Add item to [ItemZone](#), stacking if necessary.

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

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

Check if [ItemZone](#) contains  $\geq 1$  copies of item.

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

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

Check if item with given name is in [ItemZone](#).

### 4.22.3.4 getContents()

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

Return the contents of the zone directly.

### 4.22.3.5 getCurrWeight()

```
int ItemZone::getCurrWeight ( )
```

Return the sum the weight of all inventory contents.

#### 4.22.3.6 getItem()

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

Return struct corresponding to given hotkey.

#### 4.22.3.7 getMaxWeight()

```
int ItemZone::getMaxWeight ( ) [inline]
```

Return the maximum inventory capacity.

#### 4.22.3.8 getSize()

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

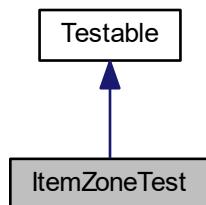
Return the number of distinct items.

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

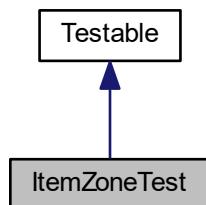
- [include/itemzone.h](#)
- [itemzone.cpp](#)

## 4.23 ItemZoneTest Class Reference

Inheritance diagram for ItemZoneTest:



Collaboration diagram for ItemZoneTest:



## Public Member Functions

- void **test** ()

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

- [test.itemzone.cpp](#)

## 4.24 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 **clear** ()
  - 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.*
  - std::tuple< [Mob](#) \*, int > **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](#), bool)
    - 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](#))

- `std::vector< Coord > getNearestGold (Coord)`  
*Gets the path to the nearest gold.*
- `void placePlayerInStartingPosition ()`  
*Place the player at a random empty position.*
- `Coord getRandomEmptyPosition ()`  
*Return a random coord within the level, with no mobs.*
- `void putRandomMonster ()`  
*Places a depth-appropriate monster somewhere in the level.*

## Static Public Member Functions

- static `Coord getSize ()`

### 4.24.1 Member Function Documentation

#### 4.24.1.1 addFeature()

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

Adds a feature.

##### Parameters

<code>Feature</code>	The feature to add
----------------------	--------------------

#### 4.24.1.2 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

<code>Coord</code>	Starting point
<code>Coord</code>	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](#)

#### 4.24.1.3 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 directions.

Parameters

<a href="#">Coord</a>	Starting point
<a href="#">Coord</a>	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](#)

#### 4.24.1.4 canSee()

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

Determines ability to see each other.

Parameters

in	<a href="#">Coord</a>	A
in	<a href="#">Coord</a>	B

Returns

True if able to see, False otherwise.

#### 4.24.1.5 getAdjPassable()

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

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

#### Parameters

<i>Coord</i>	Coordinate to check from
<i>noMonster</i>	True if tiles with Monsters should be excluded

#### Returns

A vector of coordinates onto which you can move.

#### 4.24.1.6 getFeatures()

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

Gets the features.

#### Returns

The features.

#### 4.24.1.7 getMobs()

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

Gets all the mobs on the level.

#### Returns

The mobs.

#### 4.24.1.8 getNearestGold()

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

Gets the path to the nearest gold.

#### Parameters

in	<i>Coord</i>	Origin to search from
----	--------------	-----------------------

#### Returns

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

**4.24.1.9 getRandomEmptyPosition()**

```
Coord Level::getRandomEmptyPosition ( )
```

Return a random coord within the level, with no mobs.

**Returns**

A random coord within the level, with no mobs.

**4.24.1.10 getRooms()**

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

Gets the rooms.

**Returns**

The rooms.

**4.24.1.11 monsterAt()**

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

Returns the monster that is at the location.

**Parameters**

in	<i>Coord</i>	The location to get the monster from
----	--------------	--------------------------------------

**Returns**

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

**4.24.1.12 popTurnClock()**

```
std::tuple< Mob *, int > Level::popTurnClock ( )
```

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

**Returns**

A mob

#### 4.24.1.13 pushMob()

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

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

**Parameters**

<i>Mob*</i>	Which mob
<i>int</i>	How far to push back in the clock cycle

**4.24.1.14 registerMob()**

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

Adds a mob to the mobs known by the level.

**Parameters**

<i>Mob*</i>	Pointer to the mob that is to be added
-------------	--

**4.24.1.15 removeFeature()**

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

Removes a feature.

**Parameters**

<i>Feature</i>	The feature to remove
----------------	-----------------------

**4.24.1.16 removeMob()**

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

Removes a mob.

**Parameters**

<i>Mob*</i>	Pointer to the mob that is to be removed
-------------	--

**4.24.1.17 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**

<i>Coord</i>	Where the object is being thrown from
<i>Coord</i>	The direction in which it is being thrown (Must be a unit vector!)

**Returns**

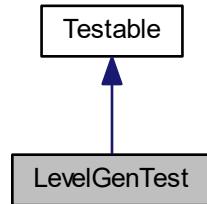
Final location

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

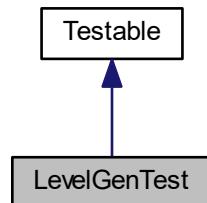
- [include/level.h](#)
- [level.cpp](#)

## 4.25 LevelGenTest Class Reference

Inheritance diagram for LevelGenTest:



Collaboration diagram for LevelGenTest:



## Public Member Functions

- void **test ()**

## Public Attributes

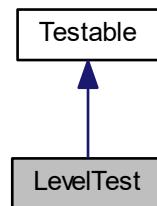
- const int **NUMBER\_OF\_TIMES\_TO\_BFS** = 1000
- const int **NUMBER\_OF\_LEVELS\_TO\_TEST** = 10

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

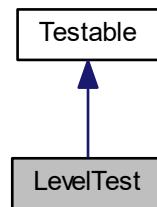
- [test.levelgen.cpp](#)

## 4.26 LevelTest Class Reference

Inheritance diagram for LevelTest:



Collaboration diagram for LevelTest:



## Public Member Functions

- void **test ()**

## Public Attributes

- const int **LEVEL\_DEPTH** = 5
- const int **RANDOM\_TEST\_COUNT** = 5

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

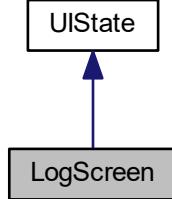
- [test.level.cpp](#)

## 4.27 LogScreen Class Reference

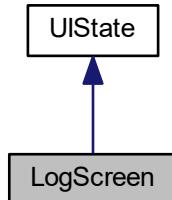
Controls the display of the event log.

```
#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.*

### 4.27.1 Detailed Description

Controls the display of the event log.

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

### 4.27.2 Member Function Documentation

#### 4.27.2.1 draw()

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

Render the previous log messages, up is more recent.

Reimplemented from [UIState](#).

#### 4.27.2.2 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:

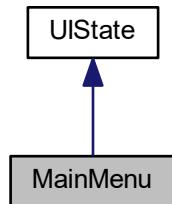
- [include/logscren.h](#)
- [logscren.cpp](#)

## 4.28 MainMenu Class Reference

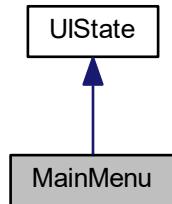
Start screen of the game.

```
#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).*

#### 4.28.1 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)

## 4.28.2 Constructor & Destructor Documentation

### 4.28.2.1 MainMenu()

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

Constructor.

## 4.28.3 Member Function Documentation

### 4.28.3.1 draw()

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

Render the splash art, name prompt.

Reimplemented from [UIState](#).

### 4.28.3.2 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](#)

## 4.29 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.*

#### 4.29.1 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:

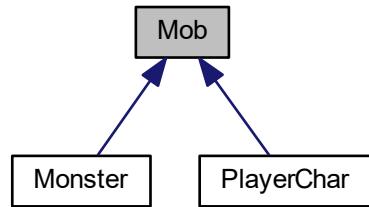
- [include/mastercontroller.h](#)
- [mastercontroller.cpp](#)

## 4.30 Mob Class Reference

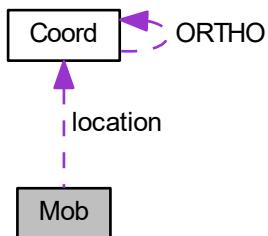
Models a creature in the dungeon, could be the player or a monster.

```
#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, TCODColor)`  
*Constructor.*
- `virtual int calculateDamage ()=0`
- `void changeArmor (int)`  
*Setter for armor.*
- `virtual int getArmorRating ()`  
*Getter for armor.*
- `virtual int getDelay ()=0`  
*Returns this Mob's turn delay.*
- `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.*
- `TCODColor getFColor ()`  
*Getter for the foreground color.*
- `void setFColor (TCODColor)`  
*Setter for the foreground color.*
- `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.*
- TCODColor **fcolor**  
*Foreground color of the mob.*

### 4.30.1 Detailed Description

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

### 4.30.2 Constructor & Destructor Documentation

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

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

Constructor used by monsters.

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

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

Constructor.

#### See also

[armor](#)  
[exp](#)  
[maxHP](#)  
[level](#)

#### 4.30.2.3 ~Mob()

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

Destructor.

### 4.30.3 Member Function Documentation

#### 4.30.3.1 changeArmor()

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

Setter for armor.

See also

[armor](#)

#### 4.30.3.2 getArmorRating()

```
int Mob::getArmorRating ( ) [virtual]
```

Getter for armor.

See also

[armor](#)

Reimplemented in [Monster](#).

#### 4.30.3.3 getDelay()

```
virtual int Mob::getDelay ( ) [pure virtual]
```

Returns this [Mob](#)'s turn delay.

Returns

The delay.

Implemented in [PlayerChar](#), and [Monster](#).

#### 4.30.3.4 getExperience()

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

Getter for XP.

See also

[exp](#)

#### 4.30.3.5 getFColor()

```
TCODColor Mob::getFColor ()
```

Getter for the foreground color.

See also

[fcolor](#)

#### 4.30.3.6 getHP()

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

Getter for HP.

See also

[currentHP](#)

#### 4.30.3.7 getLevel()

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

Getter for mob level.

See also

[level](#)

#### 4.30.3.8 getLocation()

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

Getter for mob location.

Can be edited because it returns a reference

See also

[location](#)

#### 4.30.3.9 getMaxHP()

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

Getter for max HP.

See also

[maxHP](#)

#### 4.30.3.10 getName()

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

Getter for name.

##### See also

[name](#)

#### 4.30.3.11 getSymbol()

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

Getter for symbol.

##### See also

[symbol](#)

#### 4.30.3.12 hit()

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

Called by other entities when they deal damage.

##### See also

[currentHP](#)

Reimplemented in [PlayerChar](#), and [Monster](#).

#### 4.30.3.13 isDead()

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

Determines if this mob is dead.

##### Returns

True if this mob is dead, false otherwise

#### 4.30.3.14 moveLocation()

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

Add current location and param together.

##### See also

[location](#)

#### 4.30.3.15 setCurrentHP()

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

Setter for current HP.

##### See also

[currentHP](#)

#### 4.30.3.16 setFColor()

```
void Mob::setFColor (   
    TCODColor newcol )
```

Setter for the foreground color.

##### See also

[fcolor](#)

#### 4.30.3.17 setLocation()

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

Setter for location.

##### See also

[location](#)

#### 4.30.3.18 setMaxHP()

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

Setter for max hitpoints.

##### See also

[maxHP](#)

### 4.30.4 Member Data Documentation

#### 4.30.4.1 armor

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

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

#### 4.30.4.2 exp

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

More exp indicates the mob is closer to leveling up.

#### 4.30.4.3 fcolor

```
TCODColor Mob::fcolor [protected]
```

Foreground color of the mob.

#### 4.30.4.4 level

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

Higher level characters are more powerful.

#### 4.30.4.5 location

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

Current location within the level.

#### 4.30.4.6 maxHP

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

Maximum number of hitpoints.

#### 4.30.4.7 name

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

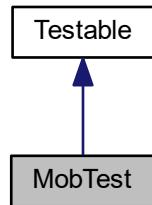
Name of the mob.

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

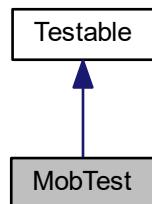
- [include/mob.h](#)
- [mob.cpp](#)

## 4.31 MobTest Class Reference

Inheritance diagram for MobTest:



Collaboration diagram for MobTest:



### Public Member Functions

- [void test \(\)](#)

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

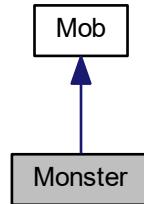
- [test.mob.cpp](#)

## 4.32 Monster Class Reference

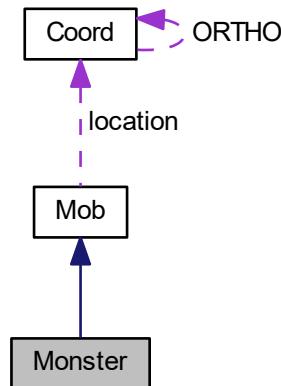
Models a monster in the dungeon.

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

Inheritance diagram for Monster:



Collaboration diagram for Monster:



### Public Types

- enum [Behaviour](#) {
 AGGRESSIVE, CANCELLED, CONFUSED, CONFUSES,
 DISAPPEAR, DRAINS\_LIFE, DROPS\_LEVEL, FLAMES,
 FLYING, FREEZES, FROZEN, GREEDY,
 HASTED, HELD, INVISIBLE, RANGED,
 REGENERATIVE, RUSTS, SLOWED, STATIONARY,
 STINGS
 }

[Monster](#) flags denoting behavioural patterns.

## Public Member Functions

- **Monster (char, Coord)**  
*Constructs a `Monster` instance of the given symbol type.*
- **void addFlag (Monster::Behaviour)**  
*Adds a flag to this `Monster`.*
- **void addFrozenTurns (int)**  
*Freezes the `Monster` for the given number of turns.*
- **void aggregate ()**  
*Aggrevates this monster to attack the player.*
- **void attack (Level \*)**  
*Attempts to attack a nearby Player Character.*
- **void attackConfuse (PlayerChar \*)**  
*Confuse `Monster` attack.*
- **void attackDrainLife (PlayerChar \*)**  
*Drain life `Monster` attack.*
- **void attackDropLevel (PlayerChar \*)**  
*Drop level `Monster` attack.*
- **void attackFreeze (PlayerChar \*)**  
*Freezes `Monster` attack.*
- **void attackRust (PlayerChar \*)**  
*Rust `Monster` attack.*
- **void attackSteal (Level \*)**  
*Gold or item steal `Monster` attack.*
- **void attackSting (PlayerChar \*)**  
*Sting `Monster` attack.*
- **int calculateDamage ()**  
*Calculates the damage of this `Monster`.*
- **int calculateHitChance (PlayerChar \*)**  
*Calculates the hit chance of this `Monster`.*
- **int getArmorRating ()**  
*Returns the effective armor of this `Monster`.*
- **int getCarryChance ()**  
*Gets the carry chance of this `Monster`.*
- **int getDelay ()**  
*Returns this `Monster`'s turn delay.*
- **std::string getName ()**  
*Returns the name of this `Monster`.*
- **std::string getName (PlayerChar \*)**  
*Returns the name of this `Monster` (factoring in invisibility).*
- **bool hasFlag (Behaviour)**
- **virtual void hit (int dmgAmount)**  
*Override mob implementation to aggrevate monster.*
- **bool isAwake ()**  
*Gets the `Monster` awake state.*
- **bool isVisible ()**  
*Determines whether or not this monster should be visible to the `PlayerChar`.*
- **void removeFlag (Monster::Behaviour)**  
*Removes a flag from the `Monster`.*
- **void setAwake (bool)**  
*Sets the `Monster` awake state.*

- void `setVisible` (bool)  
*Sets this `Monster`'s visibility to the given value.*
- 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.*
- static `Monster` \* `randomMonster` ()  
*Creates a random monster.*

## Additional Inherited Members

### 4.32.1 Detailed Description

Models a monster in the dungeon.

### 4.32.2 Constructor & Destructor Documentation

#### 4.32.2.1 `Monster()`

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

Constructs a `Monster` instance of the given symbol type.

#### Parameters

in	<code>symbol</code>	<code>Monster</code> symbol
in	<code>location</code>	<code>Monster</code> location

#### Exceptions

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

### 4.32.3 Member Function Documentation

#### 4.32.3.1 `addFlag()`

```
void Monster::addFlag (
    Monster::Behaviour flag )
```

Adds a flag to this [Monster](#).

**Parameters**

in	<i>flag</i>	Flag to add to this <a href="#">Monster</a> .
----	-------------	---

**4.32.3.2 addFrozenTurns()**

```
void Monster::addFrozenTurns (
    int turns )
```

Freezes the [Monster](#) for the given number of turns.

**Parameters**

in	<i>turns</i>	Number of turns to freeze the <a href="#">Monster</a>
----	--------------	---

**4.32.3.3 attack()**

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

Attempts to attack a nearby Player Character.

**Parameters**

<i>level</i>	Reference to the current <a href="#">Level</a>
--------------	--

**4.32.3.4 attackConfuse()**

```
void Monster::attackConfuse (
    PlayerChar * player )
```

Confuse [Monster](#) attack.

**Parameters**

<i>player</i>	Reference to the <a href="#">PlayerChar</a>
---------------	---

**4.32.3.5 attackDrainLife()**

```
void Monster::attackDrainLife (
    PlayerChar * player )
```

Drain life [Monster](#) attack.

**Parameters**

<i>player</i>	Reference to the <a href="#">PlayerChar</a>
---------------	---

**4.32.3.6 attackDropLevel()**

```
void Monster::attackDropLevel (
    PlayerChar * player )
```

Drop level [Monster](#) attack.

**Parameters**

<i>player</i>	Reference to the <a href="#">PlayerChar</a>
---------------	---

**4.32.3.7 attackFreeze()**

```
void Monster::attackFreeze (
    PlayerChar * player )
```

Freezes [Monster](#) attack.

**Parameters**

<i>player</i>	Reference to the <a href="#">PlayerChar</a>
---------------	---

**4.32.3.8 attackRust()**

```
void Monster::attackRust (
    PlayerChar * player )
```

Rust [Monster](#) attack.

**Parameters**

<i>player</i>	Reference to the <a href="#">PlayerChar</a>
---------------	---

**4.32.3.9 attackSteal()**

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

Gold or item steal [Monster](#) attack.

**Parameters**

<i>level</i>	Reference to the current <a href="#">Level</a>
--------------	--

**4.32.3.10 attackSting()**

```
void Monster::attackSting (
    PlayerChar * player )
```

Sting [Monster](#) attack.

**Parameters**

<i>player</i>	Reference to the <a href="#">PlayerChar</a>
---------------	---

**4.32.3.11 calculateDamage()**

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

Calculates the damage of this [Monster](#).

**Returns**

The computed damage.

Implements [Mob](#).

**4.32.3.12 calculateHitChance()**

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

Calculates the hit chance of this [Monster](#).

**Parameters**

<i>Reference</i>	to the player character
------------------	-------------------------

**Returns**

The computed hit chance.

**4.32.3.13 getArmorRating()**

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

Returns the effective armor of this [Monster](#).

**Returns**

The effective armor of this [Monster](#).

Reimplemented from [Mob](#).

**4.32.3.14 getCarryChance()**

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

Gets the carry chance of this [Monster](#).

**Returns**

The carry chance of this [Monster](#).

**4.32.3.15 getDelay()**

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

Returns this [Monster](#)'s turn delay.

**Returns**

The delay.

Implements [Mob](#).

**4.32.3.16 getName() [1/2]**

```
std::string Monster::getName ( )
```

Returns the name of this [Monster](#).

**Returns**

The name of this [Monster](#).

**4.32.3.17 getName() [2/2]**

```
std::string Monster::getName (
    PlayerChar * player )
```

Returns the name of this [Monster](#) (factoring in invisiblity).

**Returns**

The name of this [Monster](#).

**4.32.3.18 getSymbolsForLevel()**

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

Gets the valid [Monster](#) symbols based on the current dungeon depth.

**Parameters**

in	<i>depth</i>	Current dungeon depth
----	--------------	-----------------------

**Returns**

Vector of valid [Monster](#) symbols.

**4.32.3.19 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	<i>depth</i>	Current dungeon depth
----	--------------	-----------------------

**Returns**

Vector of valid [Monster](#) symbols.

**4.32.3.20 hit()**

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

Override mob implementation to aggregate monster.

**See also**

[aggregate](#)

Reimplemented from [Mob](#).

**4.32.3.21 isAwake()**

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

Gets the [Monster](#) awake state.

**Returns**

True if the [Monster](#) is awake, False otherwise.

#### 4.32.3.22 isVisible()

```
bool Monster::isVisible ( )
```

Determines whether or not this monster should be visible to the [PlayerChar](#).

##### Returns

True if it is visible, False otherwise.

#### 4.32.3.23 randomMonster()

```
Monster * Monster::randomMonster ( ) [static]
```

Creates a random monster.

##### Returns

Totally random monster (can be out of depth).

#### 4.32.3.24 removeFlag()

```
void Monster::removeFlag (   
    Monster::Behaviour flag )
```

Removes a flag from the [Monster](#).

param[in] flag Flag to remove from this [Monster](#).

#### 4.32.3.25 setAwake()

```
void Monster::setAwake (   
    bool awake )
```

Sets the [Monster](#) awake state.

##### Parameters

New	awake state
-----	-------------

#### 4.32.3.26 setVisible()

```
void Monster::setVisible (   
    bool visible )
```

Sets this [Monster](#)'s visibility to the given value.

**Parameters**

<i>visible</i>	New value of <a href="#">Monster</a> 's visibility.
----------------	---

**4.32.3.27 turn()**

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

Performs the actions that make up a [Monster](#)'s turn.

**Parameters**

<i>level</i>	Reference to the current <a href="#">Level</a>
--------------	--

**Returns**

Value denoting the consequential turn delay.

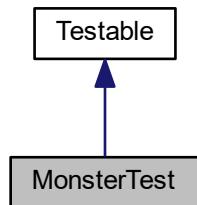
Reimplemented from [Mob](#).

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

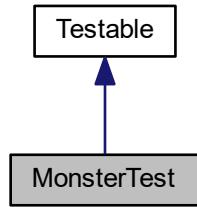
- [include/monster.h](#)
- [monster.cpp](#)

## 4.33 MonsterTest Class Reference

Inheritance diagram for [MonsterTest](#):



Collaboration diagram for MonsterTest:



### Public Member Functions

- void **test** ()

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

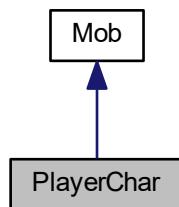
- [test.monster.cpp](#)

## 4.34 PlayerChar Class Reference

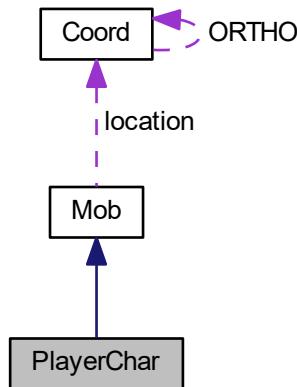
Models the user-controlled player character.

```
#include <playerchar.h>
```

Inheritance diagram for PlayerChar:



Collaboration diagram for PlayerChar:



## Public Types

- enum **Condition** {
 **BLIND**, **CONFUSED**, **CONFUSE\_MONSTER**, **DETECT\_MONSTER**,  
**DETECT\_OBJECTS**, **DIGESTION**, **Fainting**, **HALLUCINATING**,  
**HASTED**, **IMMOBILIZED**, **LEVITATING**, **Maintain\_Armor**,  
**RANDOM\_TELEPORTATION**, **REGENERATION**, **SEARCH**, **SEE\_INVISIBLE**,  
**SLEEPING**, **SLOWED**, **STEALTHY**, **SUSTAIN\_STRENGTH** }

## 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 **applyCondition (Condition, int)**  
*Applies the given condition to the [PlayerChar](#) for the provided number of turns.*
- 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 **changeCurrentStrength (int)**

- `void changeMaxStrength (int)`  
*Increases the current strength of the `PlayerChar` by the passed parameter.*
- `void changeFoodLife (int)`  
*Increases the maximum strength of the `PlayerChar` by the passed parameter.*
- `void collectGold (GoldPile *)`  
*Increases the food life of the `PlayerChar` by the passed parameter.*
- `void dropItem (Item *, Level *)`  
*Adds the gold contained in the given `GoldPile` to the `PlayerChar`'s gold total.*
- `bool dropItem (Item *, Level *)`  
*Attempts to drop the given `Item`.*
- `void dropLevel ()`  
*Decrementsthe level of the `PlayerChar`.*
- `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`.*
- `Armor * getArmor ()`  
*Gets the equipped armor.*
- `int getDexterity ()`  
*Gets the `PlayerChar`'s dexterity.*
- `int getDelay ()`  
*Gets the `PlayerChar`'s turn delay.*
- `float getSearchChance ()`
- `int getSearchRadius ()`  
*Gets the `PlayerChar`'s search radius.*
- `int getFoodLife ()`  
*Gets the `PlayerChar`'s food life.*
- `std::string getFoodStatus ()`  
*Gets the `PlayerChar`'s food status.*
- `int getGold ()`  
*Gets the `PlayerChar`'s gold total.*
- `ItemZone & getInventory ()`  
*Gets the `PlayerChar`'s inventory.*
- `int getLevel ()`  
*Gets the level of this `PlayerChar`.*
- `std::pair< Ring *, Ring * > getRings ()`  
*Gets the references to the equipped Rings.*
- `int getStrength ()`  
*Gets the `PlayerChar`'s strength.*
- `int getMaxStrength ()`  
*Gets the `PlayerChar`'s maximum strength.*
- `int getSightRadius ()`  
*Gets the `PlayerChar`'s sight radius.*
- `Weapon * getWeapon ()`  
*Getter for currently equipped weapon.*
- `bool hasAmulet ()`

- bool **hasCondition** (Condition)
 

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

Inflicts HP loss to this *PlayerChar* based on the given damage.
- bool **move** (Coord, Level \*)
 

Relocates the *PlayerChar* and updates the food life.
- bool **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 **raiseLevel** ()
 

Increments the *PlayerChar*'s level.
- bool **removeArmor** ()
 

Attempts to remove the *PlayerChar*'s equipped *Armor*.
- void **removeCondition** (Condition)
 

Removes the given condition from the *PlayerChar*.
- 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*.
- void **setGold** (int)
 

Sets the gold.
- void **setStrength** (int)
 

Sets the strength of the *PlayerChar*.
- int **update** ()
 

Updates this Player's states.
- void **updateHealthRegen** ()
 

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

Gets the *PlayerChar*'s log.
- bool **getSaveFlag** ()
 

Gets the save flag.
- void **setSaveFlag** (bool)
 

Sets the save flag.

## Static Public Member Functions

- static void **clearConditions** ()
 

Clears the current conditions from the *PlayerChar* (regardless of equipped items)

## Friends

- std::string **encode** (PlayerChar \*, Level \*)
- std::tuple< PlayerChar \*, Level \* > **decode** (std::string)

## Additional Inherited Members

### 4.34.1 Detailed Description

Models the user-controlled player character.

### 4.34.2 Constructor & Destructor Documentation

#### 4.34.2.1 PlayerChar()

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

Constructs a [PlayerChar](#) instance.

##### Parameters

in	<i>location</i>	<a href="#">PlayerChar</a> location
in	<i>name</i>	<a href="#">PlayerChar</a> name

### 4.34.3 Member Function Documentation

#### 4.34.3.1 activateItem()

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

Activates the provided item.

##### Parameters

<i>item</i>	<a href="#">Item</a> to be activated
-------------	--------------------------------------

#### 4.34.3.2 addExp()

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

Adds the given experience to the [PlayerChar](#).

##### Parameters

<i>exp</i>	Experience to be added
------------	------------------------

#### 4.34.3.3 appendLog()

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

Appends the given entry to the log.

##### Parameters

in	<i>entry</i>	Entry to be appended to the log.
----	--------------	----------------------------------

#### 4.34.3.4 applyCondition()

```
void PlayerChar::applyCondition (
    PlayerChar::Condition condition,
    int turns )
```

Applies the given condition to the [PlayerChar](#) for the provided number of turns.

##### Parameters

<i>condition</i>	State to be applied to the player
<i>turns</i>	Number of turns the condition should last (-1 for continuous)

#### 4.34.3.5 attack()

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

Attacks the given [Mob](#).

##### Parameters

<i>monster</i>	<a href="#">Monster</a> to be attacked.
----------------	---

#### 4.34.3.6 calculateDamage()

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

Calculates the damage the [PlayerChar](#) will inflict.

##### Returns

The damage to be inflicted.

Implements [Mob](#).

#### 4.34.3.7 calculateHitChance()

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

Calculates the hit chance of the [PlayerChar](#).

##### Parameters

<i>monster</i>	Moster to be hit
----------------	------------------

##### Returns

The chance the [PlayerChar](#) will hit their target.

#### 4.34.3.8 changeCurrentHP()

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

Increases the current HP of the [PlayerChar](#) by the passed parameter.

##### Parameters

<i>amount</i>	Amount to change the current HP.
---------------	----------------------------------

#### 4.34.3.9 changeCurrentStrength()

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

Increases the current strength of the [PlayerChar](#) by the passed parameter.

##### Parameters

<i>amount</i>	Amount to change the current strength.
---------------	--

#### 4.34.3.10 changeFoodLife()

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

Increases the food life of the [PlayerChar](#) by the passed parameter.

##### Parameters

<i>amount</i>	Amount to change the food life.
---------------	---------------------------------

## 4.34.3.11 changeMaxStrength()

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

Increases the maximum strength of the [PlayerChar](#) by the passed parameter.

## Parameters

<i>amount</i>	Amount to change the maximum strength.
---------------	--

## 4.34.3.12 collectGold()

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

Adds the gold contained in the given [GoldPile](#) to the [PlayerChar](#)'s gold total.

## Parameters

<i>goldPile</i>	<a href="#">GoldPile</a> to be harvested.
-----------------	---

## 4.34.3.13 dropItem()

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

Attempts to drop the given [Item](#).

## Parameters

<i>item</i>	<a href="#">Item</a> to be dropped
<i>level</i>	Reference to the current <a href="#">Level</a>

## Returns

True if the [Item](#) was successfully dropped, False otherwise.

## 4.34.3.14 eat()

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

Attempts to eat the given [Food](#).

**Parameters**

<i>food</i>	<a href="#">Food</a> to be eaten.
-------------	-----------------------------------

**4.34.3.15 equipArmor()**

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

Attempts to equip the given [Armor](#).

**Parameters**

<i>armor</i>	<a href="#">Armor</a> to be equipped.
--------------	---------------------------------------

**4.34.3.16 equipRingLeft()**

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

Attempts to equip the given [Ring](#) on the [PlayerChar](#)'s left hand.

**Parameters**

<i>ring</i>	<a href="#">Ring</a> to be equipped.
-------------	--------------------------------------

**4.34.3.17 equipRingRight()**

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

Attempts to equip the given [Ring](#) on the [PlayerChar](#)'s right hand.

**Parameters**

<i>ring</i>	<a href="#">Ring</a> to be equipped.
-------------	--------------------------------------

**4.34.3.18 equipWeapon()**

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

Attempts to equip the given [Weapon](#).

**Parameters**

<code>weapon</code>	<a href="#">Weapon</a> to be equipped.
---------------------	--

**4.34.3.19 getArmor()**

```
Armor * PlayerChar::getArmor ( )
```

Gets the equipped armor.

**Returns**

The armor the [PlayerChar](#) is wearing.

**4.34.3.20 getDelay()**

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

Gets the [PlayerChar](#)'s turn delay.

**Returns**

The turn delay.

Implements [Mob](#).

**4.34.3.21 getDexterity()**

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

Gets the [PlayerChar](#)'s dexterity.

**Returns**

The [PlayerChar](#)'s dexterity.

**4.34.3.22 getFoodLife()**

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

Gets the [PlayerChar](#)'s food life.

**Returns**

The [PlayerChar](#)'s food life.

**4.34.3.23 getFoodStatus()**

```
std::string PlayerChar::getFoodStatus ( )
```

Gets the [PlayerChar](#)'s food status.

**Returns**

This [PlayerChar](#)'s food status (NULL is full)

**4.34.3.24 getGold()**

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

Gets the [PlayerChar](#)'s gold total.

**Returns**

The [PlayerChar](#)'s gold total.

**4.34.3.25 getInventory()**

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

Gets the [PlayerChar](#)'s inventory.

**Returns**

The [PlayerChar](#)'s inventory.

**4.34.3.26 getLevel()**

```
int PlayerChar::getLevel ( )
```

Gets the level of this [PlayerChar](#).

**Returns**

The level of this [PlayerChar](#).

**4.34.3.27 getLog()**

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

Gets the [PlayerChar](#)'s log.

**Returns**

The [PlayerChar](#)'s log.

**4.34.3.28 getMaxStrength()**

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

Gets the [PlayerChar](#)'s maximum strength.

**Returns**

The [PlayerChar](#)'s maximum strength.

**4.34.3.29 getRings()**

```
std::pair< Ring *, Ring * > PlayerChar::getRings ( )
```

Gets the references to the equipped Rings.

**Returns**

The references to the Rings.

**4.34.3.30 getSaveFlag()**

```
bool PlayerChar::getSaveFlag ( )
```

Gets the save flag.

**Returns**

The save flag.

**See also**

[saveFlag](#)

**4.34.3.31 getSearchRadius()**

```
int PlayerChar::getSearchRadius ( )
```

Gets the [PlayerChar](#)'s search radius.

see [SEARCH\\_RADIUS](#)

**Returns**

Distance (taxicab) in which the player will spot secrets when searching.

#### 4.34.3.32 `getSightRadius()`

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

Gets the [PlayerChar](#)'s sight radius.

##### Returns

The [PlayerChar](#)'s sight radius.

#### 4.34.3.33 `getStrength()`

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

Gets the [PlayerChar](#)'s strength.

##### Returns

The [PlayerChar](#)'s strength.

#### 4.34.3.34 `getWeapon()`

```
Weapon * PlayerChar::getWeapon ( )
```

Getter for currently equipped weapon.

##### Parameters

<code>weapon</code>	<code>Weapon</code> player is wielding (maybe none)
---------------------	---

#### 4.34.3.35 `hasAmulet()`

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

Determines whether or not [PlayerChar](#) has the [Amulet](#) of Yendor.

##### Returns

True if [PlayerChar](#) has the [Amulet](#), False otherwise.

#### 4.34.3.36 `hasCondition()`

```
bool PlayerChar::hasCondition (   
    PlayerChar::Condition condition )
```

Determines whether or not this [PlayerChar](#) is affected by the given condition.

**Parameters**

<i>condition</i>	Condition to check
------------------	--------------------

**Returns**

True if [PlayerChar](#) is currently affected by condition, False otherwise.

**4.34.3.37 hit()**

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

Inflicts HP loss to this [PlayerChar](#) based on the given damage.

**Parameters**

in	<i>damage</i>	Baseline damage to be inflicted
----	---------------	---------------------------------

Reimplemented from [Mob](#).

**4.34.3.38 move()**

```
bool PlayerChar::move (
    Coord location,
    Level * level )
```

Relocates the [PlayerChar](#) and updates the food life.

**Parameters**

<i>location</i>	New <a href="#">PlayerChar</a> location
<i>level</i>	Reference to the current level

**4.34.3.39 pickupItem()**

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

Attempts to place the provided [Item](#) in the [PlayerChar](#)'s inventory.

**Parameters**

<i>item</i>	<a href="#">Item</a> to be inserted into the <a href="#">PlayerChar</a> 's inventory.
-------------	---

**Returns**

Success or failure in picking up item. (true for success)

**4.34.3.40 quaff()**

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

Attempts to apply the effects of the provided [Potion](#) to the given [Mob](#).

**Parameters**

<i>potion</i>	<a href="#">Potion</a> to be quaffed
<i>mob</i>	<a href="#">Mob</a> to quaff the <a href="#">Potion</a>

**4.34.3.41 removeArmor()**

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

Attempts to remove the [PlayerChar](#)'s equipped [Armor](#).

**Returns**

True if the operation was successful, False otherwise.

**4.34.3.42 removeCondition()**

```
void PlayerChar::removeCondition (
    PlayerChar::Condition condition )
```

Removes the given condition from the [PlayerChar](#).

**Parameters**

<i>condition</i>	Condition to remove
------------------	---------------------

**4.34.3.43 removeRingLeft()**

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

Attempts to remove the [PlayerChar](#)'s equipped left [Ring](#).

**Returns**

True if the operation was successful, False otherwise.

#### 4.34.3.44 removeRingRight()

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

Attempts to remove the [PlayerChar](#)'s equipped right [Ring](#).

##### Returns

True if the operation was successful, False otherwise.

#### 4.34.3.45 removeWeapon()

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

Attempts to remove the [PlayerChar](#)'s equipped [Weapon](#).

##### Returns

True if the operation was successful, False otherwise.

#### 4.34.3.46 setDexterity()

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

Sets the [PlayerChar](#)'s dexterity.

##### Parameters

<i>dexterity</i>	The <a href="#">PlayerChar</a> 's new dexterity
------------------	---

#### 4.34.3.47 setFoodLife()

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

Sets the food life of the [PlayerChar](#).

##### Parameters

<i>foodLife</i>	The new food life of the <a href="#">PlayerChar</a>
-----------------	---

#### 4.34.3.48 setGold()

```
void PlayerChar::setGold ( int gold )
```

Sets the gold.

Parameters

in	<i>gold</i>	New amount of gold
----	-------------	--------------------

#### 4.34.3.49 setSaveFlag()

```
void PlayerChar::setSaveFlag (bool flag)
```

Sets the save flag.

See also

[saveFlag](#)

#### 4.34.3.50 setStrength()

```
void PlayerChar::setStrength (int strength)
```

Sets the strength of the [PlayerChar](#).

Parameters

<i>strength</i>	The new strength of the <a href="#">PlayerChar</a>
-----------------	--

#### 4.34.3.51 update()

```
int PlayerChar::update ( )
```

Updates this Player's states.

Returns

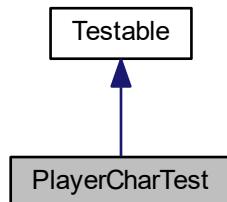
The turn delay.

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

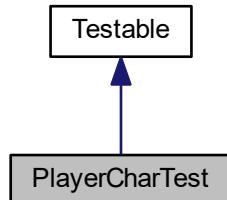
- [include/playerchar.h](#)
- [playerchar.cpp](#)

## 4.35 PlayerCharTest Class Reference

Inheritance diagram for PlayerCharTest:



Collaboration diagram for PlayerCharTest:



### Public Member Functions

- void **test** ()

### Public Attributes

- const int **RANDOM\_TEST\_COUNT** = 5

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

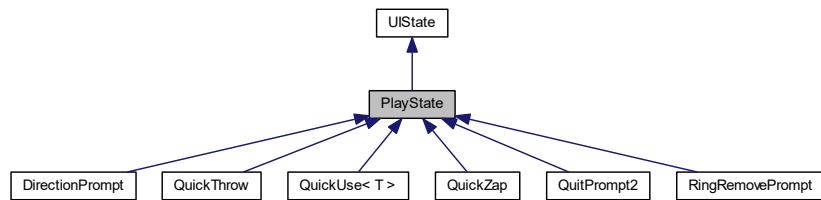
- [test.playerchar.cpp](#)

## 4.36 PlayState Class Reference

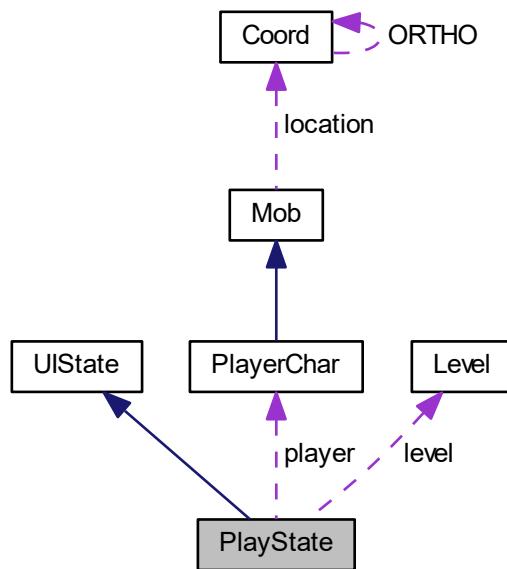
Primary interface state, showing level, player, monsters, etc.

```
#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
- static constexpr auto **NO\_SPACE\_LOG** = "You have no space in your inventory"  
*Log this to the player when they attempt an action and fail due to inventory capacity.*
- static constexpr auto **QUIT\_PROMPT** = "Do you wish to end your quest now (Yes/No) ?"
- static constexpr auto **HAND\_PROMPT** = "Which ring to remove (R/L) ?"
- static constexpr auto **DIRECTION\_PROMPT** = "Which direction?"
- static constexpr auto **HELPLESS\_MSG** = "You are helpless (press SPACE to continue")
- static constexpr auto **DROP\_PROMPT** = "Choose an item to drop"
- static constexpr auto **NO\_QUAFF\_MSG** = "You have nothing you can quaff"
- static constexpr auto **QUAFF\_PROMPT** = "Choose a potion to quaff"
- static constexpr auto **NO\_READ\_MSG** = "You have nothing you can read"
- static constexpr auto **READ\_PROMPT** = "Choose a scroll to read"
- static constexpr auto **ALREADY\_WIELD** = "You are already wielding something"
- static constexpr auto **WIELD\_PROMPT** = "Choose a weapon to wield"
- static constexpr auto **NO\_WIELD\_MSG** = "You have nothing you can wield"
- static constexpr auto **ALREADY\_WEAR** = "You are already wearing something"
- static constexpr auto **WEAR\_PROMPT** = "Choose a piece of armor to wear"
- static constexpr auto **NO\_WEAR\_MSG** = "You have nothing you can wear"
- static constexpr auto **NO\_TAKE\_OFF\_MSG** = "You are not wearing anything"
- static constexpr auto **NO\_REMOVE\_MSG** = "You are not wearing any rings"
- static constexpr auto **FINGER\_DEFICIT** = "You have no more fingers"
- static constexpr auto **NO\_PUT\_MSG** = "You have nothing to put on your finger(s)"
- static constexpr auto **PUT\_PROMPT** = "Choose a ring to put on your finger"
- static constexpr auto **NO\_STOW\_MSG** = "You are not wielding anything"
- static constexpr auto **THROW\_PROMPT** = "Choose an item to throw"
- static constexpr auto **NO\_THROW\_MSG** = "You have nothing you can throw"
- static constexpr auto **ZAP\_PROMPT** = "Choose a wand to zap with"
- static constexpr auto **NO\_ZAP\_MSG** = "You have nothing with which to zap"
- static constexpr auto **EAT\_PROMPT** = "Choose a piece of food to eat"
- static constexpr auto **NO\_EAT\_MSG** = "You have nothing to eat"
- static constexpr auto **NO\_ASCEND\_MSG** = "Some magical force prevents your passage upward."
- static constexpr auto **OPEN\_DOOR\_MSG** = "You open the door"
- static constexpr auto **CLOSE\_DOOR\_MSG** = "You close the door"
- static constexpr auto **TIGHTEN\_FINGER** = "tightens its grip on your finger"
- static constexpr auto **LOOSEN\_GRIP** = "You cannot loosen your grip on the "
- static constexpr auto **REST\_MSG** = "You rest briefly"
- static constexpr auto **SEARCH\_MSG** = "You search your surroundings for secrets"
- static constexpr auto **SECRET\_MSG** = "You uncover a secret"
- static constexpr auto **SAVE\_ON\_MSG** = "You will save at the end of the this `level`"
- static constexpr auto **SAVE\_OFF\_MSG** = "You won't save at the end of the this `level`"
- static constexpr auto **NO\_DROP\_MSG** = "You have nothing to drop"
- static constexpr auto **ALREADY THERE\_MSG** = "There is already something there"

#### 4.36.1 Detailed Description

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

#### 4.36.2 Constructor & Destructor Documentation

##### 4.36.2.1 PlayState()

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

Constructor.

##### 4.36.2.2 ~PlayState()

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

Delete internal components.

#### 4.36.3 Member Function Documentation

##### 4.36.3.1 handleInput()

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

Handle the various controls.

Reimplemented from [UIState](#).

Reimplemented in [DirectionPrompt](#), [QuickUse< T >](#), [QuickThrow](#), [QuickZap](#), [RingRemovePrompt](#), and [Quit←Prompt2](#).

#### 4.36.4 Member Data Documentation

##### 4.36.4.1 level

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

Reference to current dungeon level.

#### 4.36.4.2 player

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

reference to player character.

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

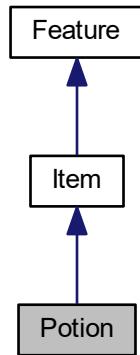
- [include/playstate.h](#)
- [playstate.cpp](#)

## 4.37 Potion Class Reference

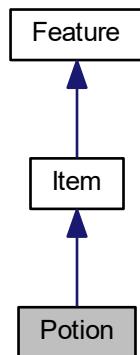
Represents potions.

```
#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

### 4.37.1 Detailed Description

Represents potions.

### 4.37.2 Constructor & Destructor Documentation

#### 4.37.2.1 [Potion\(\)](#) [1/2]

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

Constructs a [Potion](#) instance with a random type.

#### Parameters

in	<i>location</i>	<a href="#">Potion</a> <i>location</i>
----	-----------------	--

#### 4.37.2.2 [Potion\(\)](#) [2/2]

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

Constructs a [Potion](#) instance.

#### Parameters

in	<i>location</i>	<a href="#">Potion</a> <i>location</i>
in	<i>context</i>	<a href="#">Potion</a> <i>context</i>
in	<i>type</i>	<a href="#">Potion</a> <i>type</i>

### 4.37.3 Member Function Documentation

#### 4.37.3.1 activate()

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

Applies the effects derived from quaffing this [Potion](#).

##### Parameters

<i>mob</i>	Reference to the <a href="#">Mob</a> instance
------------	---

##### Returns

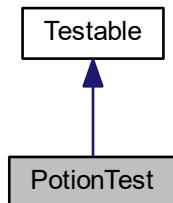
A value reflecting the success of the activation operation.

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

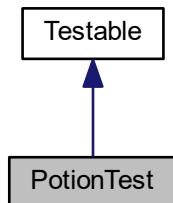
- [include/potion.h](#)
- [potion.cpp](#)

## 4.38 PotionTest Class Reference

Inheritance diagram for [PotionTest](#):



Collaboration diagram for [PotionTest](#):



## Public Member Functions

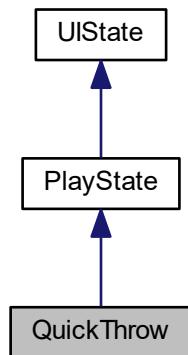
- void **test** ()

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

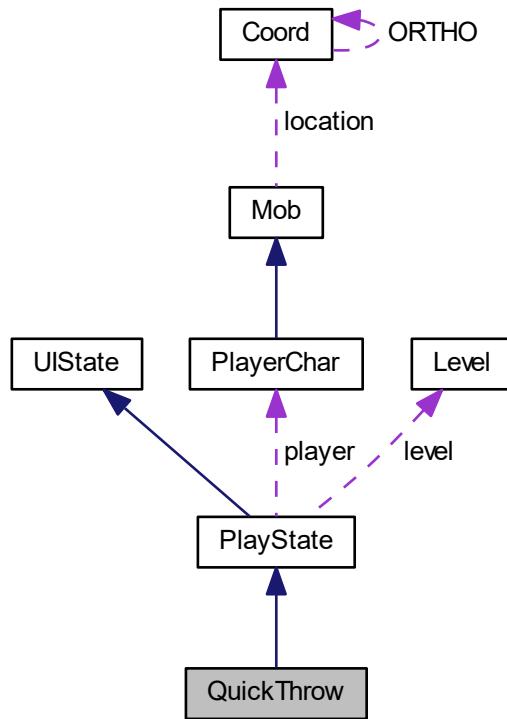
- [test.potion.cpp](#)

## 4.39 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

### 4.39.1 Member Function Documentation

#### 4.39.1.1 handleInput()

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

Handle the various controls.

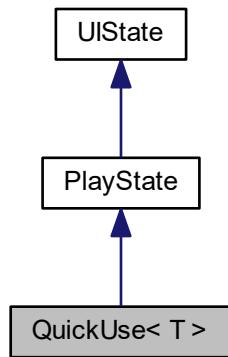
Reimplemented from [PlayState](#).

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

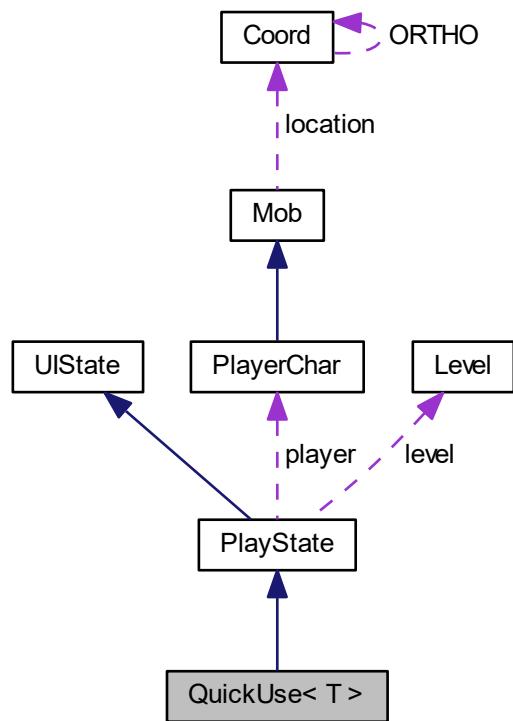
- [playstate.cpp](#)

## 4.40 QuickUse< T > Class Template Reference

Inheritance diagram for QuickUse< T >:



Collaboration diagram for QuickUse< T >:



## Public Member Functions

- **QuickUse** (`PlayerChar *player, Level *level, Item *item, std::function< UIState *(T *)> makeUseOf, bool del=true)`
- virtual `UIState * handleInput (TCOD_key_t key)`

*Handle the various controls.*

## Additional Inherited Members

### 4.40.1 Member Function Documentation

#### 4.40.1.1 handleInput()

```
template<typename T >
virtual UIState* QuickUse< T >::handleInput (
    TCOD_key_t key ) [inline], [virtual]
```

Handle the various controls.

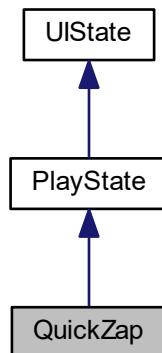
Reimplemented from [PlayState](#).

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

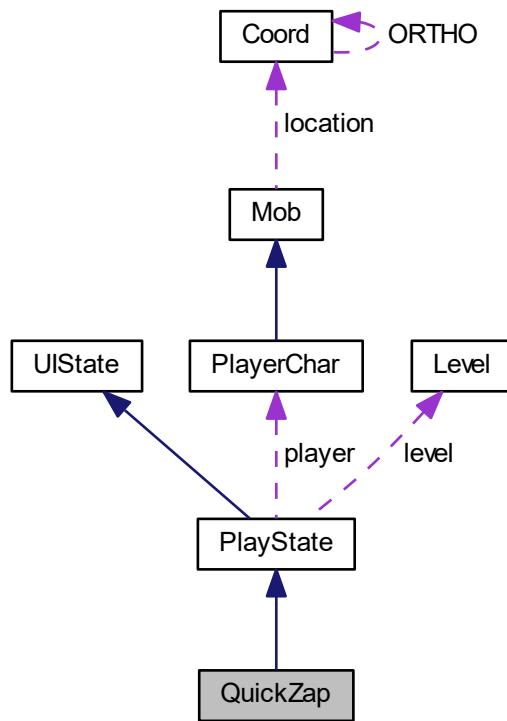
- [playstate.cpp](#)

## 4.41 QuickZap Class Reference

Inheritance diagram for QuickZap:



Collaboration diagram for QuickZap:



## Public Member Functions

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

*Handle the various controls.*

## Additional Inherited Members

### 4.41.1 Member Function Documentation

#### 4.41.1.1 handleInput()

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

*Handle the various controls.*

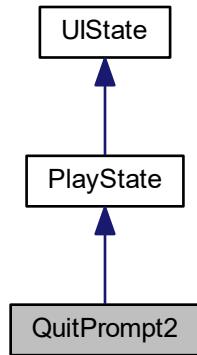
Reimplemented from [PlayState](#).

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

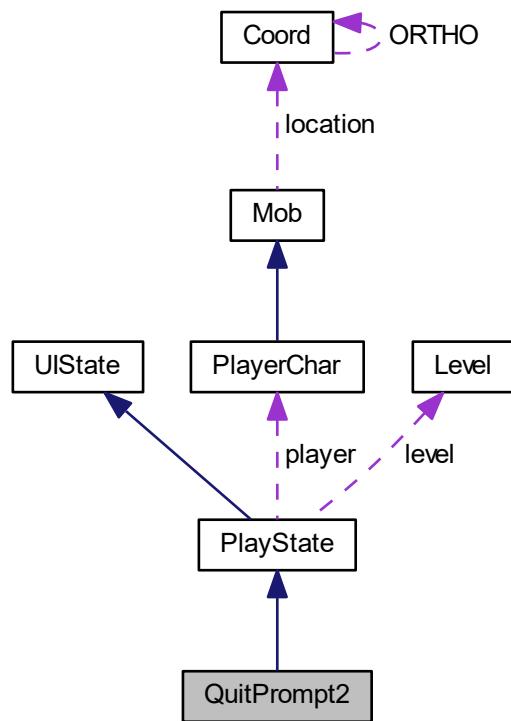
- [playstate.cpp](#)

## 4.42 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

### 4.42.1 Member Function Documentation

#### 4.42.1.1 [handleInput\(\)](#)

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

Handle the various controls.

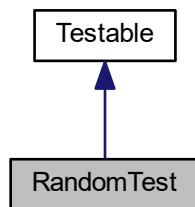
Reimplemented from [PlayState](#).

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

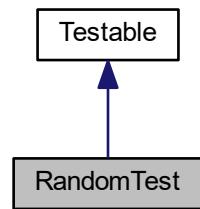
- [playstate.cpp](#)

## 4.43 RandomTest Class Reference

Inheritance diagram for RandomTest:



Collaboration diagram for RandomTest:



## Public Member Functions

- void **test** ()

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

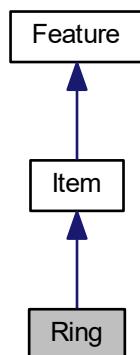
- [test.random.cpp](#)

## 4.44 Ring Class Reference

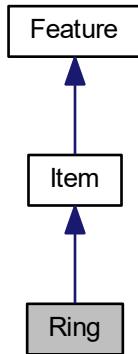
Represents rings.

```
#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 \(PlayerChar \\*\)](#)  
*Applies the effects derived from equipping this `Ring`.*
- [bool deactivate \(PlayerChar \\*\)](#)  
*Deactivates this `Ring`'s effects on the `PlayerChar`.*

## Additional Inherited Members

### 4.44.1 Detailed Description

Represents rings.

### 4.44.2 Constructor & Destructor Documentation

#### 4.44.2.1 `Ring()` [1/2]

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

Constructs a `Ring` instance with a random type.

#### Parameters

in	<code>location</code>	<code>Ring</code> location
----	-----------------------	----------------------------

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

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

Constructs a [Ring](#) instance.

##### Parameters

in	<i>location</i>	<a href="#">Ring</a> location
in	<i>context</i>	<a href="#">Ring</a> context
in	<i>type</i>	<a href="#">Ring</a> type

### 4.44.3 Member Function Documentation

#### 4.44.3.1 activate()

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

Applies the effects derived from equipping this [Ring](#).

##### Parameters

<i>player</i>	Reference to the <a href="#">PlayerChar</a> instance
---------------	--

##### Returns

True if the operation was successful, False otherwise.

#### 4.44.3.2 deactivate()

```
bool Ring::deactivate (
    PlayerChar * player )
```

Deactivates this [Ring](#)'s effects on the [PlayerChar](#).

##### Parameters

<i>player</i>	Reference to the player
---------------	-------------------------

##### Returns

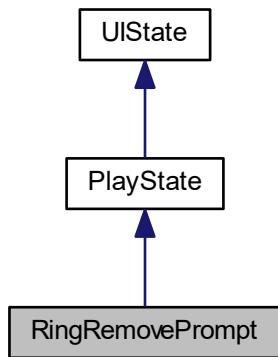
True if the operation was successful, False otherwise.

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

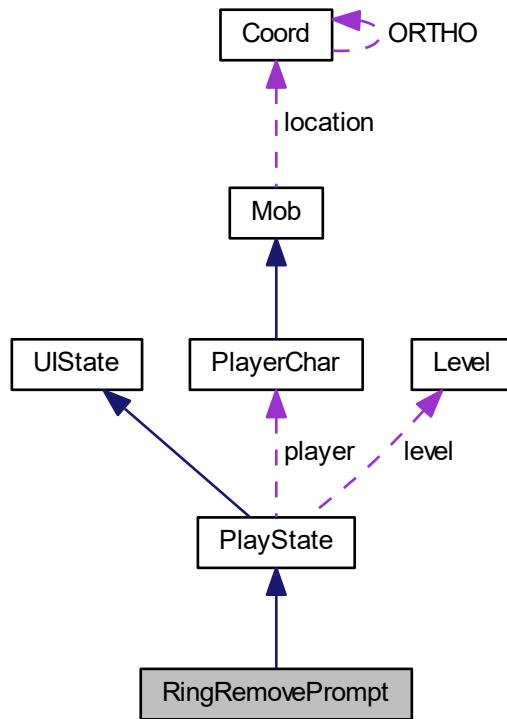
- [include/ring.h](#)
- [ring.cpp](#)

## 4.45 RingRemovePrompt Class Reference

Inheritance diagram for RingRemovePrompt:



Collaboration diagram for RingRemovePrompt:



## Public Member Functions

- **RingRemovePrompt** (`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

### 4.45.1 Member Function Documentation

#### 4.45.1.1 handleInput()

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

Handle the various controls.

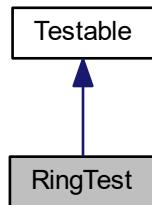
Reimplemented from [PlayState](#).

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

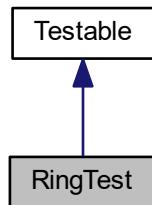
- [playstate.cpp](#)

## 4.46 RingTest Class Reference

Inheritance diagram for RingTest:



Collaboration diagram for RingTest:



### Public Member Functions

- void **test ()**

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

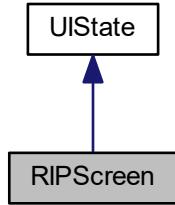
- [test.ring.cpp](#)

## 4.47 RIPScreen Class Reference

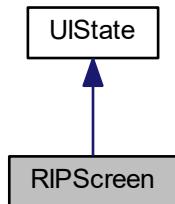
Interface state for post-death/retirement, looking at the high-score table.

```
#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.*

### 4.47.1 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

## 4.47.2 Constructor & Destructor Documentation

### 4.47.2.1 RIPScreen()

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

Constructor.

#### Parameters

<i>cause</i>	Cause of death/retirement
<i>level</i>	<a href="#">Level</a> on which player died/retired

## 4.47.3 Member Function Documentation

### 4.47.3.1 draw()

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

Render.

Reimplemented from [UIState](#).

### 4.47.3.2 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](#)

## 4.48 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** ()

### 4.48.1 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](#)

### 4.48.2 Member Function Documentation

#### 4.48.2.1 **contains()**

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

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

#### Parameters

<i>Coord</i>	The coordinate to test
--------------	------------------------

**Returns**

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

**4.48.2.2 dig()**

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

Clears a passable room in the designated level.

**Parameters**

<i>Level</i>	The level in which to dig
--------------	---------------------------

**4.48.2.3 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.

**4.48.2.4 printInfo()**

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

A diagnostic tool.

**Parameters**

in	<i>An</i>	integer to go along with the info (Used when printing info of multiple rooms).
----	-----------	--

**4.48.2.5 touches()**

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

Tells you whether or not the coordinate touches the room.

**Parameters**

in	<i>Coord</i>	The location to test
----	--------------	----------------------

**Returns**

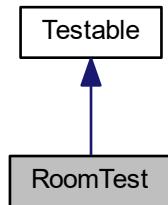
True if coord can touch or intersect with the room, false otherwise

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

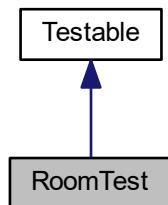
- [include/room.h](#)
- [room.cpp](#)

## 4.49 RoomTest Class Reference

Inheritance diagram for RoomTest:



Collaboration diagram for RoomTest:



### Public Member Functions

- [bool \*\*Ite\*\* \(Coord a, Coord b\)](#)
- [void \*\*test\*\* \(\)](#)

## Public Attributes

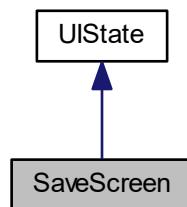
- const int **NUM\_OF\_RANDOM\_TESTS** = 10

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

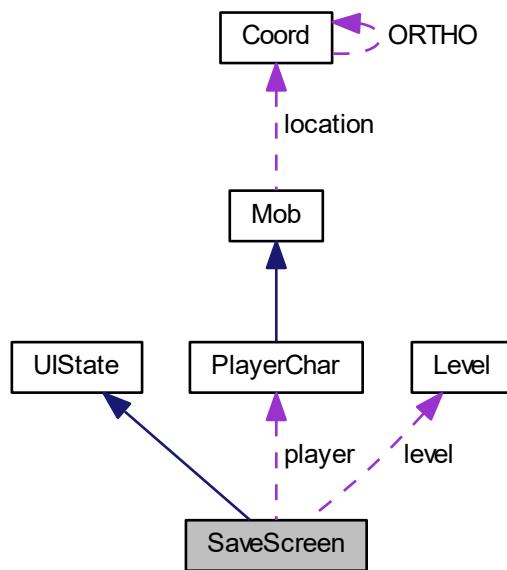
- [test.room.cpp](#)

## 4.50 SaveScreen Class Reference

Inheritance diagram for SaveScreen:



Collaboration diagram for SaveScreen:



## Public Member Functions

- **SaveScreen** ([PlayerChar](#) \*, [Level](#) \*)
- void [draw](#) ([TCODConsole](#) \*con)  
*Render the current UI.*
- [UIState](#) \* [handleInput](#) ([TCOD\\_key\\_t](#))  
*Do whatever is needed in response to keypresses then return state to transition to (can be self).*

## Protected Attributes

- [PlayerChar](#) \* **player**
- [Level](#) \* **level**
- std::string **nameBuffer**

## Static Protected Attributes

- static const std::string **PROMPT** = "Enter save file: "
- static const std::string **KEY\_HINT** = "Press ESCAPE to continue"

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

- include/[savescreen.h](#)
- [savescreen.cpp](#)

## 4.51 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:

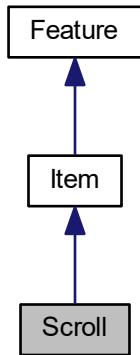
- [ripscreen.cpp](#)

## 4.52 Scroll Class Reference

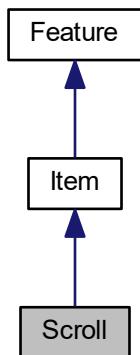
Represents scrolls.

```
#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.*
- [std::tuple< bool, UIState \\* > 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`.*

## Static Public Attributes

- [static bool ignoreTransitions = false](#)  
*Controls whether or not state transitions occur in `activate`.*

## Additional Inherited Members

### 4.52.1 Detailed Description

Represents scrolls.

### 4.52.2 Constructor & Destructor Documentation

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

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

Constructs a `Scroll` instance with a random type.

##### Parameters

in	<i>location</i>	<code>Scroll</code> location
----	-----------------	------------------------------

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

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

Constructs a `Scroll` instance.

**Parameters**

<i>in</i>	<i>location</i>	<a href="#">Scroll</a> location
<i>in</i>	<i>context</i>	<a href="#">Scroll</a> context
<i>in</i>	<i>type</i>	<a href="#">Scroll</a> type

**4.52.3 Member Function Documentation****4.52.3.1 activate()**

```
std::tuple< bool, UIState * > Scroll::activate (
    Level * level )
```

Applies the effects derived from reading this [Scroll](#).

**Parameters**

<i>level</i>	Reference to the <a href="#">Level</a> instance
--------------	---

**Returns**

A value reflecting the success of the activation operation.

**4.52.3.2 initializeScrollNames()**

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

Initializes the unidentified names of each [Scroll](#).

**Returns**

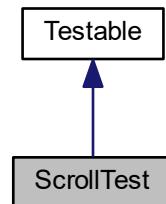
Returns a vector of strings denoting random [Scroll](#) names indexed by type.

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

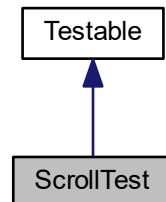
- [include/scroll.h](#)
- [scroll.cpp](#)

## 4.53 ScrollTest Class Reference

Inheritance diagram for ScrollTest:



Collaboration diagram for ScrollTest:



### Public Member Functions

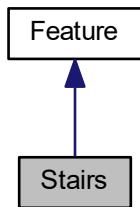
- void **test** ()

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

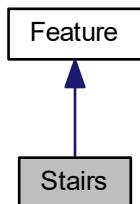
- [test.scroll.cpp](#)

## 4.54 Stairs Class Reference

Inheritance diagram for Stairs:



Collaboration diagram for Stairs:



### Public Member Functions

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

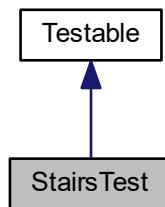
### Additional Inherited Members

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

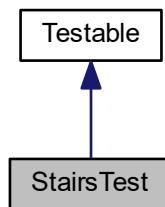
- include/stairs.h
- stairs.cpp

## 4.55 StairsTest Class Reference

Inheritance diagram for StairsTest:



Collaboration diagram for StairsTest:



### Public Member Functions

- void **test** ()

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

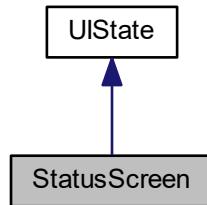
- [test.stairs.cpp](#)

## 4.56 StatusScreen Class Reference

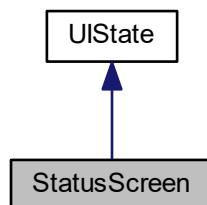
Screen which shows detailed player statistics.

```
#include <statusscreen.h>
```

Inheritance diagram for StatusScreen:



Collaboration diagram for StatusScreen:



## Public Member Functions

- [StatusScreen \(PlayerChar \\*, Level \\*\)](#)  
*Constructor.*
- [virtual void draw \(TCODConsole \\*\)](#)  
*Render the controls.*
- [virtual UIState \\* handleInput \(TCOD\\_key\\_t\)](#)  
*Handle the player input (just quitting).*

### 4.56.1 Detailed Description

Screen which shows detailed player statistics.

### 4.56.2 Constructor & Destructor Documentation

#### 4.56.2.1 StatusScreen()

```
StatusScreen::StatusScreen (
```

`PlayerChar * pc,`  
`Level * lvl )`

Constructor.

### 4.56.3 Member Function Documentation

#### 4.56.3.1 draw()

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

Render the controls.

Reimplemented from [UIState](#).

#### 4.56.3.2 handleInput()

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

Handle the player input (just quitting).

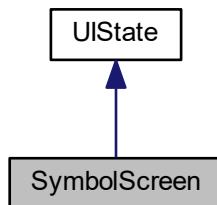
Reimplemented from [UIState](#).

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

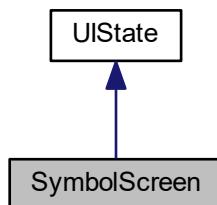
- [include/statusscreen.h](#)
- [statusscreen.cpp](#)

## 4.57 SymbolScreen Class Reference

Inheritance diagram for SymbolScreen:



Collaboration diagram for SymbolScreen:



## Public Member Functions

- [SymbolScreen \(PlayerChar \\*, Level \\*\)](#)  
*Constructor.*
- [virtual void draw \(TCODConsole \\*\)](#)  
*Render the various symbols and explanations.*
- [virtual UIState \\* handleInput \(TCOD\\_key\\_t\)](#)  
*Handle the player input (just quitting).*

### 4.57.1 Constructor & Destructor Documentation

#### 4.57.1.1 SymbolScreen()

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

Constructor.

### 4.57.2 Member Function Documentation

#### 4.57.2.1 draw()

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

Render the various symbols and explanations.

Reimplemented from [UIState](#).

#### 4.57.2.2 handleInput()

```
UIState * SymbolScreen::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:

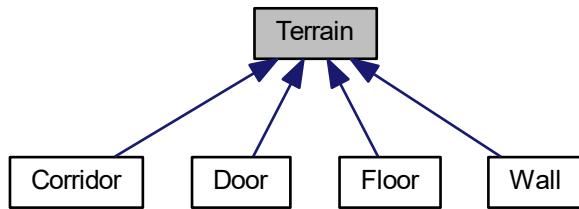
- [include/symbolscreen.h](#)
- [symbolscreen.cpp](#)

## 4.58 Terrain Class Reference

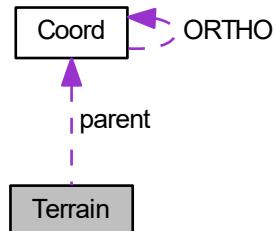
Represents a tile in the dungeon.

```
#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.`
- `Terrain (char, Visibility, Passability, TCODColor)`  
`Alternative constructor for tiles with a specific color.`
- `char getSymbol ()`  
`Getter for character.`
- `void setSymbol (char)`  
`Setter for character.`
- `Passability isPassable ()`  
`Getter for passable.`
- `void setPassable (Passability)`  
`Setter for passable.`
- `TCODColor getColor ()`  
`Getter for foreground color.`
- `Mapped isSeen ()`  
`Getter for seen.`
- `Visibility getVisibility ()`  
`Getter for visible.`
- `void setIsSeen (Mapped)`  
`Setter for seen.`
- `virtual ~Terrain ()`  
`Destructor.`

## Public Attributes

- `bool checked = false`  
`Used by other modules for various searches.`
- `Coord parent`  
`Used by other modules for various searches.`
- `char character`  
`Symbol which represents the tile on the display.`
- `Visibility visible`  
`Degree of transparent of the tile.`
- `Passability passable`  
`Whether the tile can be walked through.`

### 4.58.1 Detailed Description

Represents a tile in the dungeon.

### 4.58.2 Member Enumeration Documentation

#### 4.58.2.1 Mapped

```
enum Terrain::Mapped
```

Whether the player has previous seen the tile.

### 4.58.2.2 Passability

```
enum Terrain::Passability
```

Tiles can be walk-through-able or not.

## 4.58.3 Constructor & Destructor Documentation

### 4.58.3.1 Terrain() [1/2]

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

Constructor.

### 4.58.3.2 Terrain() [2/2]

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

Alternative constructor for tiles with a specific color.

### 4.58.3.3 ~Terrain()

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

Destructor.

## 4.58.4 Member Function Documentation

### 4.58.4.1 getColor()

```
TCODColor Terrain::getColor ( )
```

Getter for foreground color.

#### See also

color

#### 4.58.4.2 `getSymbol()`

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

Getter for character.

##### See also

[character](#)

#### 4.58.4.3 `getVisibility()`

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

Getter for visible.

##### See also

[visible](#)

#### 4.58.4.4 `isPassable()`

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

Getter for passable.

##### See also

[passable](#)

#### 4.58.4.5 `isSeen()`

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

Getter for seen.

##### See also

[seen](#)

#### 4.58.4.6 `setIsSeen()`

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

Setter for seen.

##### See also

[seen](#)

#### 4.58.4.7 setPassable()

```
void Terrain::setPassable (
    Passability pass )
```

Setter for passable.

##### See also

[passable](#)

#### 4.58.4.8 setSymbol()

```
void Terrain::setSymbol (
    char newSymbol )
```

Setter for character.

##### See also

[character](#)

### 4.58.5 Member Data Documentation

#### 4.58.5.1 character

```
char Terrain::character
```

Symbol which represents the tile on the display.

#### 4.58.5.2 checked

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

Used by other modules for various searches.

##### See also

[parent](#)

#### 4.58.5.3 parent

```
Coord Terrain::parent
```

Used by other modules for various searches.

##### See also

[checked](#)

#### 4.58.5.4 passable

**Passability** Terrain::passable

Whether the tile can be walked through.

#### 4.58.5.5 visible

**Visibility** Terrain::visible

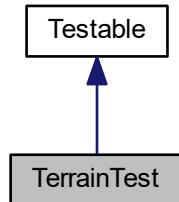
Degree of transparent of the tile.

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

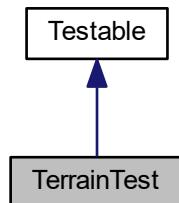
- [include/terrain.h](#)
- [terrain.cpp](#)

## 4.59 TerrainTest Class Reference

Inheritance diagram for TerrainTest:



Collaboration diagram for TerrainTest:



## Public Member Functions

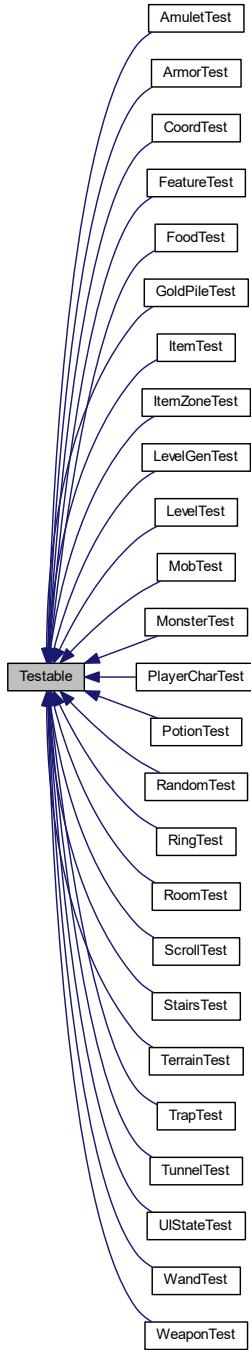
- void **test** ()

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

- [test.terrain.cpp](#)

## 4.60 Testable Class Reference

Inheritance diagram for Testable:



### Public Member Functions

- `virtual void test ()=0`
- `void assert (bool condition, std::string comment)`

- void **comment** (std::string comment)

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

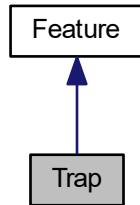
- [test.testable.h](#)
- [test.testable.cpp](#)

## 4.61 Trap Class Reference

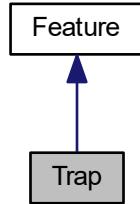
Various hidden traps throughout the dungeon can trigger and endanger the player.

```
#include <trap.h>
```

Inheritance diagram for Trap:



Collaboration diagram for Trap:



### Public Member Functions

- [Trap \(Coord location, unsigned char type, bool visible\)](#)  
*Constructor.*
- [Level \\* activate \(Mob \\*, Level \\*\)](#)  
*Trigger the trap on the given mob.*

## Static Public Member Functions

- static `Trap * randomTrap (Coord)`  
*Creates/returns a random trap at the given coordinates.*

## Static Public Attributes

- static constexpr unsigned int `MAX_TYPE = 5`  
*Records how many type of traps there are for rand-gen purposes.*

## Additional Inherited Members

### 4.61.1 Detailed Description

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

### 4.61.2 Constructor & Destructor Documentation

#### 4.61.2.1 Trap()

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

Constructor.

#### Parameters

<code>location</code>	Position of the trap
<code>type</code>	Type of trap (dart, teleport, pitfall, etc)
<code>visible</code>	Whether the trap is revealed

### 4.61.3 Member Function Documentation

#### 4.61.3.1 activate()

```
Level * Trap::activate (
    Mob * mob,
    Level * level )
```

Trigger the trap on the given mob.

Return the next level if a pitfall, otherwise return the given level.

#### 4.61.3.2 randomTrap()

```
Trap * Trap::randomTrap (  
    Coord location ) [static]
```

Creates/returns a random trap at the given coordinates.

#### 4.61.4 Member Data Documentation

##### 4.61.4.1 MAX\_TYPE

```
constexpr unsigned int Trap::MAX_TYPE = 5 [static]
```

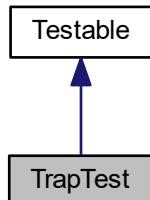
Records how many type of traps there are for rand-gen purposes.

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

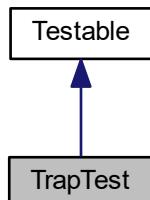
- [include/trap.h](#)
- [trap.cpp](#)

## 4.62 TrapTest Class Reference

Inheritance diagram for TrapTest:



Collaboration diagram for TrapTest:



## Public Member Functions

- void **test** ()

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

- [test.trap.cpp](#)

## 4.63 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](#) \*)  
*Creates a tunnel between the two rooms.*
- void **dig** ([Level](#) &)  
*Digs the specified tunnel in the given level.*

### 4.63.1 Detailed Description

Tunnels are step-orthogonal paths connecting rooms.

### 4.63.2 Constructor & Destructor Documentation

#### 4.63.2.1 Tunnel()

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

Creates a tunnel between the two rooms.

##### Parameters

<i>Room*</i>	The room to go FROM
<i>Room*</i>	The room to go TO

### 4.63.3 Member Function Documentation

#### 4.63.3.1 dig()

```
void Tunnel::dig (
```

<code>Level &amp;</code>	The level in which to dig this tunnel
--------------------------	---------------------------------------

Digs the specified tunnel in the given level.

##### Parameters

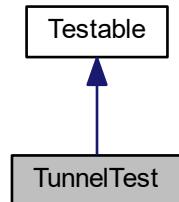
<code>Level&amp;</code>	The level in which to dig this tunnel
-------------------------	---------------------------------------

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

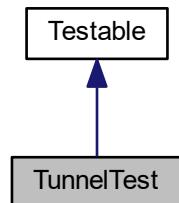
- [include/tunnel.h](#)
- [tunnel.cpp](#)

## 4.64 TunnelTest Class Reference

Inheritance diagram for TunnelTest:



Collaboration diagram for TunnelTest:



## Public Member Functions

- void **test ()**

## Public Attributes

- const int **NUM\_OF\_TESTS = 5**

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

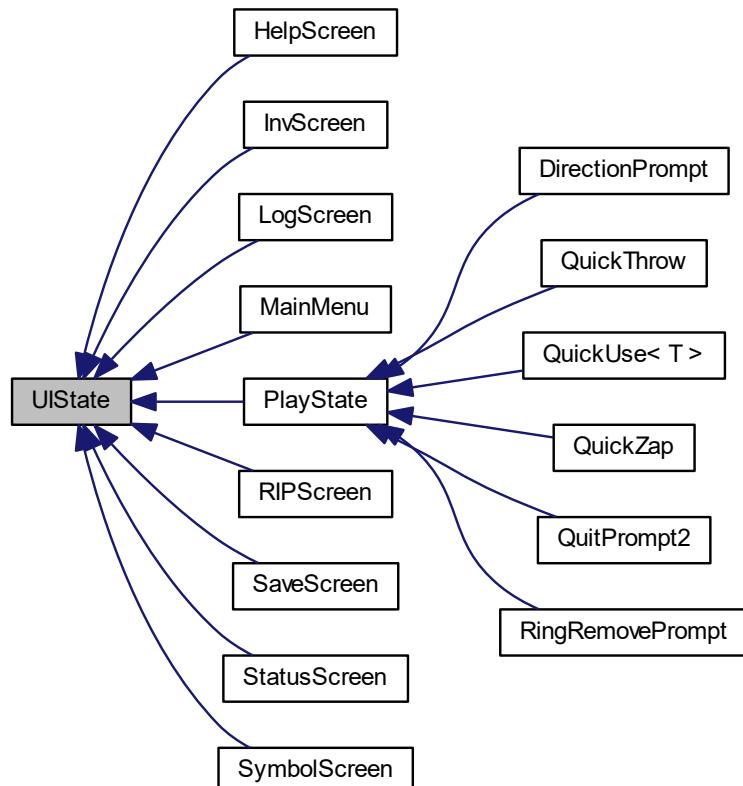
- [test.tunnel.cpp](#)

## 4.65 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.*

### 4.65.1 Detailed Description

Class modeling a state of the game interface.

Game transitions between these states like a finite state machine.

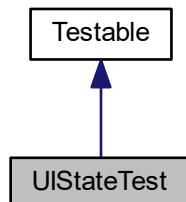
Environment variables: input device (e.g., keyboard) and output device (e.g., monitor)

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

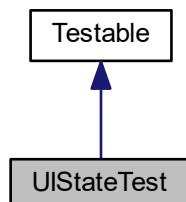
- [include/uistate.h](#)
- [uistate.cpp](#)

## 4.66 UIStateTest Class Reference

Inheritance diagram for UIStateTest:



Collaboration diagram for UIStateTest:



## Public Member Functions

- void **test ()**

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

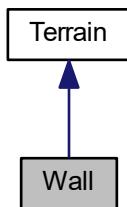
- [test.uistate.cpp](#)

## 4.67 Wall Class Reference

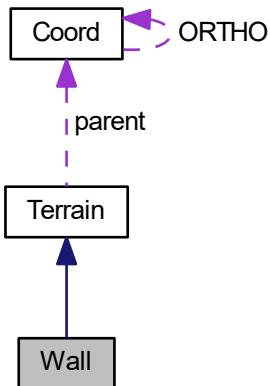
Regular dungeon wall.

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

Inheritance diagram for Wall:



Collaboration diagram for Wall:



## Additional Inherited Members

### 4.67.1 Detailed Description

Regular dungeon wall.

Has no visibility or passability.

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

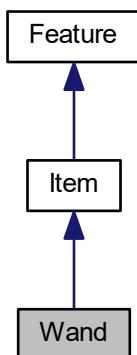
- [include/tiles.h](#)
- [tiles.cpp](#)

## 4.68 Wand Class Reference

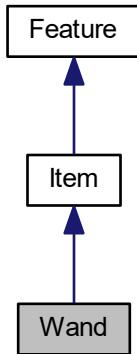
Represents a wand item.

```
#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 *, Mob *)`  
*Applies the effects derived from using a zap from this Wand.*
- `int getCharges ()`  
*Gets the charges.*

## Additional Inherited Members

### 4.68.1 Detailed Description

Represents a wand item.

### 4.68.2 Constructor & Destructor Documentation

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

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

Constructs a Wand instance with a random type.

#### Parameters

in	<code>location</code>	<code>Wand location</code>
----	-----------------------	----------------------------

## 4.68.2.2 Wand() [2/2]

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

Constructs a [Wand](#) instance.

#### Parameters

in	<i>location</i>	<a href="#">Wand</a> location
in	<i>context</i>	<a href="#">Wand</a> context
in	<i>type</i>	<a href="#">Wand</a> type

## 4.68.3 Member Function Documentation

## 4.68.3.1 activate()

```
bool Wand::activate (
    Level * level,
    Mob * mob )
```

Applies the effects derived from using a zap from this [Wand](#).

#### Parameters

<i>level</i>	Reference to the <a href="#">Level</a> instance
<i>mob</i>	<a href="#">Mob</a> to target (NULL for no hit target)

#### Returns

A value reflecting the success of the activation operation.

## 4.68.3.2 getCharges()

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

Gets the charges.

#### Returns

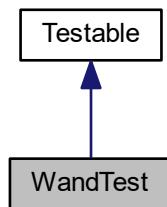
The charges.

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

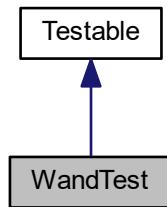
- [include/wand.h](#)
- [wand.cpp](#)

## 4.69 WandTest Class Reference

Inheritance diagram for WandTest:



Collaboration diagram for WandTest:



### Public Member Functions

- void **test ()**

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

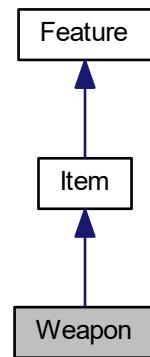
- [test.wand.cpp](#)

## 4.70 Weapon Class Reference

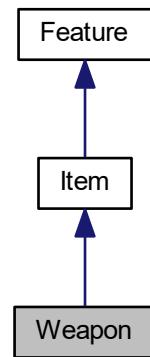
Represents weapons.

```
#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`.*
- [std::pair< int, int > getEnchantments \(\)](#)

- Gets the enchantments.
- bool **isMelee** ()
  - Determines if this [Weapon](#) is a melee weapon.
- void **setEnchantments** (int, int)
  - Sets this [Weapon](#)'s enchantments.
- void **updateName** ()
  - Updates this [Weapon](#)'s name.

## Friends

- std::string **encode** ([PlayerChar](#) \*, [Level](#) \*)
  - std::tuple< [PlayerChar](#) \*, [Level](#) \* > **decode** (std::string)

## Additional Inherited Members

### 4.70.1 Detailed Description

Represents weapons.

### 4.70.2 Constructor & Destructor Documentation

#### 4.70.2.1 [Weapon\(\)](#) [1/2]

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

Constructs a [Weapon](#) instance with a random type.

##### Parameters

in	<i>location</i>	<a href="#">Weapon</a> location
----	-----------------	---------------------------------

#### 4.70.2.2 [Weapon\(\)](#) [2/2]

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

Constructs a [Weapon](#) instance.

##### Parameters

in	<i>location</i>	<a href="#">Weapon</a> location
in	<i>context</i>	<a href="#">Weapon</a> context
in	<i>type</i>	<a href="#">Weapon</a> type

### 4.70.3 Member Function Documentation

#### 4.70.3.1 getChance()

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

Gets the chance of applying a successful hit.

##### Returns

The chance of applying a successful hit.

#### 4.70.3.2 getDamage()

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

Gets the damage triple corresponding to this [Weapon](#).

##### Returns

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

#### 4.70.3.3 getEnchantments()

```
std::pair< int, int > Weapon::getEnchantments ( )
```

Gets the enchantments.

##### Returns

The enchantments.

#### 4.70.3.4 isMelee()

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

Determines if this [Weapon](#) is a melee weapon.

##### Returns

True if melee, False otherwise.

#### 4.70.3.5 setEnchantments()

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

Sets this [Weapon](#)'s enchantments.

**Parameters**

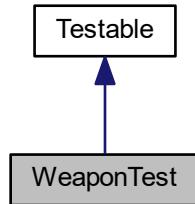
<i>enchantHit</i>	Hit enchantment
<i>enchantDamage</i>	Damage enchantment

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

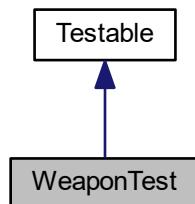
- [include/weapon.h](#)
- [weapon.cpp](#)

## 4.71 WeaponTest Class Reference

Inheritance diagram for WeaponTest:



Collaboration diagram for WeaponTest:



### Public Member Functions

- `void test ()`

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

- [test.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"
```

#### 5.1.1 Detailed Description

Member definitions for the [Amulet](#) class.

##### Author

Team Rogue++

##### Date

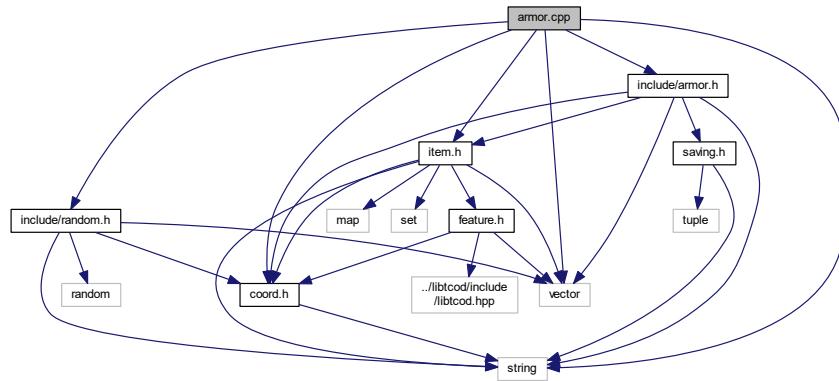
December 06, 2016

### 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

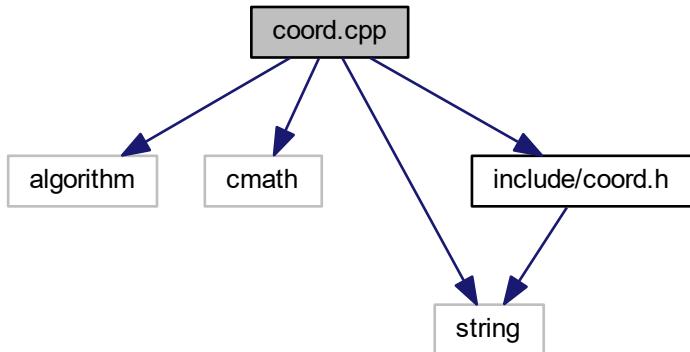
December 06, 2016

## 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

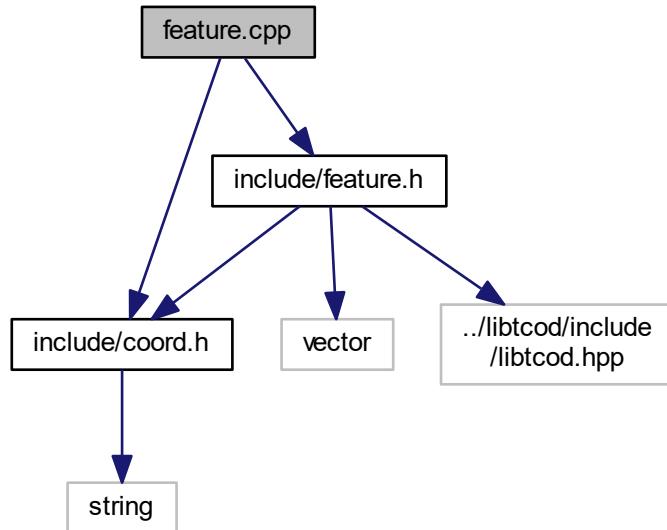
December 06, 2016

## 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

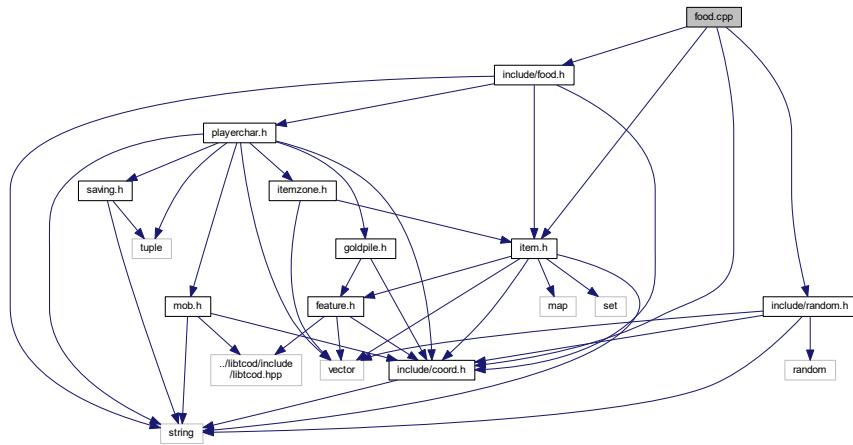
December 06, 2016

### 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

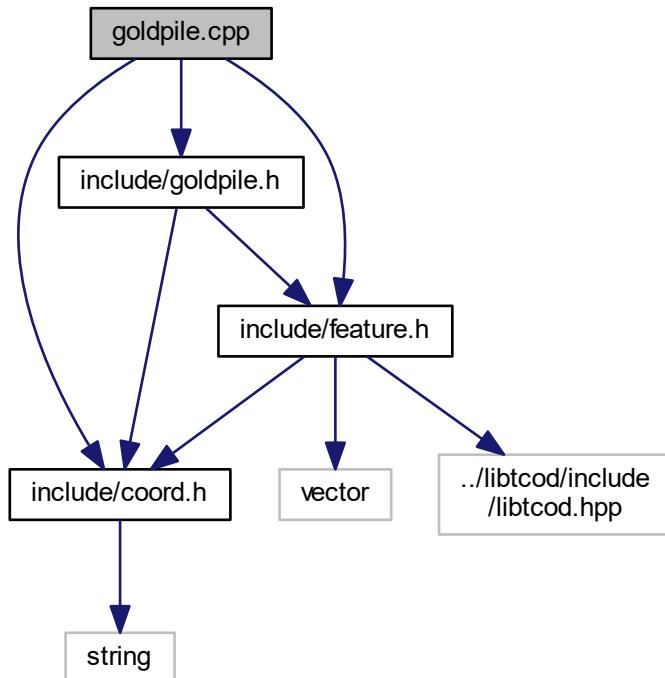
December 06, 2016

## 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

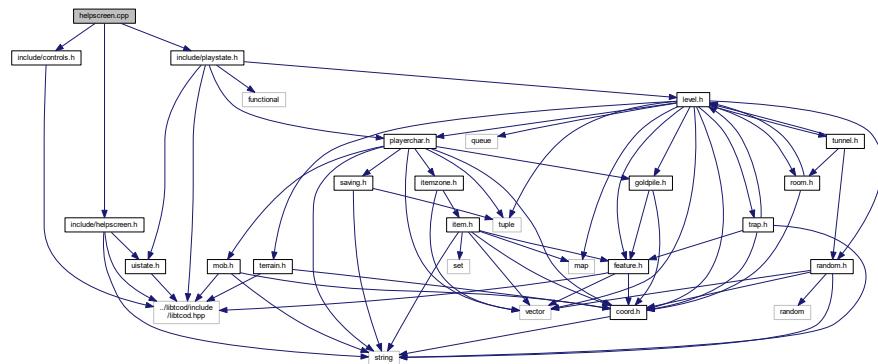
December 06, 2016

## 5.7 helpscreen.cpp File Reference

Member definitions for the [HelpScreen](#) class.

```
#include "include/controls.h"
#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

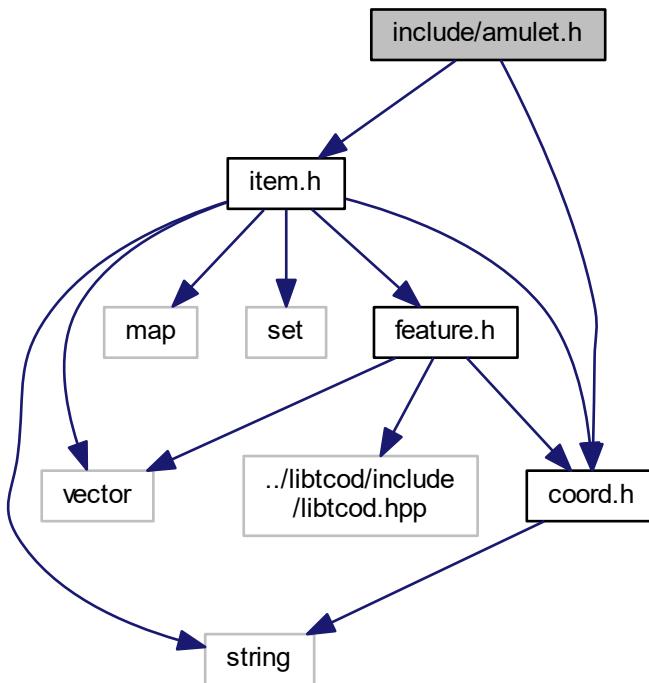
December 06, 2016

## 5.8 include/amulet.h File Reference

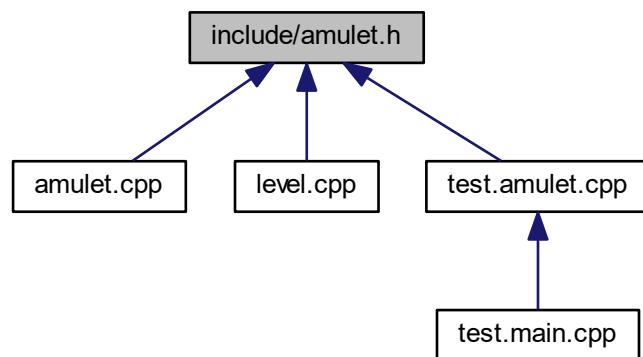
Member declarations for the [Amulet](#) class.

```
#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

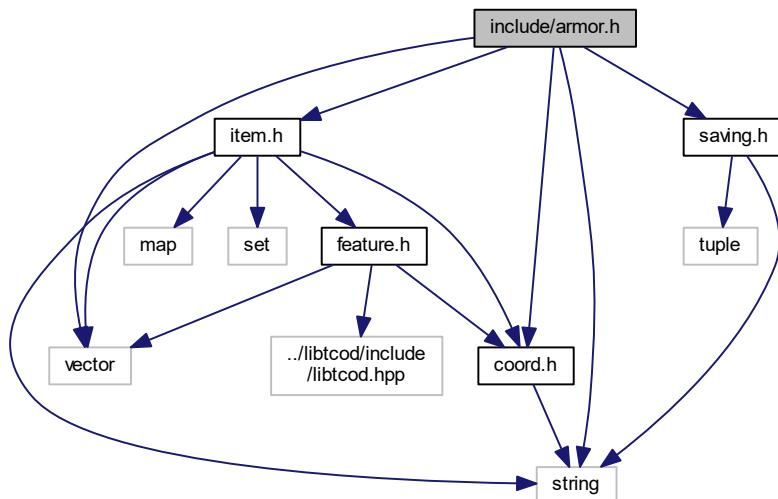
December 06, 2016

## 5.9 include/armor.h File Reference

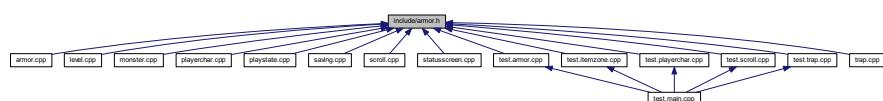
Member declarations for the [Armor](#) class.

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

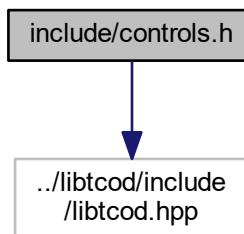
December 06, 2016

### 5.10 include/controls.h File Reference

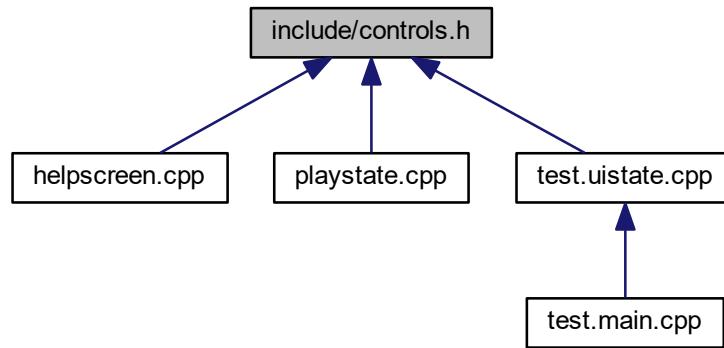
Global members.

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

Include dependency graph for controls.h:



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



## Variables

- const auto **KEYS::HELP** = '?'
- const auto **KEYS::LOG** = 'o'
- const auto **KEYS::INV** = 'i'
- const auto **KEYS::INFO** = 'c'
- const auto **KEYS::QUAFF** = 'q'
- const auto **KEYS::PUT** = 'P'
- const auto **KEYS::STOW** = 'S'
- const auto **KEYS::DROP** = 'd'
- const auto **KEYS::EAT** = 'e'
- const auto **KEYS::TAKEOFF** = 'T'
- const auto **KEYS::WIELD** = 'w'
- const auto **KEYS::WEAR** = 'W'
- const auto **KEYS::REMOVE** = 'R'
- const auto **KEYS::ASCEND** = '<'
- const auto **KEYS::DESCEND** = '>'
- const auto **KEYS::SYMBOLS** = '/'
- const auto **KEYS::READ** = 'r'
- const auto **KEYS::QUIT** = 'Q'
- const auto **KEYS::ZAP** = 'Z'
- const auto **KEYS::TOGGLE\_SAVE** = 'N'
- const auto **KEYS::SEARCH** = 's'
- const auto **KEYS::REST** = '!
- const auto **KEYS::THROW** = 't'

### 5.10.1 Detailed Description

Global members.

Author

Team Rogue++

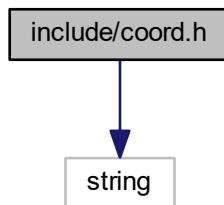
Date

December 06, 2016

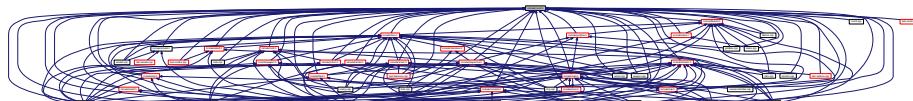
## 5.11 include/coord.h File Reference

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.*

#### 5.11.1 Detailed Description

Member declarations for the [Coord](#) class.

##### Author

Team Rogue++

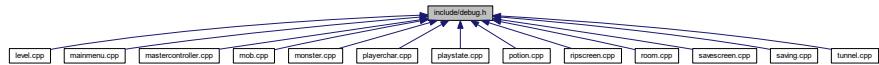
##### Date

December 06, 2016

## 5.12 include/debug.h File Reference

Global members.

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



### 5.12.1 Detailed Description

Global members.

Author

Team Rogue++

Date

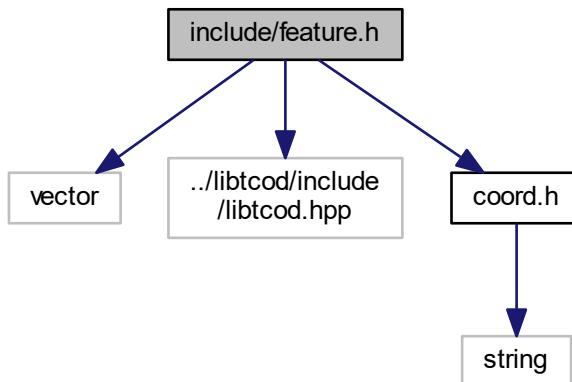
December 06, 2016

## 5.13 include/feature.h File Reference

Member declarations for the [Feature](#) class.

```
#include <vector>
#include "../libtcod/include/libtcod.hpp"
#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.*

### 5.13.1 Detailed Description

Member declarations for the [Feature](#) class.

#### Author

Team Rogue++

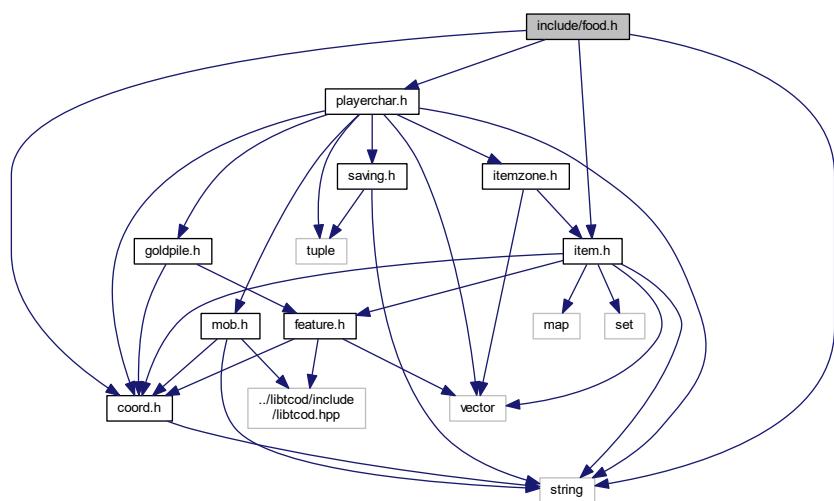
#### Date

December 06, 2016

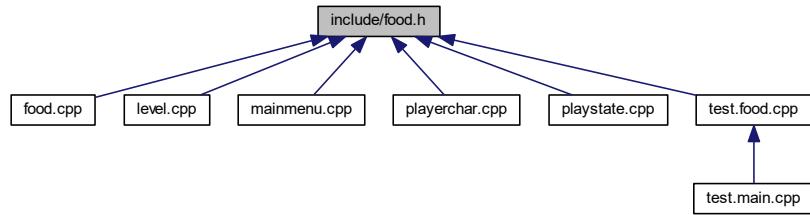
## 5.14 include/food.h File Reference

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.*

### 5.14.1 Detailed Description

Member declarations for the [Food](#) class.

#### Author

Team Rogue++

#### Date

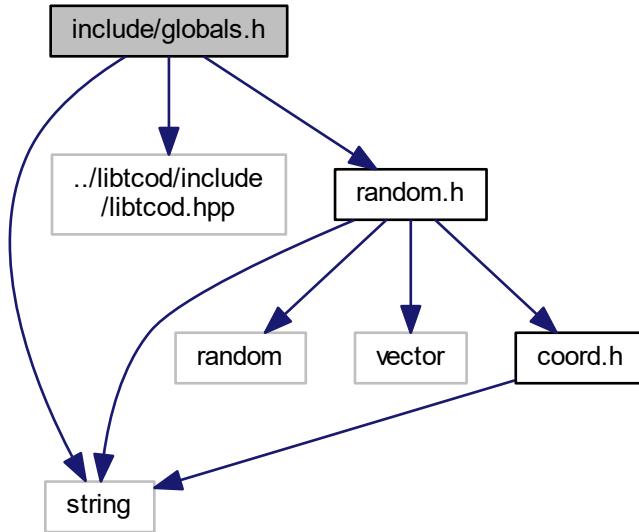
December 06, 2016

## 5.15 include/globals.h File Reference

Global members.

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

```
#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 std::string **HALLUC\_CHARS** = "fPgb!esilMIEFONZnVL%woXHhyCTp\*?UrdxS/KCkmDBvRJ[aGP←Wzqu:AQY=t"
- const int **TURN\_TIME** = 50
- const int **SLOW\_TIME** = 103
- const int **FAST\_TIME** = 23
- constexpr int **SCREEN\_WIDTH** = 80
- constexpr int **SCREEN\_HEIGHT** = 40
- const std::string **VICTORY\_STR** = "Retrieved the Amulet of Yendor"
- const int **DISAPPEAR\_DELAY** = -1

### 5.15.1 Detailed Description

Global members.

#### Author

Team Rogue++

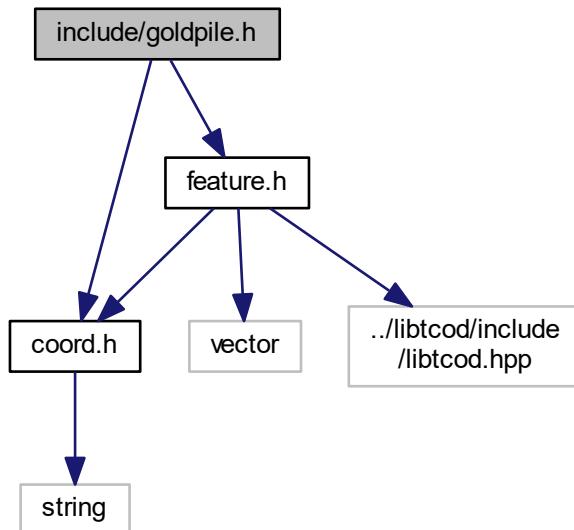
#### Date

December 06, 2016

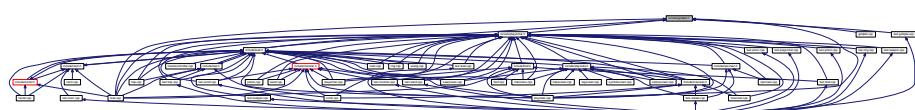
## 5.16 include/goldpile.h File Reference

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.*

### 5.16.1 Detailed Description

Member declarations for the [GoldPile](#) class.

#### Author

Team Rogue++

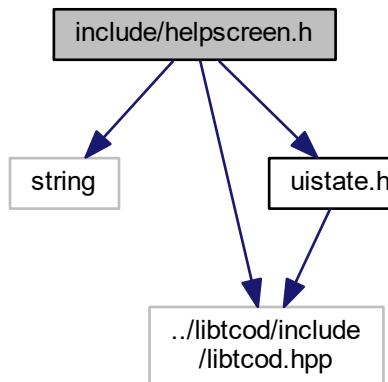
#### Date

December 06, 2016

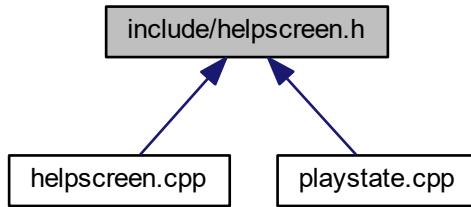
## 5.17 include/helpscreen.h File Reference

Member declarations for the [HelpScreen](#) class.

```
#include <string>
#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.*

### 5.17.1 Detailed Description

Member declarations for the [HelpScreen](#) class.

#### Author

Team Rogue++

#### Date

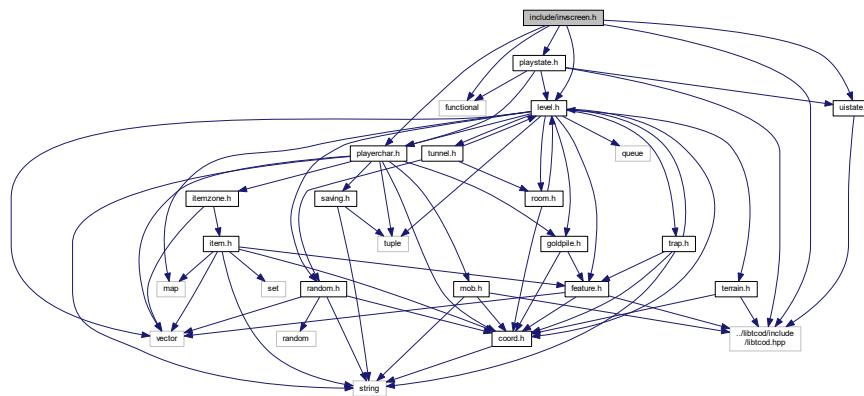
December 06, 2016

## 5.18 include/invscreen.h File Reference

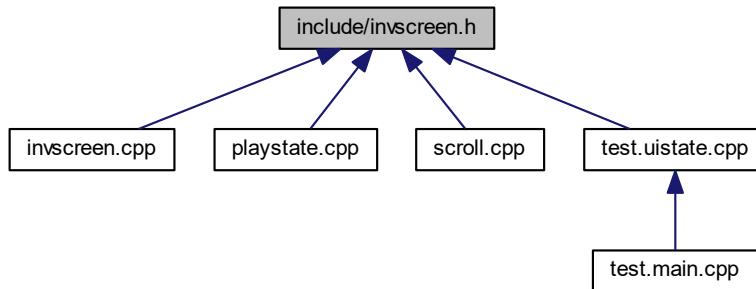
Member declarations for the [InvScreen](#) class.

```
#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.*

### 5.18.1 Detailed Description

Member declarations for the [InvScreen](#) class.

#### Author

Team Rogue++

#### Date

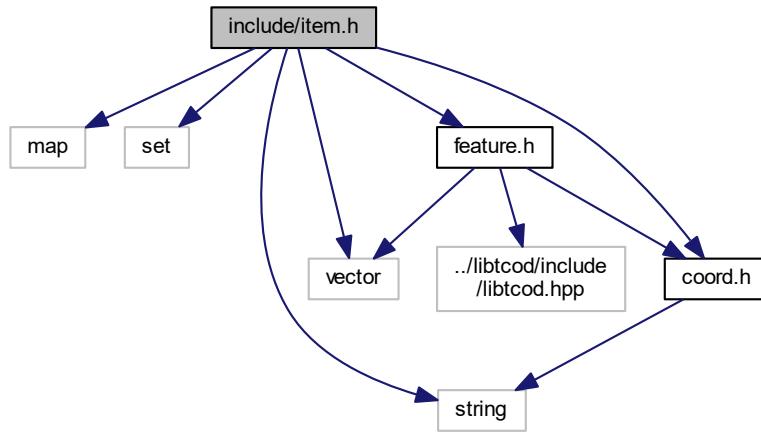
December 06, 2016

## 5.19 include/item.h File Reference

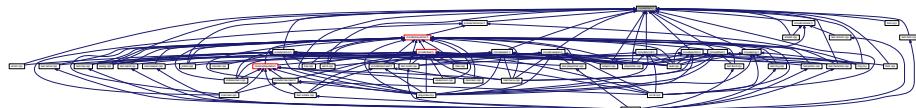
Member declarations for the [Item](#) class.

```
#include <map>
#include <set>
#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.*

#### 5.19.1 Detailed Description

Member declarations for the [Item](#) class.

##### Author

Team Rogue++

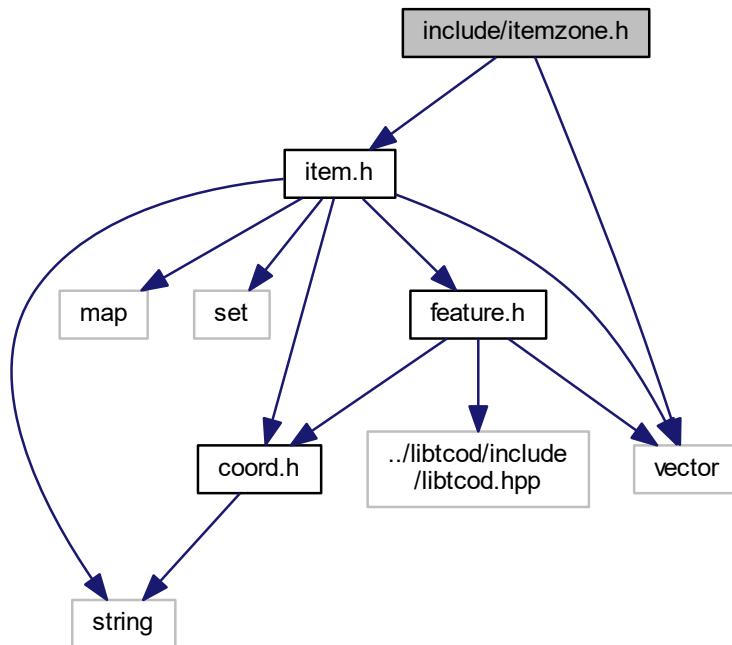
##### Date

December 06, 2016

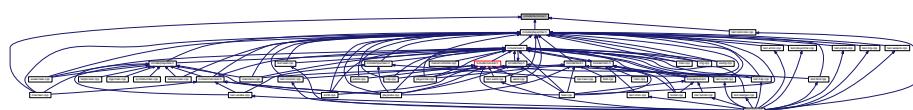
## 5.20 include/itemzone.h File Reference

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.*

#### 5.20.1 Detailed Description

Member declarations for the [ItemZone](#) class.

**Author**

Team Rogue++

**Date**

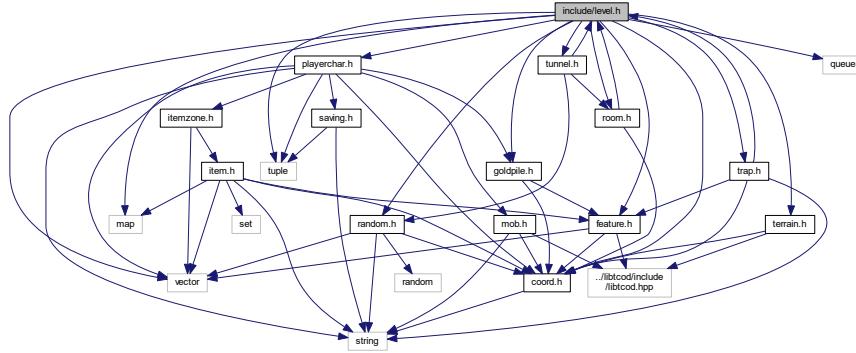
December 06, 2016

## 5.21 include/level.h File Reference

Member declarations for the [Level](#) class.

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

### 5.21.1 Detailed Description

Member declarations for the [Level](#) class.

#### Author

Team Rogue++

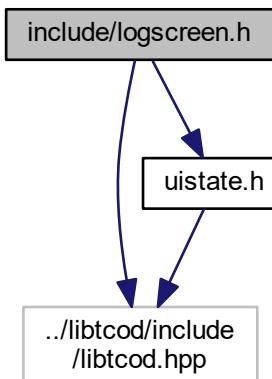
#### Date

December 06, 2016

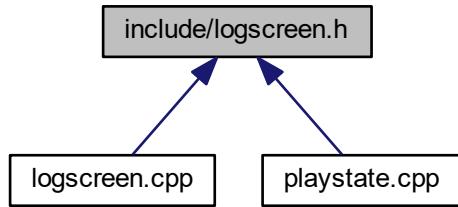
## 5.22 include/logscren.h File Reference

Member declarations for the [LogScreen](#) class.

```
#include "../libtcod/include/libtcod.hpp"
#include "uistate.h"
Include dependency graph for logscren.h:
```



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



## Classes

- class [LogScreen](#)

*Controls the display of the event log.*

### 5.22.1 Detailed Description

Member declarations for the [LogScreen](#) class.

#### Author

Team Rogue++

#### Date

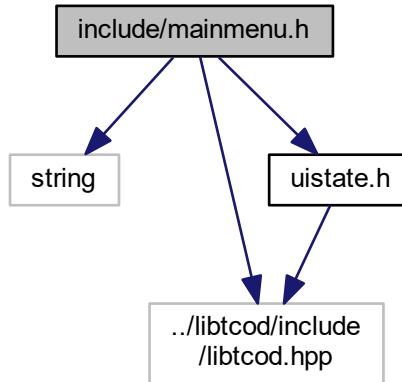
December 06, 2016

## 5.23 include/mainmenu.h File Reference

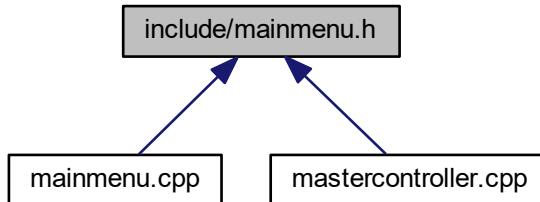
Member declarations for the [MainMenu](#) class.

```
#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.*

### 5.23.1 Detailed Description

Member declarations for the [MainMenu](#) class.

#### Author

Team Rogue++

#### Date

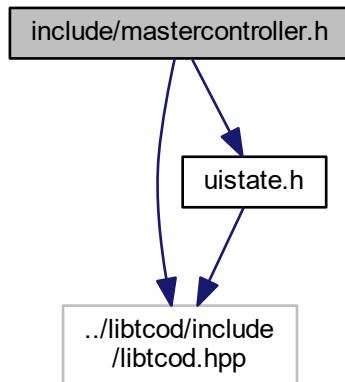
December 06, 2016

## 5.24 include/mastercontroller.h File Reference

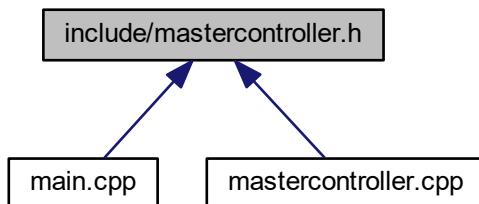
Member declarations for the [MasterController](#) class.

```
#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.*

### 5.24.1 Detailed Description

Member declarations for the [MasterController](#) class.

#### Author

Team Rogue++

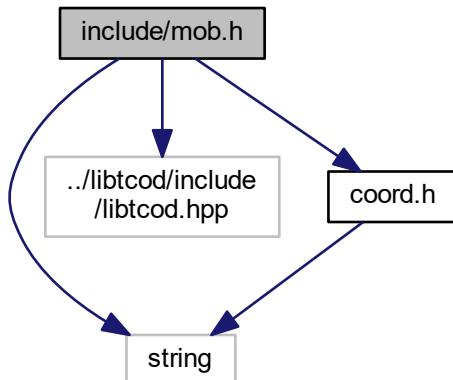
#### Date

December 06, 2016

## 5.25 include/mob.h File Reference

Member declarations for the [Mob](#) class.

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

### 5.25.1 Detailed Description

Member declarations for the [Mob](#) class.

#### Author

Team Rogue++

#### Date

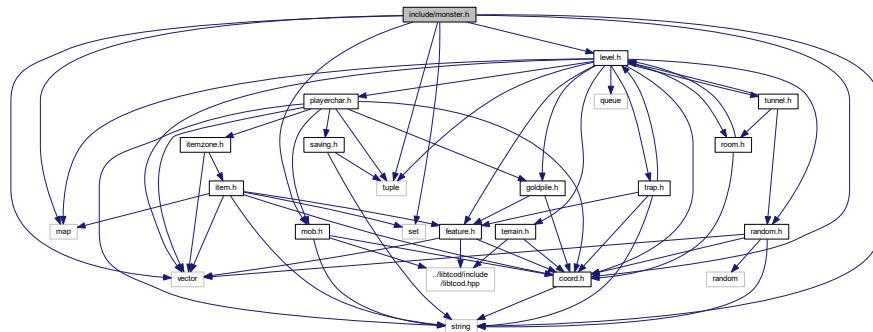
December 06, 2016

## 5.26 include/monster.h File Reference

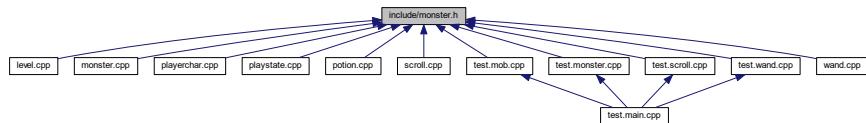
Member declarations for the [Monster](#) class.

```
#include <map>
#include <set>
#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>)*

### 5.26.1 Detailed Description

Member declarations for the [Monster](#) class.

#### Author

Team Rogue++

#### Date

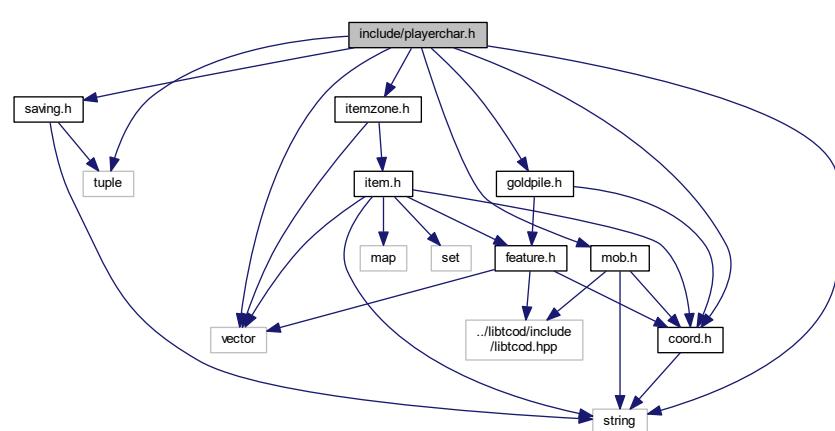
December 06, 2016

## 5.27 include/playerchar.h File Reference

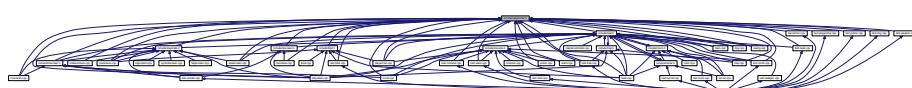
Member declarations for the [PlayerChar](#) class.

```
#include <string>
#include <tuple>
#include <vector>
#include "coord.h"
#include "goldpile.h"
#include "itemzone.h"
#include "mob.h"
#include "saving.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.*

### 5.27.1 Detailed Description

Member declarations for the [PlayerChar](#) class.

#### Author

Team Rogue++

#### Date

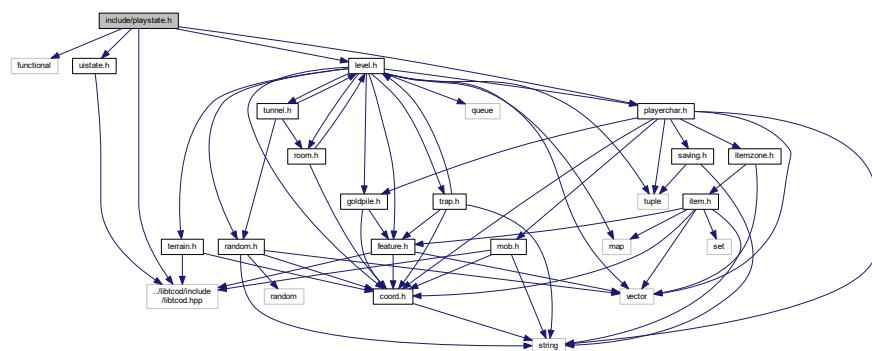
December 06, 2016

## 5.28 include/playstate.h File Reference

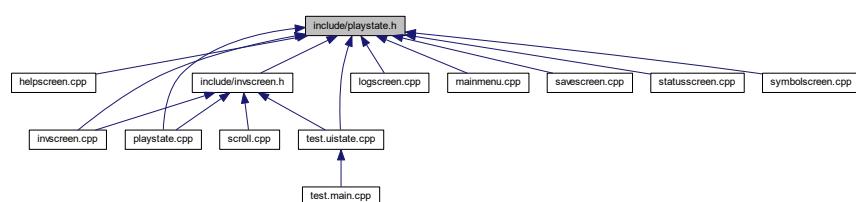
Member declarations for the [PlayState](#) class.

```
#include <functional>
#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.*

### 5.28.1 Detailed Description

Member declarations for the [PlayState](#) class.

#### Author

Team Rogue++

#### Date

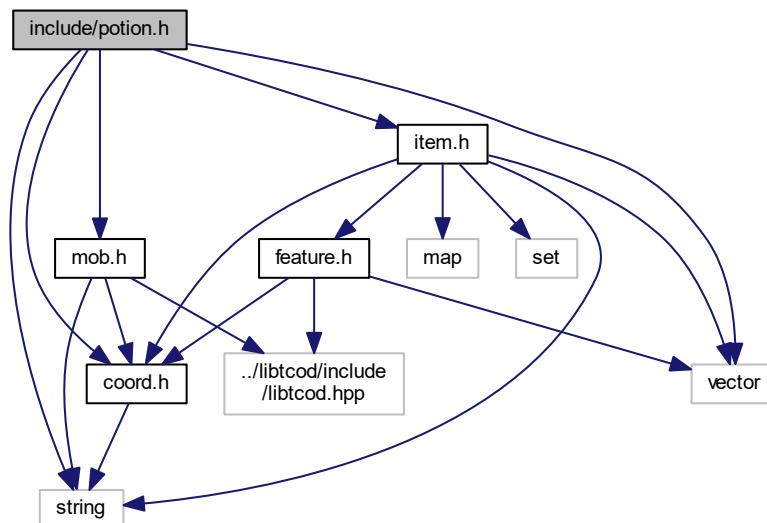
December 06, 2016

## 5.29 include/potion.h File Reference

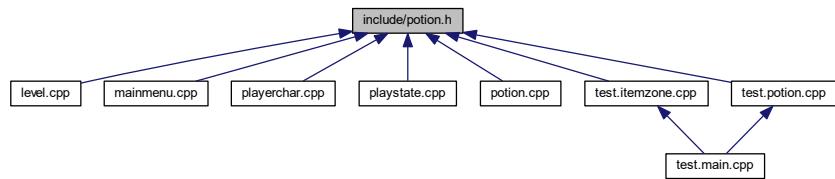
Member declarations for the [Potion](#) class.

```
#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>)*

### 5.29.1 Detailed Description

Member declarations for the [Potion](#) class.

#### Author

Team Rogue++

#### Date

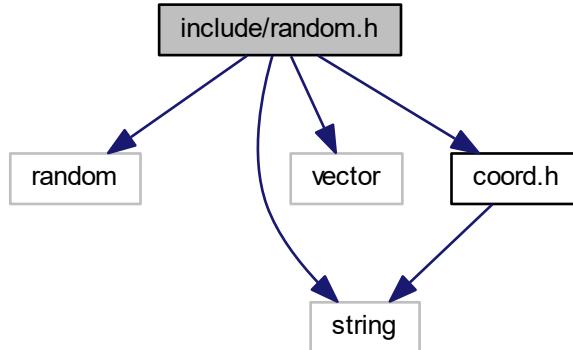
December 06, 2016

## 5.30 include/random.h File Reference

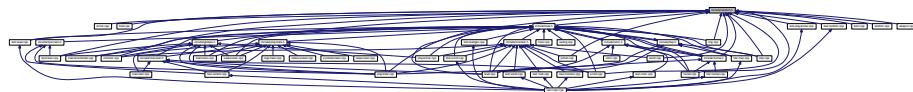
Member declarations for the [Generator](#) class.

```
#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.*

### 5.30.1 Detailed Description

Member declarations for the [Generator](#) class.

#### Author

Team Rogue++

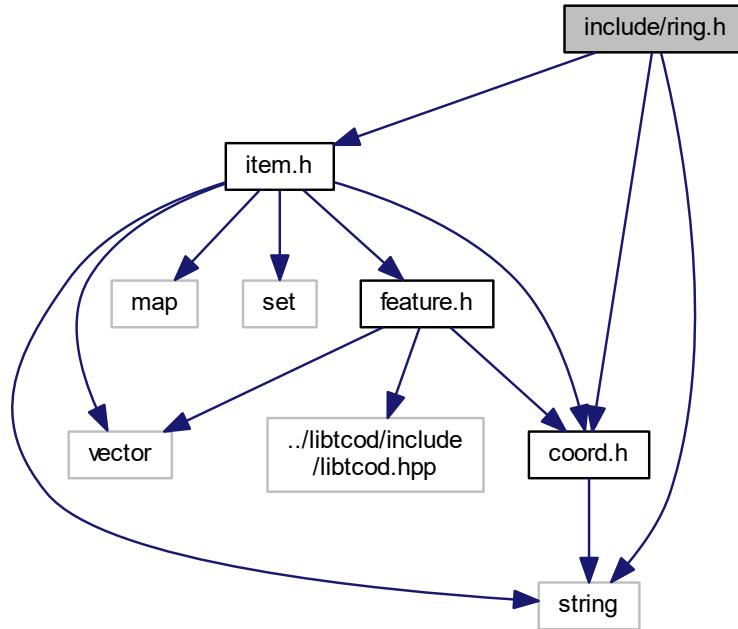
#### Date

December 06, 2016

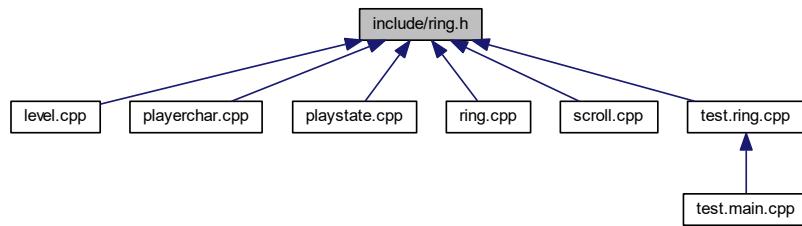
## 5.31 include/ring.h File Reference

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>)*

### 5.31.1 Detailed Description

Member declarations for the `Ring` class.

#### Author

Team Rogue++

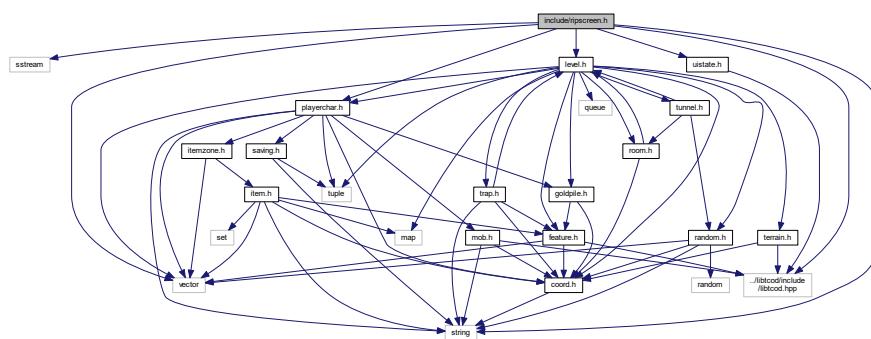
#### Date

December 06, 2016

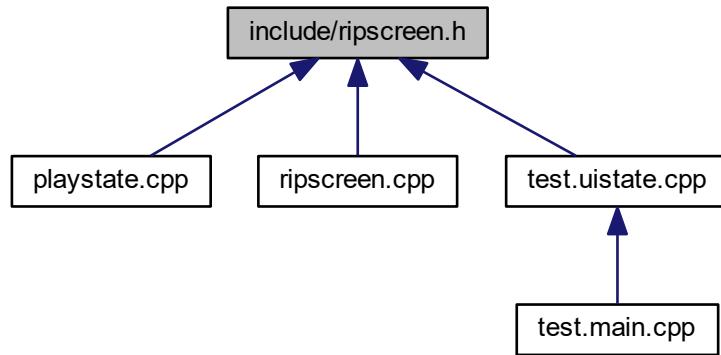
## 5.32 include/ripscreen.h File Reference

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.*

### 5.32.1 Detailed Description

Member declarations for the [RIPScreen](#) class.

#### Author

Team Rogue++

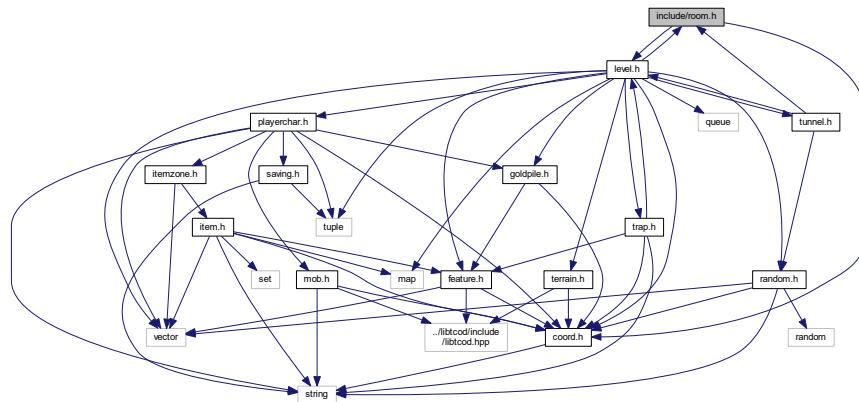
#### Date

December 06, 2016

## 5.33 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.*

### 5.33.1 Detailed Description

Member declarations for the [Room](#) class.

#### Author

Team Rogue++

#### Date

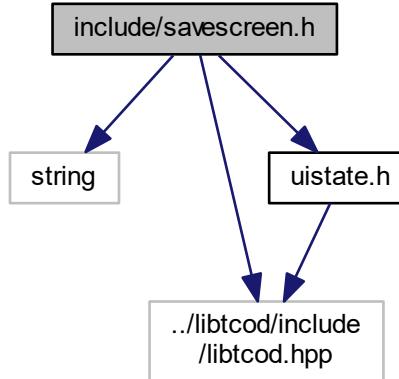
December 06, 2016

## 5.34 include/savescreen.h File Reference

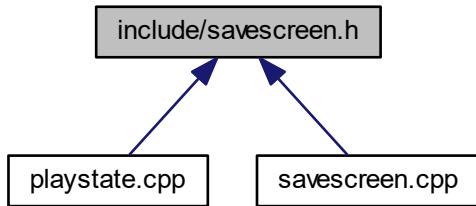
Member declarations for the [SaveScreen](#) class.

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

Include dependency graph for savescreen.h:



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



### Classes

- class [SaveScreen](#)

### 5.34.1 Detailed Description

Member declarations for the [SaveScreen](#) class.

#### Author

Team Rogue++

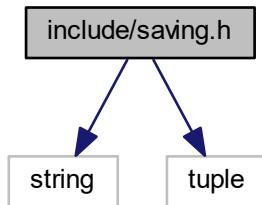
#### Date

December 06, 2016

## 5.35 include/saving.h File Reference

Global members.

```
#include <string>
#include <tuple>
Include dependency graph for saving.h:
```



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



## Functions

- std::string **encode** (PlayerChar \*, Level \*)
- std::tuple< PlayerChar \*, Level \* > **decode** (std::string)

### 5.35.1 Detailed Description

Global members.

#### Author

Team Rogue++

#### Date

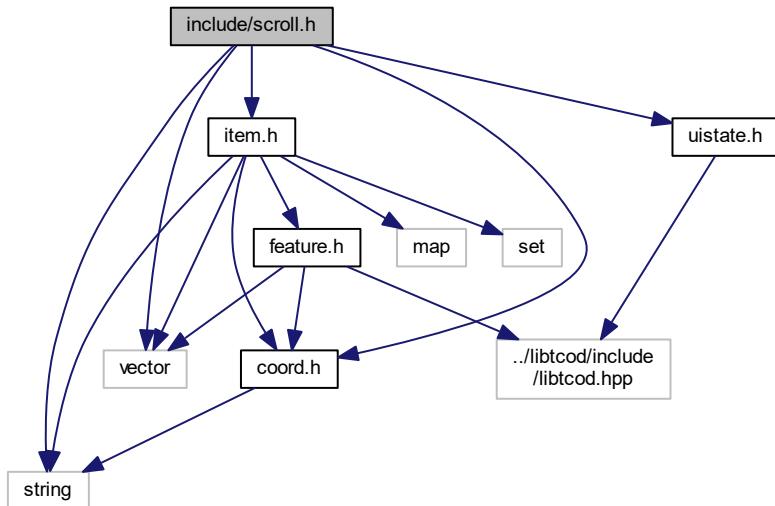
December 06, 2016

## 5.36 include/scroll.h File Reference

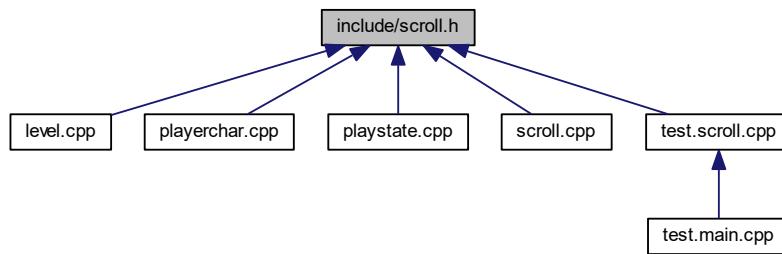
Member declarations for the [Scroll](#) class.

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

### 5.36.1 Detailed Description

Member declarations for the [Scroll](#) class.

#### Author

Team Rogue++

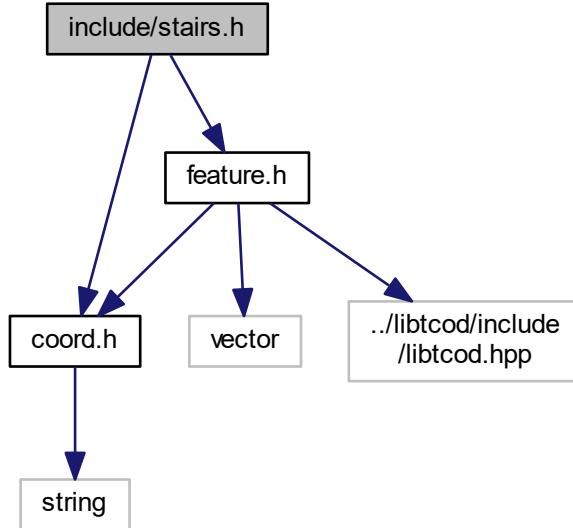
#### Date

December 06, 2016

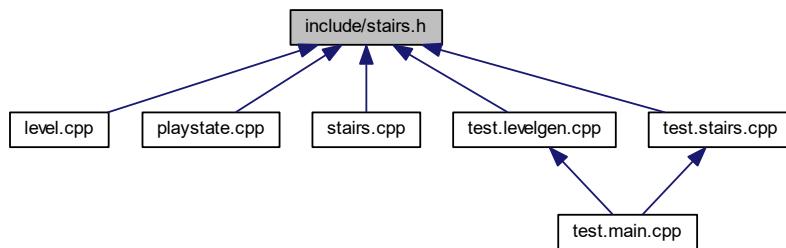
## 5.37 include/stairs.h File Reference

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](#)

### 5.37.1 Detailed Description

Member declarations for the [Stairs](#) class.

**Author**

Team Rogue++

**Date**

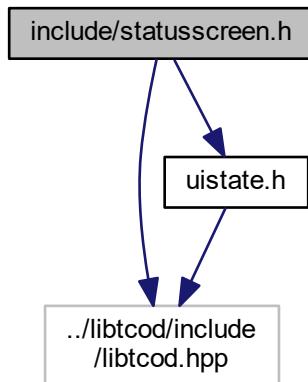
December 06, 2016

### 5.38 include/statusscreen.h File Reference

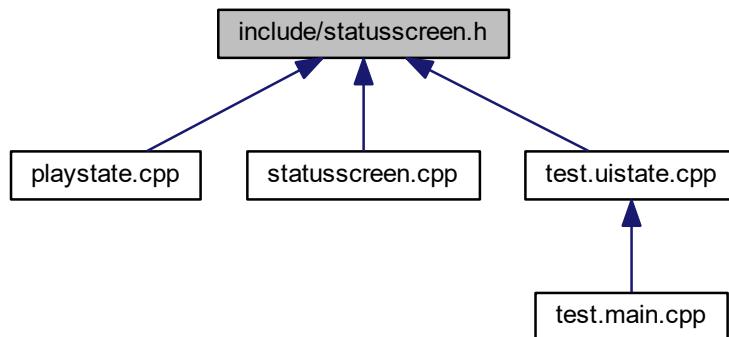
Member declarations for the [StatusScreen](#) class.

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

Include dependency graph for statusscreen.h:



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



## Classes

- class [StatusScreen](#)

*Screen which shows detailed player statistics.*

### 5.38.1 Detailed Description

Member declarations for the [StatusScreen](#) class.

#### Author

Team Rogue++

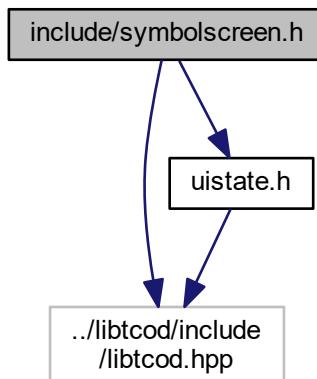
#### Date

December 06, 2016

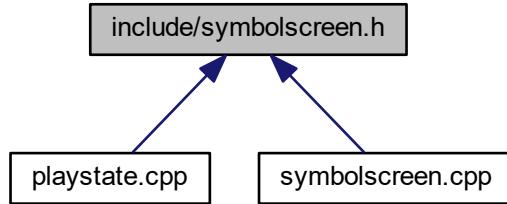
## 5.39 include/symbolscreen.h File Reference

Member declarations for the [SymbolScreen](#) class.

```
#include "../libtcod/include/libtcod.hpp"
#include "uistate.h"
Include dependency graph for symbolscreen.h:
```



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



## Classes

- class [SymbolScreen](#)

### 5.39.1 Detailed Description

Member declarations for the [SymbolScreen](#) class.

#### Author

Team Rogue++

#### Date

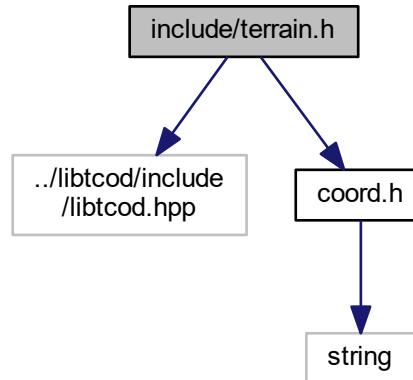
December 06, 2016

## 5.40 include/terrain.h File Reference

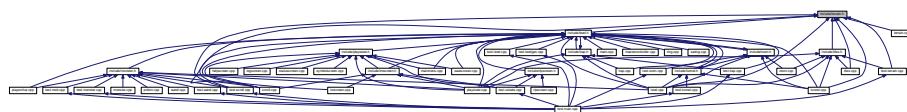
Member declarations for the [Terrain](#) class.

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

### 5.40.1 Detailed Description

Member declarations for the [Terrain](#) class.

#### Author

Team Rogue++

#### Date

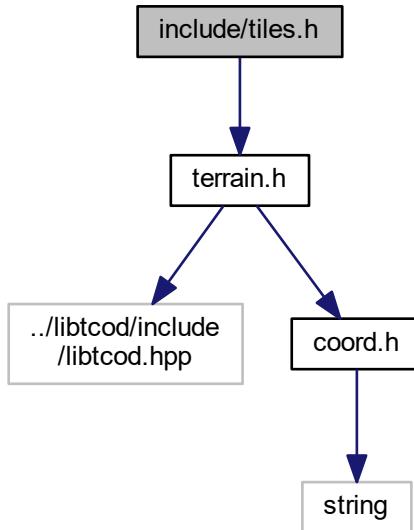
December 06, 2016

## 5.41 include/tiles.h File Reference

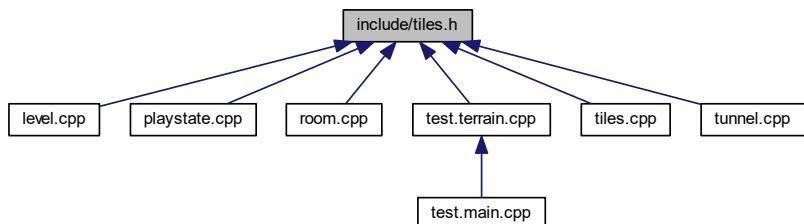
Member declarations for the [Corridor](#), [Door](#), [Floor](#), [Wall](#) classes.

```
#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.*

### 5.41.1 Detailed Description

Member declarations for the [Corridor](#), [Door](#), [Floor](#), [Wall](#) classes.

#### Author

Team Rogue++

#### Date

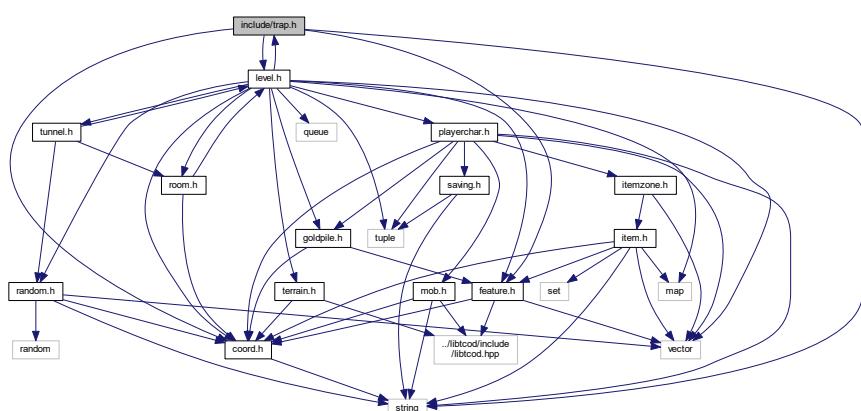
December 06, 2016

## 5.42 include/trap.h File Reference

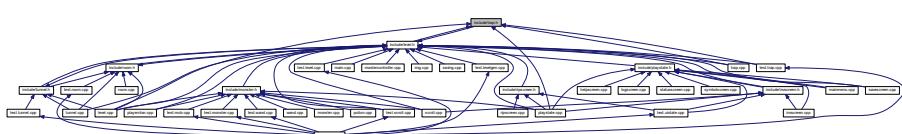
Member declarations for the [Trap](#) class.

```
#include <string>
#include "coord.h"
#include "feature.h"
#include "level.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.*

### 5.42.1 Detailed Description

Member declarations for the [Trap](#) class.

#### Author

Team Rogue++

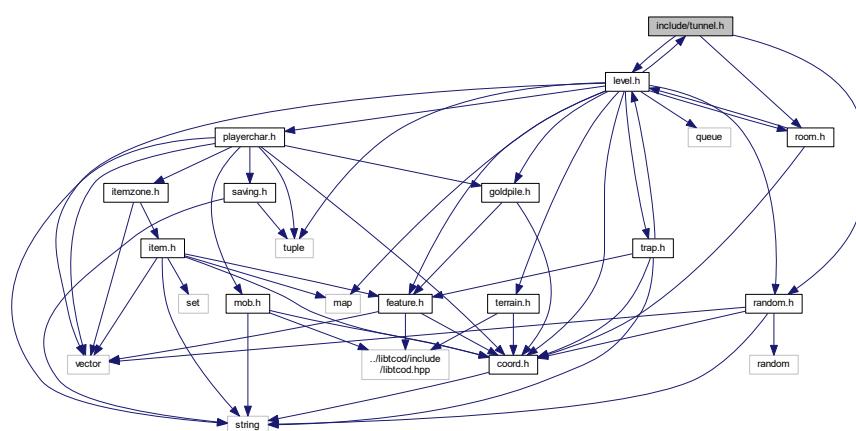
#### Date

December 06, 2016

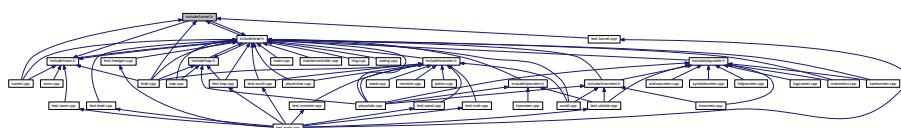
## 5.43 include/tunnel.h File Reference

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.*

### 5.43.1 Detailed Description

Member declarations for the [Tunnel](#) class.

#### Author

Team Rogue++

#### Date

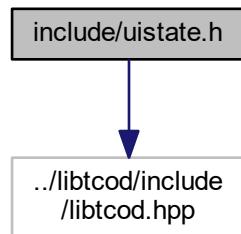
December 06, 2016

## 5.44 include/uistate.h File Reference

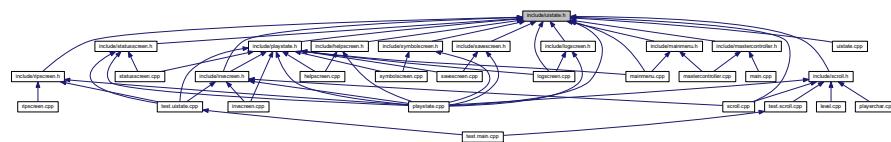
Member declarations for the [UIState](#) class.

```
#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.*

### 5.44.1 Detailed Description

Member declarations for the [UIState](#) class.

#### Author

Team Rogue++

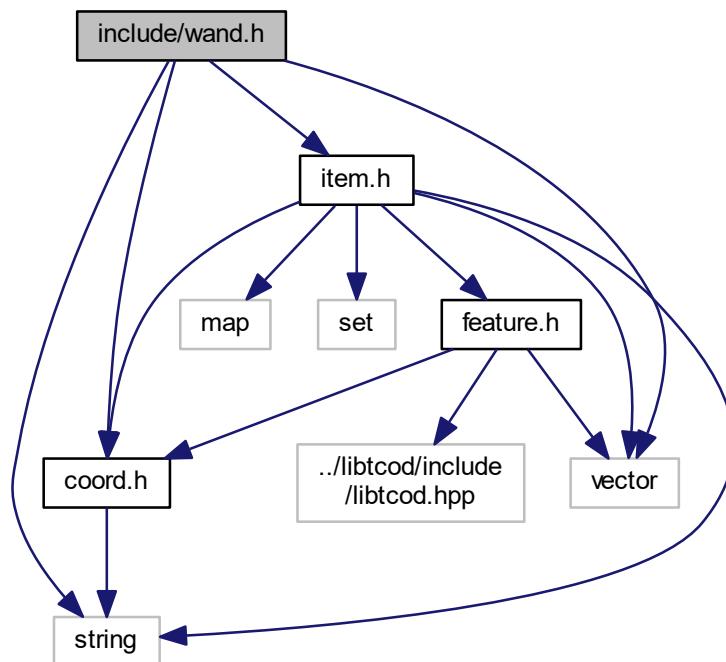
#### Date

December 06, 2016

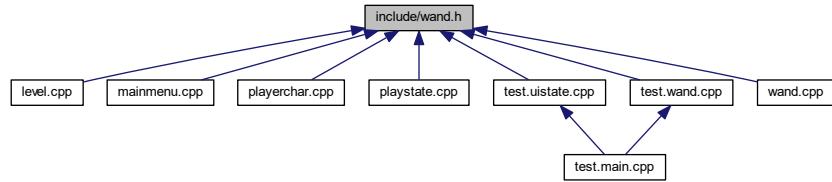
## 5.45 include/wand.h File Reference

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>)*

### 5.45.1 Detailed Description

Member declarations for the [Wand](#) class.

#### Author

Team Rogue++

#### Date

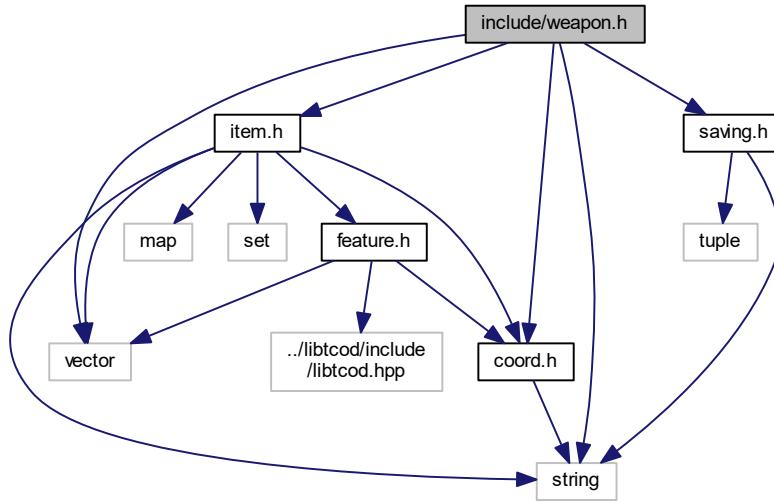
December 06, 2016

## 5.46 include/weapon.h File Reference

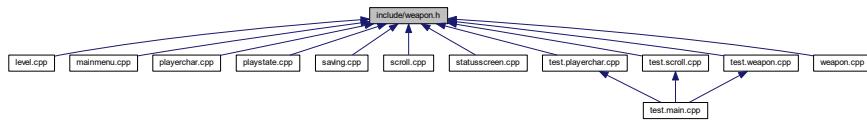
Member declarations for the [Weapon](#) class.

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

```
#include "saving.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>)*

### 5.46.1 Detailed Description

Member declarations for the `Weapon` class.

#### Author

Team Rogue++

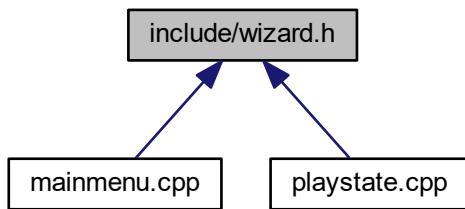
#### Date

December 06, 2016

## 5.47 include/wizard.h File Reference

Global members.

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



### 5.47.1 Detailed Description

Global members.

#### Author

Team Rogue++

#### Date

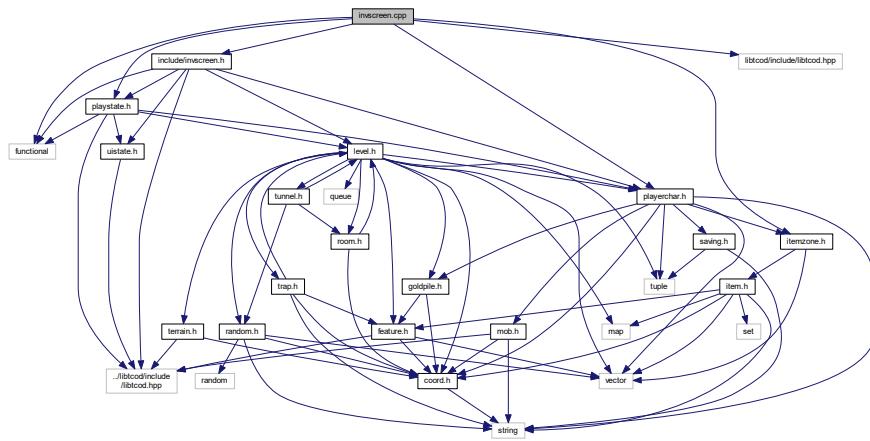
December 06, 2016

## 5.48 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:
```



### 5.48.1 Detailed Description

Member definitions for the [InvScreen](#) class.

#### Author

Team Rogue++

#### Date

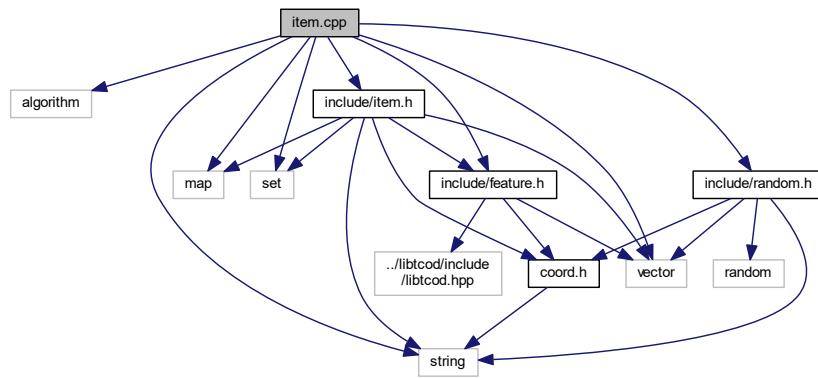
December 06, 2016

## 5.49 item.cpp File Reference

Member definitions for the [Item](#) class.

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

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



#### 5.49.1 Detailed Description

Member definitions for the [Item](#) class.

##### Author

Team Rogue++

##### Date

December 06, 2016

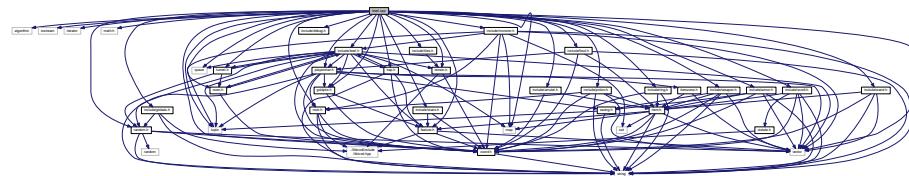
## 5.50 level.cpp File Reference

Member definitions for the [Level](#) class.

```
#include <algorithm>
#include <iostream>
#include <iterator>
#include <map>
#include <math.h>
#include <queue>
#include <tuple>
#include <vector>
#include "include/amulet.h"
#include "include/armor.h"
#include "include/coord.h"
#include "include/debug.h"
#include "include/feature.h"
#include "include/food.h"
#include "include/globals.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/ring.h"
#include "include/room.h"
#include "include	scroll.h"
#include "include/stairs.h"
#include "include/terrain.h"
#include "include/tiles.h"
#include "include/trap.h"
#include "include/tunnel.h"
#include "include/wand.h"
#include "include/weapon.h"
```

Include dependency graph for level.cpp:



### 5.50.1 Detailed Description

Member definitions for the [Level](#) class.

#### Author

Team Rogue++

#### Date

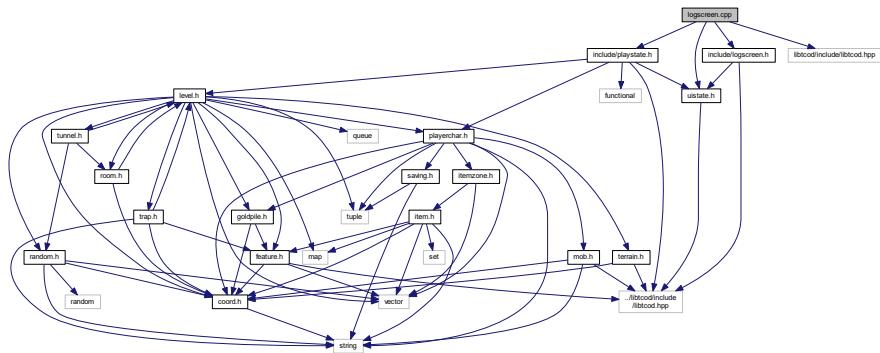
December 06, 2016

## 5.51 logscreen.cpp File Reference

Member definitions for the [LogScreen](#) class.

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

```
#include "libtcod/include/libtcod.hpp"
Include dependency graph for logscreen.cpp:
```



### 5.51.1 Detailed Description

Member definitions for the [LogScreen](#) class.

#### Author

Team Rogue++

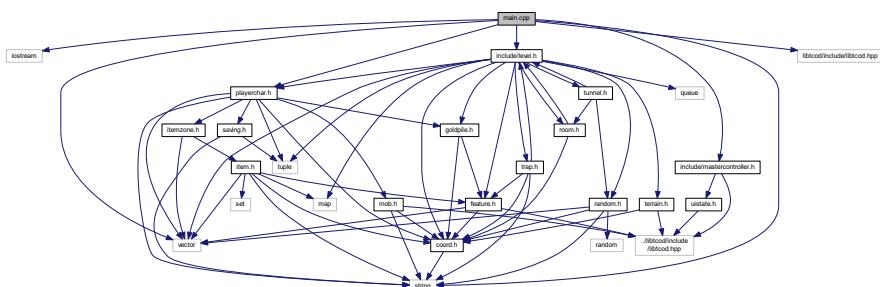
#### Date

December 06, 2016

## 5.52 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 argc, char \*\*args)

*Execution starts here.*

*Execution starts here.*

### **5.52.1 Detailed Description**

## Global members.

## Author

Team Rogue++

Date

December 06, 2016

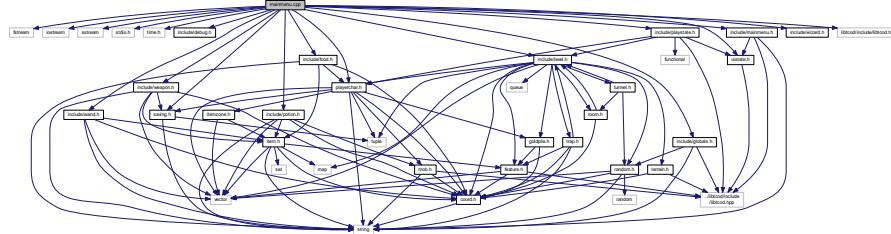
## 5.53 mainmenu.cpp File Reference

Member definitions for the [MainMenu](#) class.

```
#include <fstream>
#include <iostream>
#include <sstream>
#include <stdio.h>
#include <time.h>
#include "include/debug.h"
#include "include/food.h"
#include "include/globals.h"
#include "include/level.h"
#include "include/mainmenu.h"
#include "include/playerchar.h"
#include "include/playstate.h"
#include "include/potion.h"
#include "include/saving.h"
#include "include/uistate.h"
#include "include/wand.h"
#include "include/weapon.h"
#include "include/wizard.h"
#include "libtcod/include/libtcod.hpp"
```

Include dependency graph for mainmenu.cpp:

Include dependency graph for mainmenu.cpp:



### 5.53.1 Detailed Description

Member definitions for the [MainMenu](#) class.

#### Author

Team Rogue++

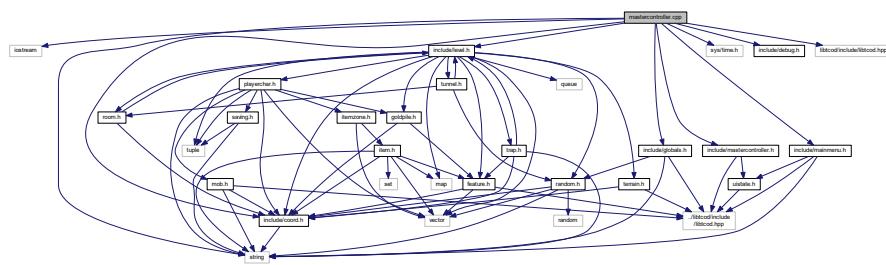
#### Date

December 06, 2016

## 5.54 mastercontroller.cpp File Reference

Member definitions for the [MasterController](#) class.

```
#include <iostream>
#include <string>
#include <sys/time.h>
#include "include/coord.h"
#include "include/debug.h"
#include "include/globals.h"
#include "include/level.h"
#include "include/mainmenu.h"
#include "include/mastercontroller.h"
#include "libtcod/include/libtcod.hpp"
Include dependency graph for mastercontroller.cpp:
```



### 5.54.1 Detailed Description

Member definitions for the [MasterController](#) class.

#### Author

Team Rogue++

#### Date

December 06, 2016

## 5.55 misc/codeformatter.py File Reference

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

### Functions

- def `codeformatter.cleanPragmas` (content)
 

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

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

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

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

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

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

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

*Execution entry point.*

### Variables

- `codeformatter.RE_PATH_IGNORE` = re.compile(r"libtcod|ParseTest|html|misc|assets")
 

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

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

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

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

*C++ source class declaration.*

### 5.55.1 Detailed Description

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

#### Author

Mikhail Andrenkov

### 5.55.2 Function Documentation

#### 5.55.2.1 addHeader()

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

Adds a header to the given C++ file.

**Parameters**

<i>cppFile</i>	The name of the C++ file
<i>content</i>	The content of the C++ file

**Returns**

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

**5.55.2.2 cleanPragmas()**

```
def codeformatter.cleanPragmas (
    content )
```

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

**Parameters**

<i>content</i>	The content of the C++ file
----------------	-----------------------------

**Returns**

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

**5.55.2.3 formatContent()**

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

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

**Parameters**

<i>cppFile</i>	The name of the C++ file
<i>content</i>	The content of the C++ file

**Returns**

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

**5.55.2.4 formatFiles()**

```
def codeformatter.formatFiles (
    cppFiles )
```

Formats all of the given C++ source files.

**Parameters**

<code>cppFiles</code>	The C++ source files
-----------------------	----------------------

**5.55.2.5 sortIncludes()**

```
def codeformatter.sortIncludes (
    content )
```

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

**Parameters**

<code>content</code>	The content of the C++ file
----------------------	-----------------------------

**Returns**

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

**5.55.2.6 trim()**

```
def codeformatter.trim (
    content )
```

Trims the given C++ file.

**Parameters**

<code>content</code>	The content of the C++ file
----------------------	-----------------------------

**Returns**

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

## 5.56 misc/stringfinder.py File Reference

Accumulates all strings throughout the C++ files.

### Functions

- def **stringfinder.findStrings** (cppFiles)
- def **stringfinder.findFiles** ()  
*Recursively finds all C++ source files.*
- def **stringfinder.main** ()  
*Execution entry point.*

## Variables

- `stringfinder.RE_PATH_IGNORE` = `re.compile(r"libtcod|ParseTest|html|misc|assets")`  
*Ignored paths.*
- `stringfinder.RE_EXTENSION` = `re.compile(r"\.(cpp|h)")`  
*C++ file extensions.*
- `stringfinder.RE_HEADER_EXTENSION` = `re.compile(r"\.h$")`  
*C++ header file.*
- string `stringfinder.OUTPUT_FILE` = `"./Strings.txt"`

### 5.56.1 Detailed Description

Accumulates all strings throughout the C++ files.

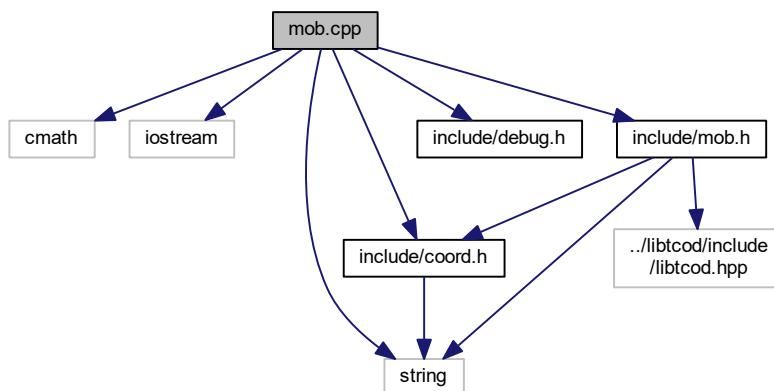
#### Author

Mikhail Andrenkov

## 5.57 mob.cpp File Reference

Member definitions for the `Mob` class.

```
#include <cmath>
#include <iostream>
#include <string>
#include "include/coord.h"
#include "include/debug.h"
#include "include/mob.h"
Include dependency graph for mob.cpp:
```



### 5.57.1 Detailed Description

Member definitions for the [Mob](#) class.

#### Author

Team Rogue++

#### Date

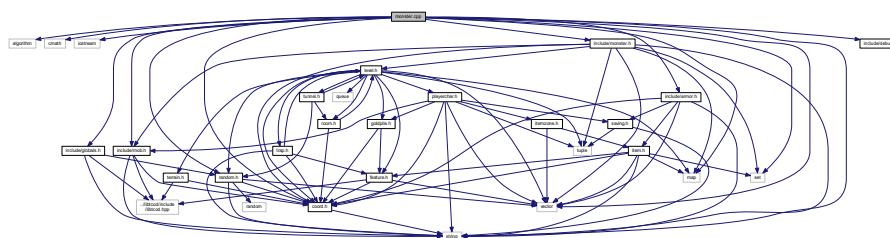
December 06, 2016

## 5.58 monster.cpp File Reference

Member definitions for the [Monster](#) class.

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

Include dependency graph for monster.cpp:



### 5.58.1 Detailed Description

Member definitions for the [Monster](#) class.

#### Author

Team Rogue++

#### Date

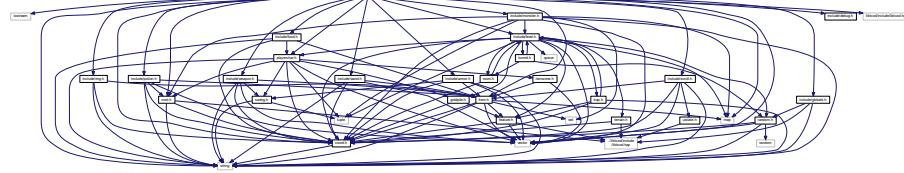
December 06, 2016

## 5.59 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/debug.h"
#include "include/food.h"
#include "include/globals.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 "libtcod/include/libtcod.hpp"
```

Include dependency graph for playerchar.cpp:



### 5.59.1 Detailed Description

Member definitions for the [PlayerChar](#) class.

#### Author

Team Rogue++

#### Date

December 06, 2016

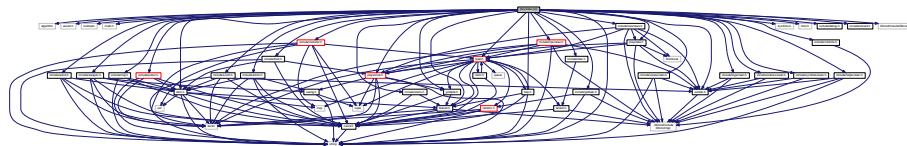
## 5.60 playstate.cpp File Reference

Member definitions for the [PlayState](#) class.

```
#include <algorithm>
#include <assert.h>
```

```
#include <iostream>
#include <math.h>
#include <string>
#include <sys/time.h>
#include <time.h>
#include "include/armor.h"
#include "include/controls.h"
#include "include/debug.h"
#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/monster.h"
#include "include/playerchar.h"
#include "include/playstate.h"
#include "include/potion.h"
#include "include/ring.h"
#include "include/ripscreen.h"
#include "include/savescreen.h"
#include "include/scroll.h"
#include "include/stairs.h"
#include "include/statusscreen.h"
#include "include/symbolscreen.h"
#include "include/tiles.h"
#include "include/trap.h"
#include "include/uistate.h"
#include "include/wand.h"
#include "include/weapon.h"
#include "include/wizard.h"
#include "libtcod/include/libtcod.hpp"
```

Include dependency graph for playstate.cpp:



## Classes

- class [QuitPrompt2](#)
- class [RingRemovePrompt](#)
- class [QuickZap](#)
- class [QuickThrow](#)
- class [QuickUse< T >](#)
- class [DirectionPrompt](#)

### 5.60.1 Detailed Description

Member definitions for the [PlayState](#) class.

**Author**

Team Rogue++

**Date**

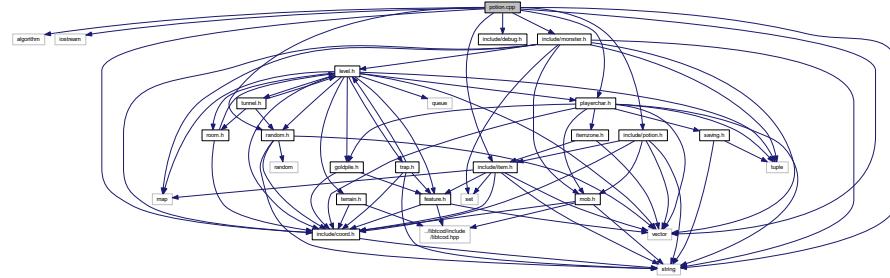
December 06, 2016

## 5.61 potion.cpp File Reference

Member definitions for the [Potion](#) class.

```
#include <algorithm>
#include <iostream>
#include <string>
#include <vector>
#include "include/coord.h"
#include "include/debug.h"
#include "include/item.h"
#include "include/monster.h"
#include "include/playerchar.h"
#include "include/potion.h"
#include "include/random.h"
```

Include dependency graph for potion.cpp:



### 5.61.1 Detailed Description

Member definitions for the [Potion](#) class.

**Author**

Team Rogue++

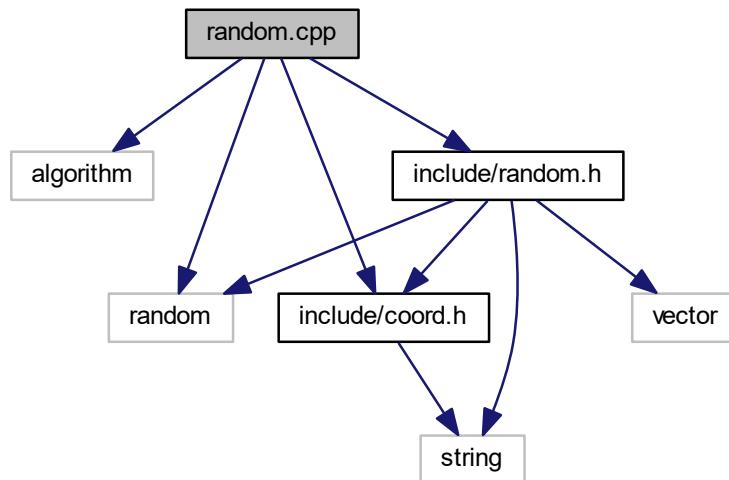
**Date**

December 06, 2016

## 5.62 random.cpp File Reference

Global members.

```
#include <algorithm>
#include <random>
#include "include/coord.h"
#include "include/random.h"
Include dependency graph for random.cpp:
```



### 5.62.1 Detailed Description

Global members.

#### Author

Team Rogue++

#### Date

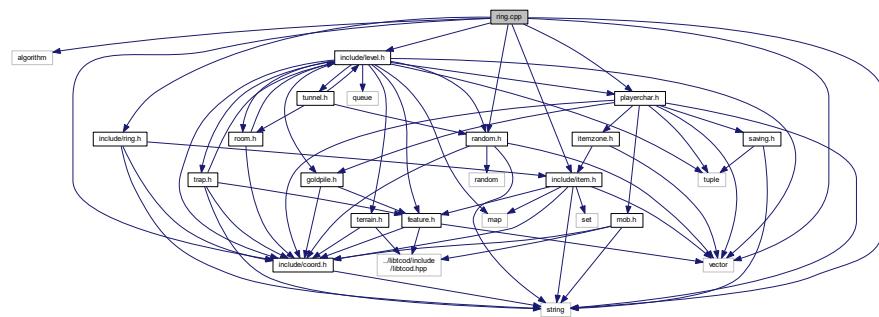
December 06, 2016

## 5.63 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/level.h"
#include "include/playerchar.h"
#include "include/random.h"
#include "include/ring.h"
Include dependency graph for ring.cpp:
```



### **5.63.1 Detailed Description**

Member definitions for the [Ring](#) class.

## Author

Team Rogue++

Date

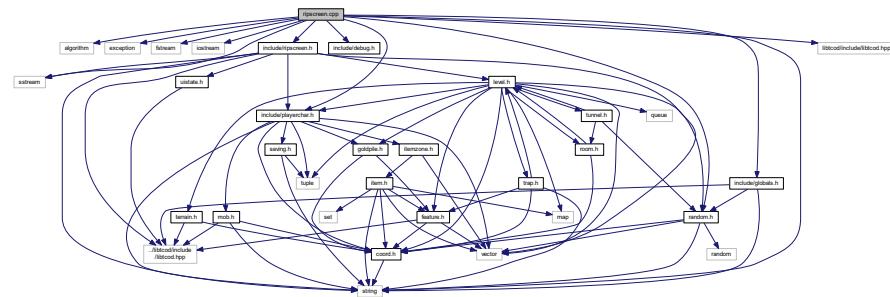
December 06, 2016

## 5.64 ripscreen.cpp File Reference

Member definitions for the [RIPScreen](#) class.

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

```
#include "libtcod/include/libtcod.hpp"
Include dependency graph for ripscreen.cpp:
```



## Classes

- struct [ScoreItem](#)

### 5.64.1 Detailed Description

Member definitions for the [RIPScreen](#) class.

#### Author

Team Rogue++

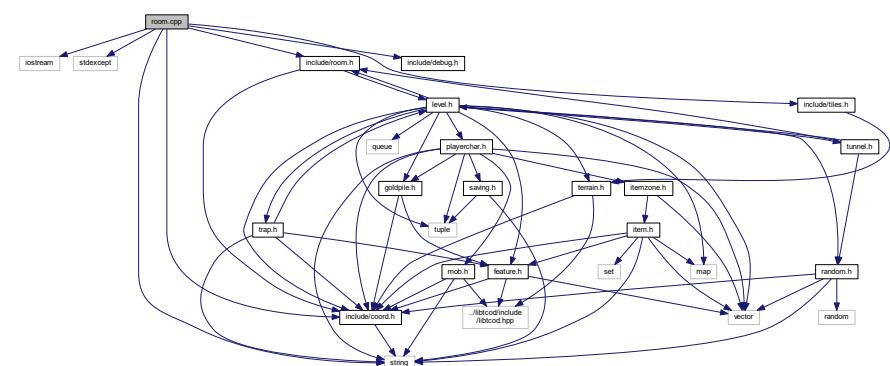
#### Date

December 06, 2016

## 5.65 room.cpp File Reference

Member definitions for the [Room](#) class.

```
#include <iostream>
#include <stdexcept>
#include <string>
#include "include/coord.h"
#include "include/debug.h"
#include "include/room.h"
#include "include/tiles.h"
Include dependency graph for room.cpp:
```



### 5.65.1 Detailed Description

Member definitions for the [Room](#) class.

#### Author

Team Rogue++

#### Date

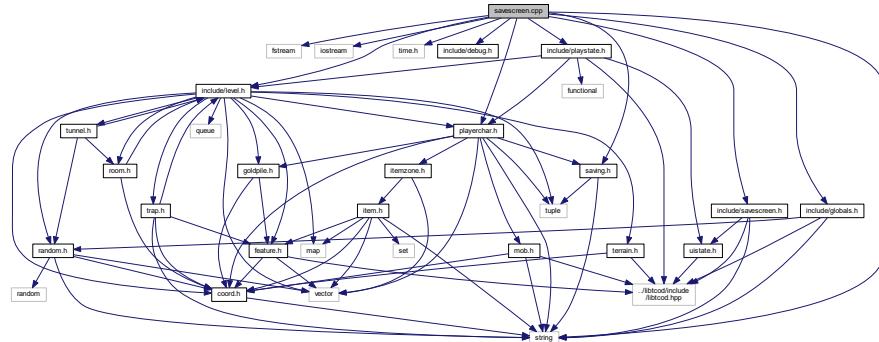
December 06, 2016

## 5.66 savescreen.cpp File Reference

Member definitions for the [SaveScreen](#) class.

```
#include <fstream>
#include <iostream>
#include <string>
#include <time.h>
#include "include/debug.h"
#include "include/globals.h"
#include "include/level.h"
#include "include/playerchar.h"
#include "include/playstate.h"
#include "include/savescreen.h"
#include "include/saving.h"
```

Include dependency graph for savescreen.cpp:



### 5.66.1 Detailed Description

Member definitions for the [SaveScreen](#) class.

#### Author

Team Rogue++

#### Date

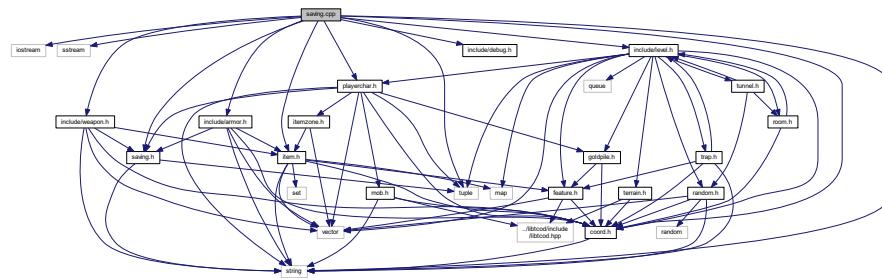
December 06, 2016

## 5.67 saving.cpp File Reference

Global members.

```
#include <iostream>
#include <sstream>
#include <string>
#include <tuple>
#include "include/armor.h"
#include "include/coord.h"
#include "include/debug.h"
#include "include/item.h"
#include "include/level.h"
#include "include/playerchar.h"
#include "include/saving.h"
#include "include/weapon.h"
```

Include dependency graph for saving.cpp:



## Functions

- string **encode** (PlayerChar \*player, Level \*level)
- tuple< PlayerChar \*, Level \* > **decode** (string encoding)

### 5.67.1 Detailed Description

Global members.

#### Author

Team Rogue++

#### Date

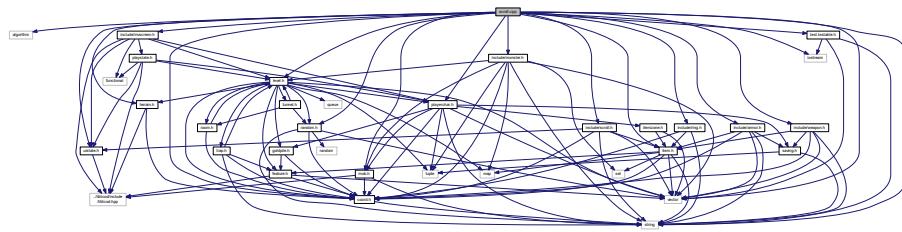
December 06, 2016

## 5.68 scroll.cpp File Reference

Member definitions for the [Scroll](#) class.

```
#include <algorithm>
#include <iostream>
#include <string>
#include <vector>
#include "include/armor.h"
#include "include/coord.h"
#include "include/invscreen.h"
#include "include/item.h"
#include "include/level.h"
#include "include/mob.h"
#include "include/monster.h"
#include "include/playerchar.h"
#include "include/random.h"
#include "include/ring.h"
#include "include/scroll.h"
#include "include/terrain.h"
#include "include/uistate.h"
#include "include/weapon.h"
#include "test/testable.h"
```

Include dependency graph for scroll.cpp:



### 5.68.1 Detailed Description

Member definitions for the [Scroll](#) class.

#### Author

Team Rogue++

#### Date

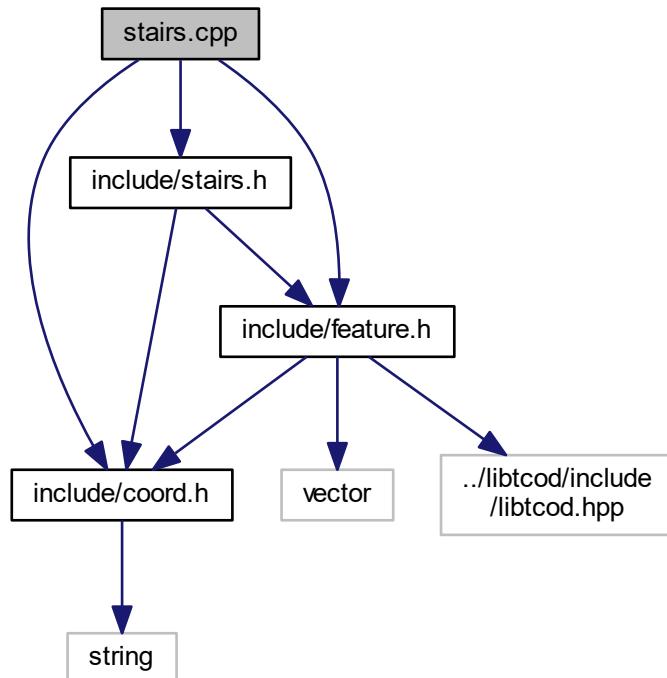
December 06, 2016

## 5.69 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:
```



### 5.69.1 Detailed Description

Member definitions for the [Stairs](#) class.

#### Author

Team Rogue++

#### Date

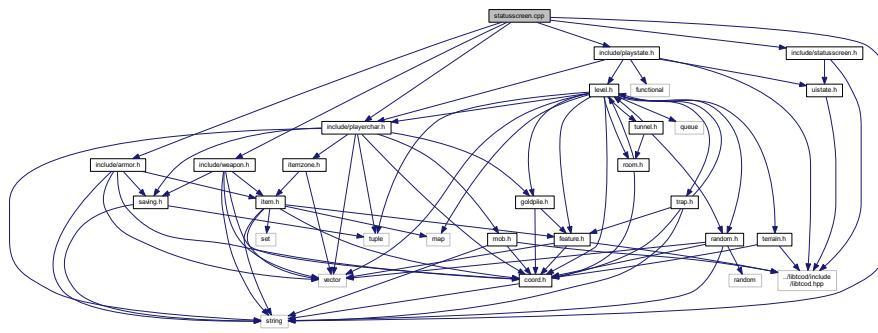
December 06, 2016

## 5.70 statusscreen.cpp File Reference

Member definitions for the [StatusScreen](#) class.

```
#include <string>
#include "include/armor.h"
#include "include/playerchar.h"
#include "include/playstate.h"
```

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



### 5.70.1 Detailed Description

Member definitions for the [StatusScreen](#) class.

#### Author

Team Rogue++

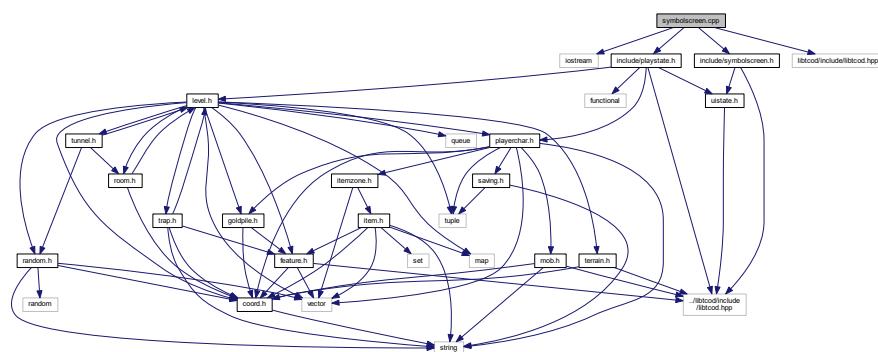
#### Date

December 06, 2016

## 5.71 symbolscreen.cpp File Reference

Member definitions for the [SymbolScreen](#) class.

```
#include <iostream>
#include "include/playstate.h"
#include "include/symbolscreen.h"
#include "libtcod/include/libtcod.hpp"
Include dependency graph for symbolscreen.cpp:
```



### 5.71.1 Detailed Description

Member definitions for the [SymbolScreen](#) class.

Author

Team Rogue++

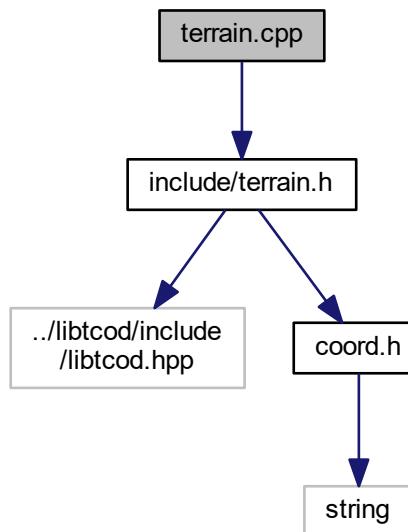
Date

December 06, 2016

## 5.72 terrain.cpp File Reference

Member definitions for the [Terrain](#) class.

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



### 5.72.1 Detailed Description

Member definitions for the [Terrain](#) class.

Author

Team Rogue++

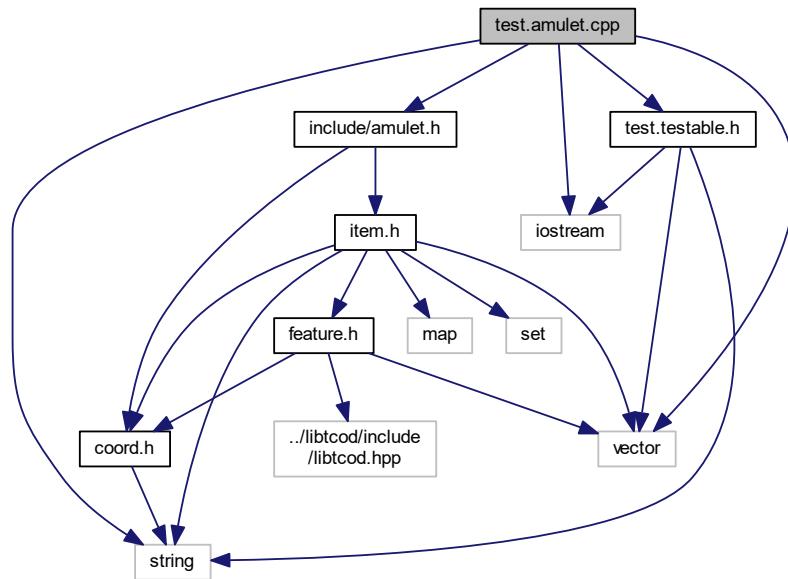
Date

December 06, 2016

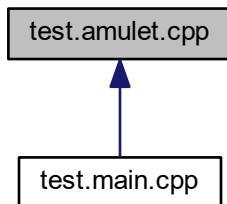
## 5.73 test.amulet.cpp File Reference

Global members.

```
#include <iostream>
#include <string>
#include <vector>
#include "include/amulet.h"
#include "test.testable.h"
Include dependency graph for test.amulet.cpp:
```



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



## Classes

- class [AmuletTest](#)

### 5.73.1 Detailed Description

Global members.

Author

Team Rogue++

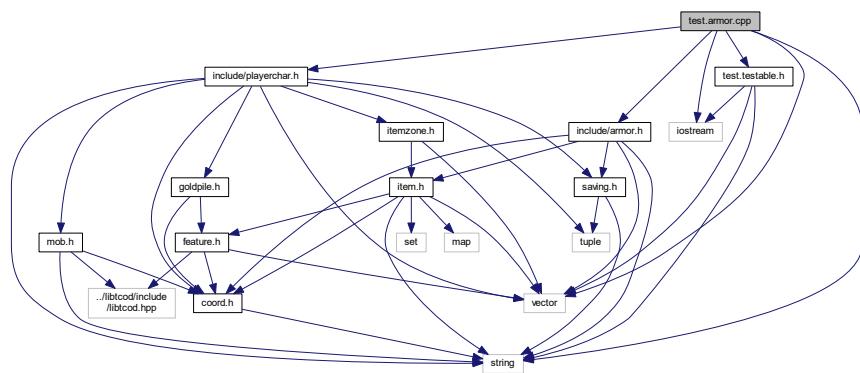
Date

December 06, 2016

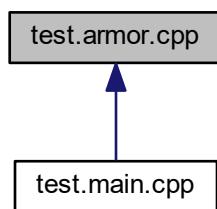
## 5.74 test.armor.cpp File Reference

Global members.

```
#include <iostream>
#include <string>
#include <vector>
#include "include/armor.h"
#include "include/playerchar.h"
#include "test.testable.h"
Include dependency graph for test.armor.cpp:
```



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



## Classes

- class [ArmorTest](#)

### 5.74.1 Detailed Description

Global members.

#### Author

Team Rogue++

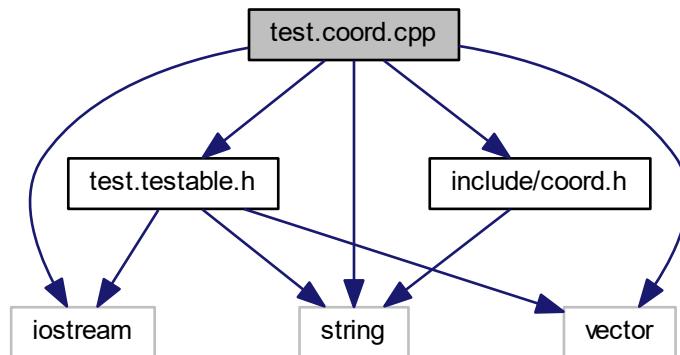
#### Date

December 06, 2016

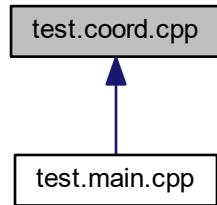
## 5.75 test.coord.cpp File Reference

Global members.

```
#include <iostream>
#include <string>
#include <vector>
#include "include/coord.h"
#include "test.testable.h"
Include dependency graph for test.coord.cpp:
```



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



## Classes

- class [CoordTest](#)

### 5.75.1 Detailed Description

Global members.

#### Author

Team Rogue++

#### Date

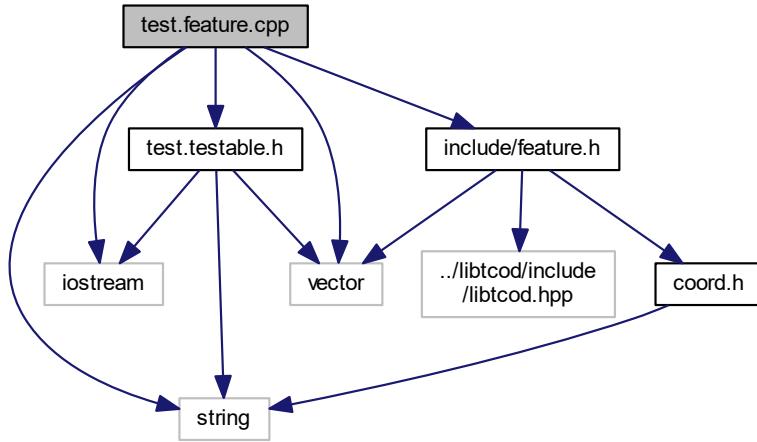
December 06, 2016

## 5.76 test.feature.cpp File Reference

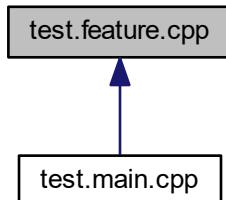
Global members.

```
#include <iostream>
#include <string>
#include <vector>
#include "include/feature.h"
```

```
#include "test.testable.h"
Include dependency graph for test.feature.cpp:
```



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



## Classes

- class [FeatureTest](#)

### 5.76.1 Detailed Description

Global members.

#### Author

Team Rogue++

#### Date

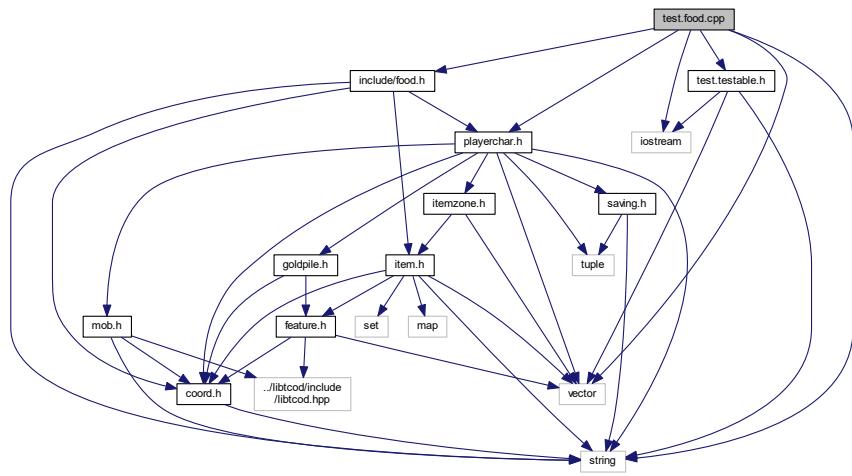
December 06, 2016

## 5.77 test.food.cpp File Reference

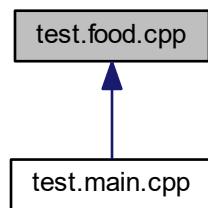
Global members.

```
#include <iostream>
#include <string>
#include <vector>
#include "include/food.h"
#include "include/playerchar.h"
#include "test.testable.h"

Include dependency graph for test.food.cpp:
```



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



### Classes

- class [FoodTest](#)

### 5.77.1 Detailed Description

Global members.

#### Author

Team Rogue++

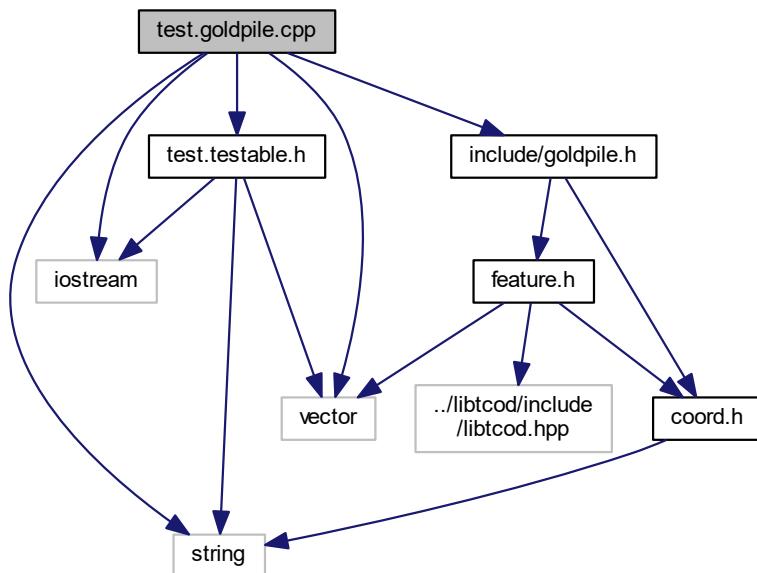
#### Date

December 06, 2016

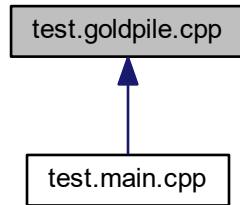
## 5.78 test.goldpile.cpp File Reference

Global members.

```
#include <iostream>
#include <string>
#include <vector>
#include "include/goldpile.h"
#include "test.testable.h"
Include dependency graph for test.goldpile.cpp:
```



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



## Classes

- class [GoldPileTest](#)

### 5.78.1 Detailed Description

Global members.

#### Author

Team Rogue++

#### Date

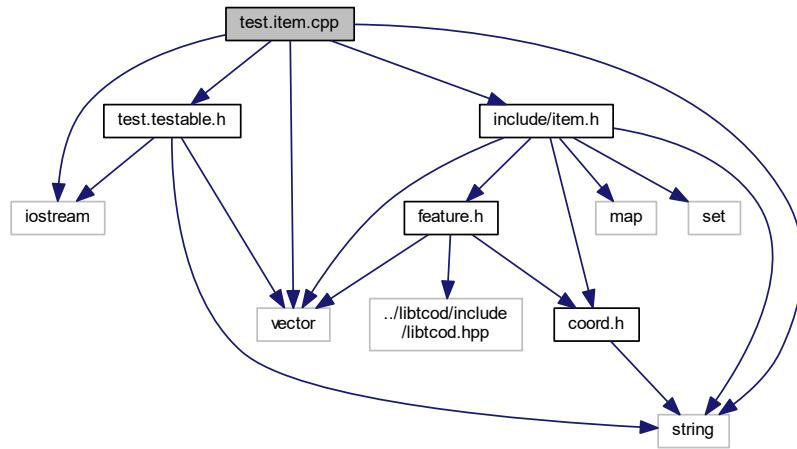
December 06, 2016

## 5.79 test.item.cpp File Reference

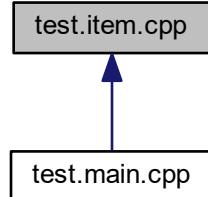
Global members.

```
#include <iostream>
#include <string>
#include <vector>
#include "include/item.h"
```

```
#include "test.testable.h"
Include dependency graph for test.item.cpp:
```



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



## Classes

- class [ItemTest](#)

### 5.79.1 Detailed Description

Global members.

#### Author

Team Rogue++

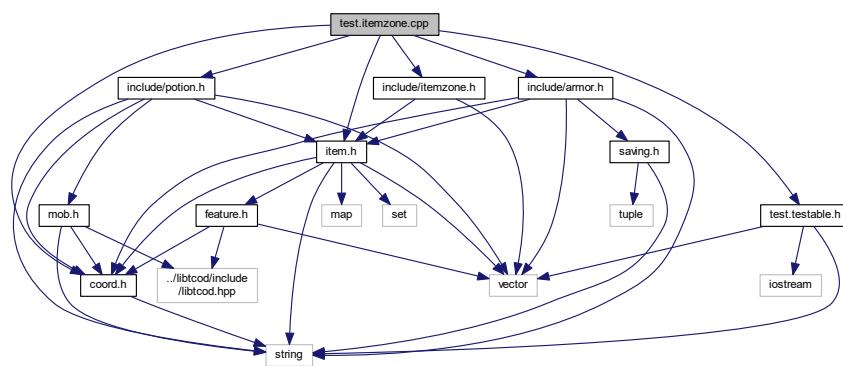
#### Date

December 06, 2016

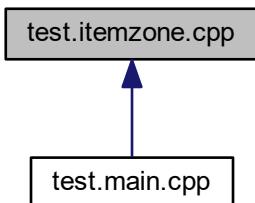
## 5.80 test.itemzone.cpp File Reference

Global members.

```
#include "include/armor.h"
#include "include/coord.h"
#include "include/item.h"
#include "include/itemzone.h"
#include "include/potion.h"
#include "test.testable.h"
Include dependency graph for test.itemzone.cpp:
```



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



### Classes

- class [ItemZoneTest](#)

#### 5.80.1 Detailed Description

Global members.

**Author**

Team Rogue++

**Date**

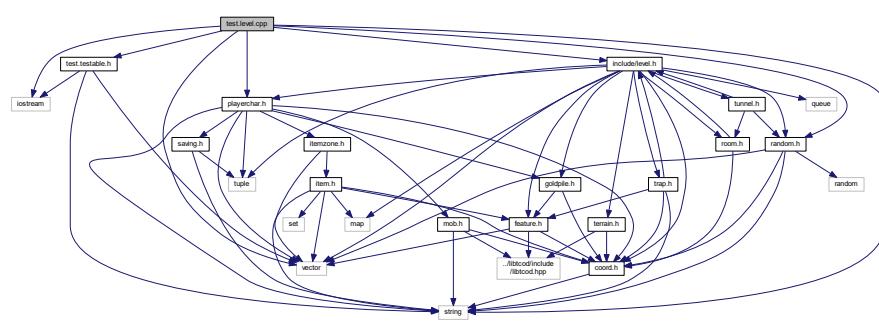
December 06, 2016

## 5.81 test.level.cpp File Reference

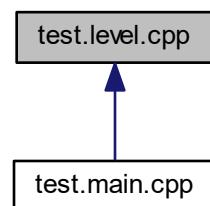
Global members.

```
#include <iostream>
#include <string>
#include <vector>
#include "include/level.h"
#include "include/playerchar.h"
#include "include/random.h"
#include "test.testable.h"
```

Include dependency graph for test.level.cpp:



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



## Classes

- class [LevelTest](#)

### 5.81.1 Detailed Description

Global members.

#### Author

Team Rogue++

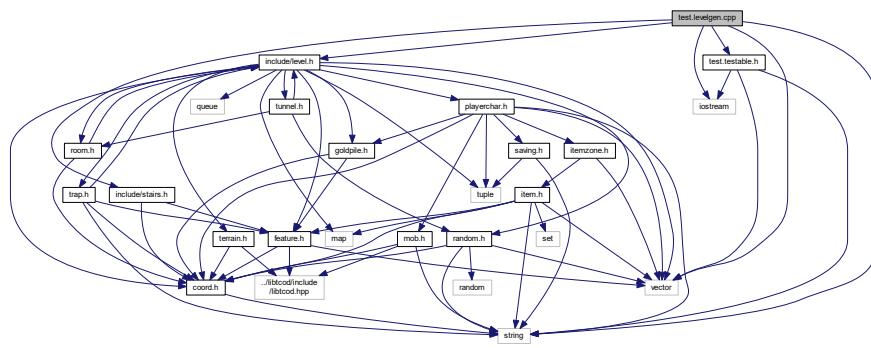
#### Date

December 06, 2016

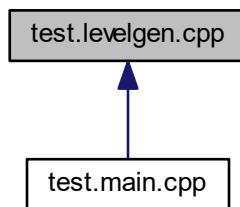
## 5.82 test.levelgen.cpp File Reference

Global members.

```
#include <iostream>
#include <string>
#include <vector>
#include "include/level.h"
#include "include/stairs.h"
#include "test/testable.h"
Include dependency graph for test.levelgen.cpp:
```



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



## Classes

- class [LevelGenTest](#)

### 5.82.1 Detailed Description

Global members.

#### Author

Team Rogue++

#### Date

December 06, 2016

## 5.83 test.main.cpp File Reference

Global members.

```
#include <iostream>
#include <string>
#include <vector>
#include "test.amulet.cpp"
#include "test.armor.cpp"
#include "test.coord.cpp"
#include "test.feature.cpp"
#include "test.food.cpp"
#include "test.goldpile.cpp"
#include "test.item.cpp"
#include "test.itemzone.cpp"
#include "test.level.cpp"
#include "test.levelgen.cpp"
#include "test.mob.cpp"
#include "test.monster.cpp"
#include "test.playerchar.cpp"
#include "test.potion.cpp"
#include "test.random.cpp"
#include "test.ring.cpp"
#include "test.room.cpp"
#include "test.scroll.cpp"
#include "test.stairs.cpp"
#include "test.terrain.cpp"
#include "test.testable.h"
#include "test.trap.cpp"
#include "test.tunnel.cpp"
#include "test.uistate.cpp"
#include "test.wand.cpp"
#include "test.weapon.cpp"
```

Include dependency graph for test.main.cpp:



## Functions

- int **main ()**

### 5.83.1 Detailed Description

Global members.

#### Author

Team Rogue++

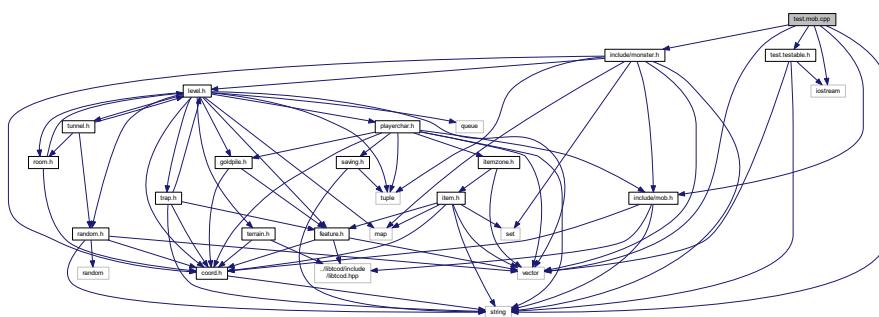
#### Date

December 06, 2016

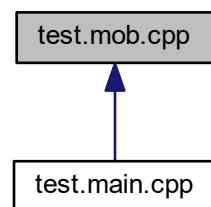
## 5.84 test.mob.cpp File Reference

Global members.

```
#include <iostream>
#include <string>
#include <vector>
#include "include/mob.h"
#include "include/monster.h"
#include "test/testable.h"
Include dependency graph for test.mob.cpp:
```



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



## Classes

- class [MobTest](#)

### 5.84.1 Detailed Description

Global members.

#### Author

Team Rogue++

#### Date

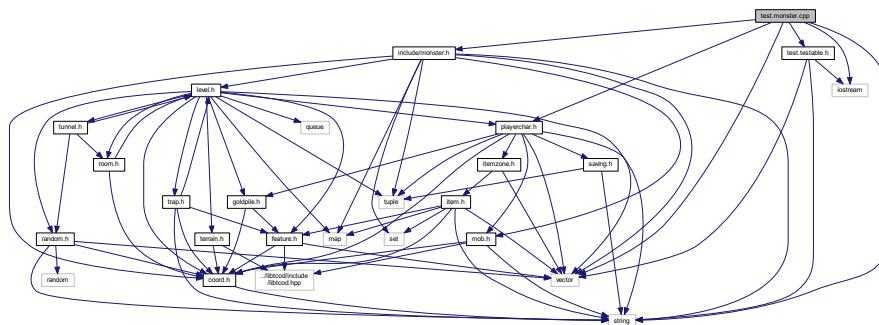
December 06, 2016

## 5.85 test.monster.cpp File Reference

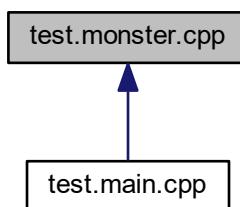
Global members.

```
#include <iostream>
#include <string>
#include <vector>
#include "include/monster.h"
#include "include/playerchar.h"
#include "test.testable.h"
```

Include dependency graph for test.monster.cpp:



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



## Classes

- class [MonsterTest](#)

### 5.85.1 Detailed Description

Global members.

Author

Team Rogue++

Date

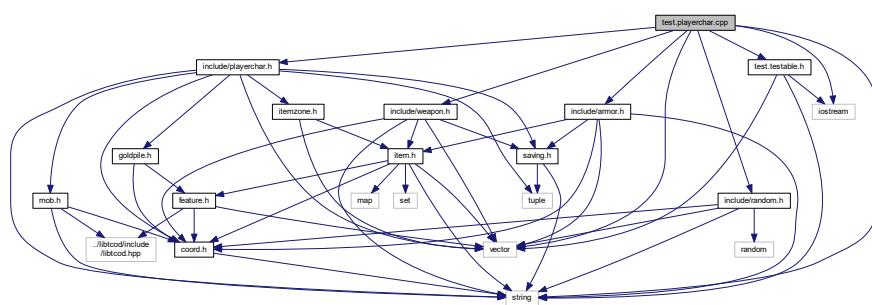
December 06, 2016

## 5.86 test.playerchar.cpp File Reference

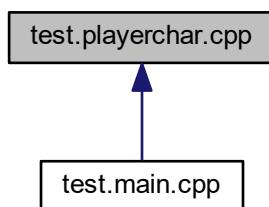
Global members.

```
#include <iostream>
#include <string>
#include <vector>
#include "include/armor.h"
#include "include/playerchar.h"
#include "include/random.h"
#include "include/weapon.h"
#include "test.testable.h"
```

Include dependency graph for test.playerchar.cpp:



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



## Classes

- class [PlayerCharTest](#)

### 5.86.1 Detailed Description

Global members.

Author

Team Rogue++

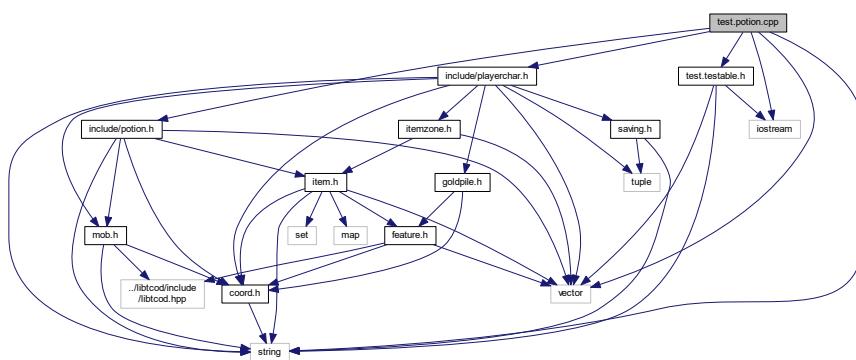
Date

December 06, 2016

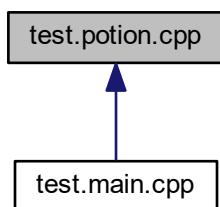
## 5.87 test.potion.cpp File Reference

Global members.

```
#include <iostream>
#include <string>
#include <vector>
#include "include/playerchar.h"
#include "include/potion.h"
#include "test.testable.h"
Include dependency graph for test.potion.cpp:
```



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



## Classes

- class [PotionTest](#)

### 5.87.1 Detailed Description

Global members.

#### Author

Team Rogue++

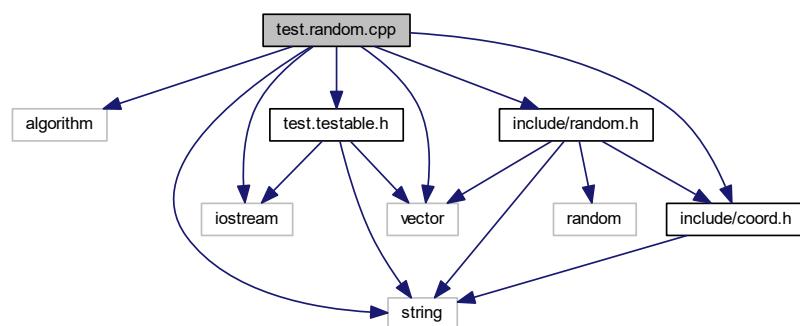
#### Date

December 06, 2016

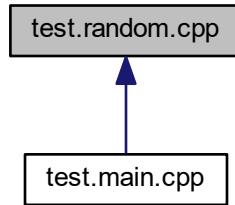
## 5.88 test.random.cpp File Reference

Global members.

```
#include <algorithm>
#include <iostream>
#include <string>
#include <vector>
#include "include/coord.h"
#include "include/random.h"
#include "test.testable.h"
Include dependency graph for test.random.cpp:
```



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



## Classes

- class [RandomTest](#)

### 5.88.1 Detailed Description

Global members.

#### Author

Team Rogue++

#### Date

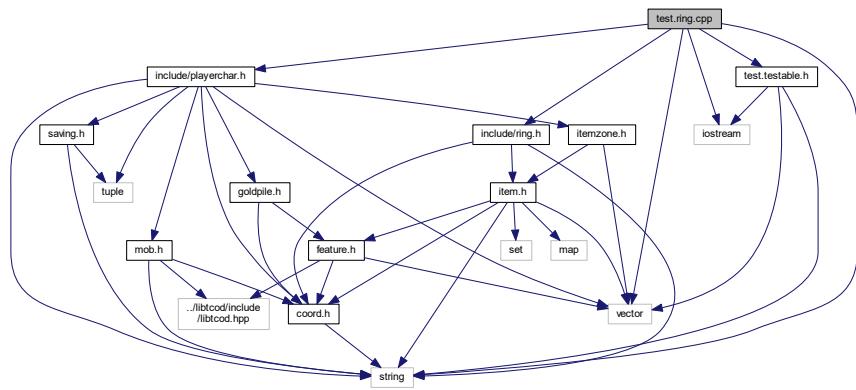
December 06, 2016

## 5.89 test.random.cpp File Reference

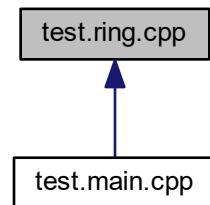
Global members.

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

```
#include "test.testable.h"
Include dependency graph for test.ring.cpp:
```



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



## Classes

- class [RingTest](#)

### 5.89.1 Detailed Description

Global members.

#### Author

Team Rogue++

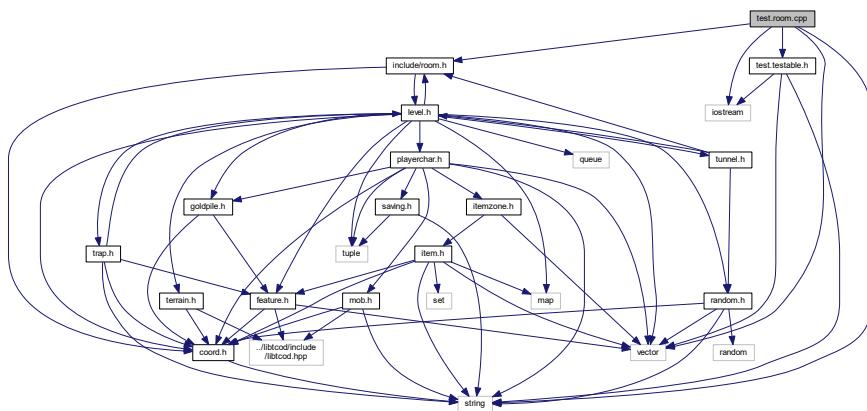
#### Date

December 06, 2016

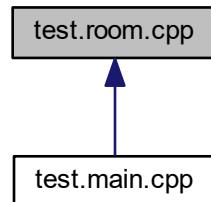
## 5.90 test.room.cpp File Reference

Global members.

```
#include <iostream>
#include <string>
#include <vector>
#include "include/room.h"
#include "test.testable.h"
Include dependency graph for test.room.cpp:
```



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



### Classes

- class [RoomTest](#)

#### 5.90.1 Detailed Description

Global members.

**Author**

Team Rogue++

**Date**

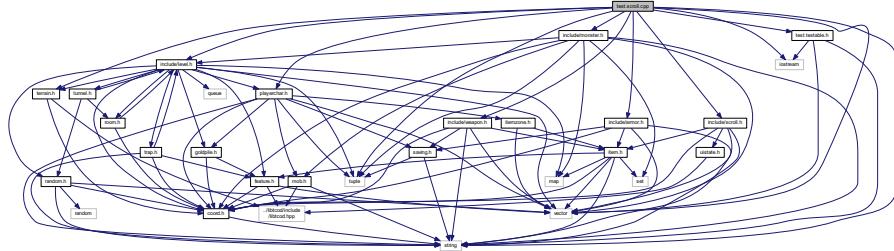
December 06, 2016

## 5.91 test.scroll.cpp File Reference

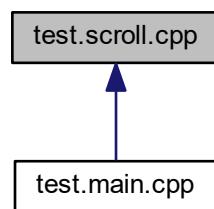
Global members.

```
#include <iostream>
#include <string>
#include <tuple>
#include <vector>
#include "include/armor.h"
#include "include/level.h"
#include "include/monster.h"
#include "include/playerchar.h"
#include "include/scroll.h"
#include "include/terrain.h"
#include "include/weapon.h"
#include "test.testable.h"
```

Include dependency graph for test.scroll.cpp:



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



## Classes

- class [ScrollTest](#)

### 5.91.1 Detailed Description

Global members.

#### Author

Team Rogue++

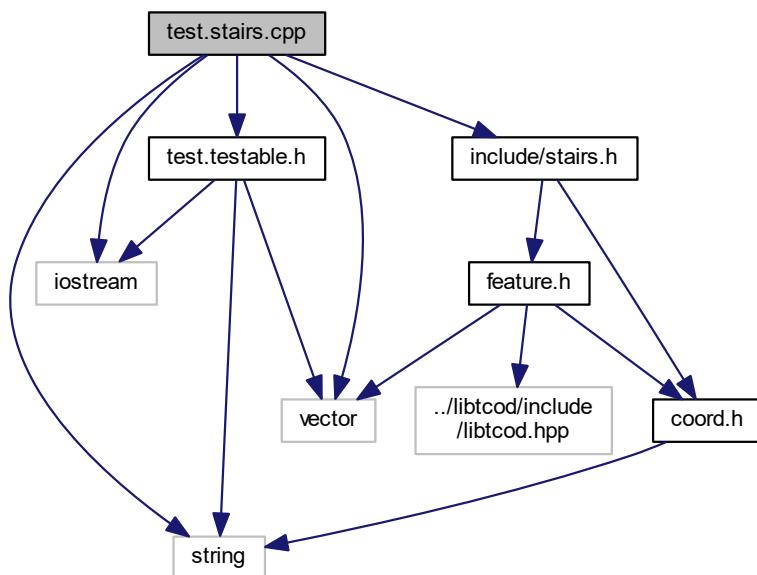
#### Date

December 06, 2016

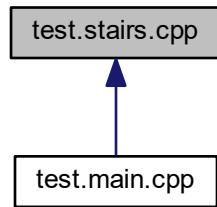
## 5.92 test.stairs.cpp File Reference

Global members.

```
#include <iostream>
#include <string>
#include <vector>
#include "include/stairs.h"
#include "test.testable.h"
Include dependency graph for test.stairs.cpp:
```



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



## Classes

- class [StairsTest](#)

### 5.92.1 Detailed Description

Global members.

#### Author

Team Rogue++

#### Date

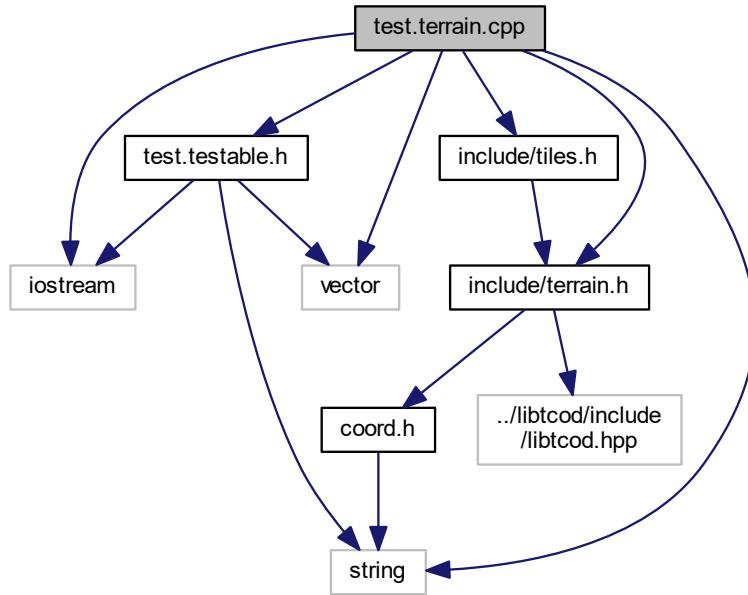
December 06, 2016

## 5.93 test.terrain.cpp File Reference

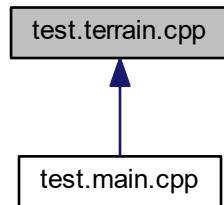
Global members.

```
#include <iostream>
#include <string>
#include <vector>
#include "include/terrain.h"
#include "include/tiles.h"
```

```
#include "test.testable.h"
Include dependency graph for test.terrain.cpp:
```



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



## Classes

- class [TerrainTest](#)

### 5.93.1 Detailed Description

Global members.

**Author**

Team Rogue++

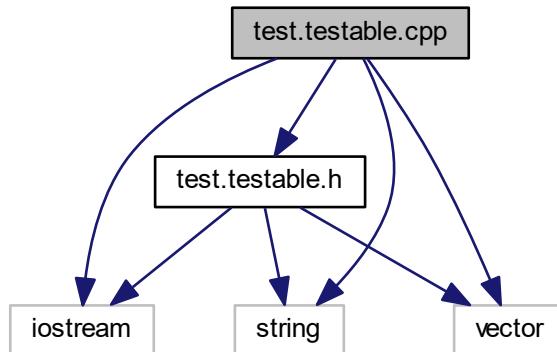
**Date**

December 06, 2016

## 5.94 test.testable.cpp File Reference

Global members.

```
#include <iostream>
#include <string>
#include <vector>
#include "test.testable.h"
Include dependency graph for test.testable.cpp:
```



### 5.94.1 Detailed Description

Global members.

**Author**

Team Rogue++

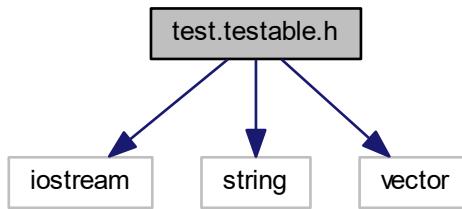
**Date**

December 06, 2016

## 5.95 test.testable.h File Reference

Member declarations for the [Testable](#) class.

```
#include <iostream>
#include <string>
#include <vector>
Include dependency graph for test.testable.h:
```



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



### Classes

- class [Testable](#)

#### 5.95.1 Detailed Description

Member declarations for the [Testable](#) class.

##### Author

Team Rogue++

##### Date

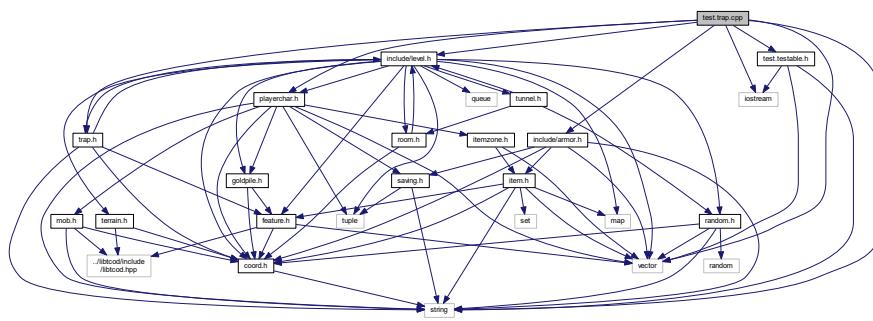
December 06, 2016

## 5.96 test.trap.cpp File Reference

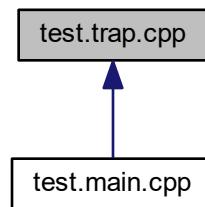
Global members.

```
#include <iostream>
#include <string>
#include <vector>
#include "include/armor.h"
#include "include/level.h"
#include "include/playerchar.h"
#include "include/trap.h"
#include "test.testable.h"
```

Include dependency graph for test.trap.cpp:



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



### Classes

- class [TrapTest](#)

#### 5.96.1 Detailed Description

Global members.

## Author

Team Rogue++

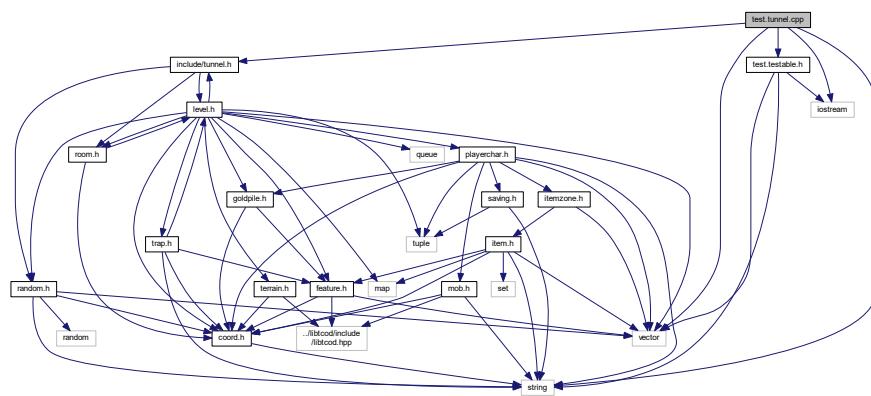
## Date

December 06, 2016

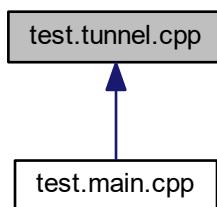
## 5.97 test.tunnel.cpp File Reference

Global members.

```
#include <iostream>
#include <string>
#include <vector>
#include "include/tunnel.h"
#include "test.testable.h"
Include dependency graph for test.tunnel.cpp:
```



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



## Classes

- class [TunnelTest](#)

### 5.97.1 Detailed Description

Global members.

Author

Team Rogue++

Date

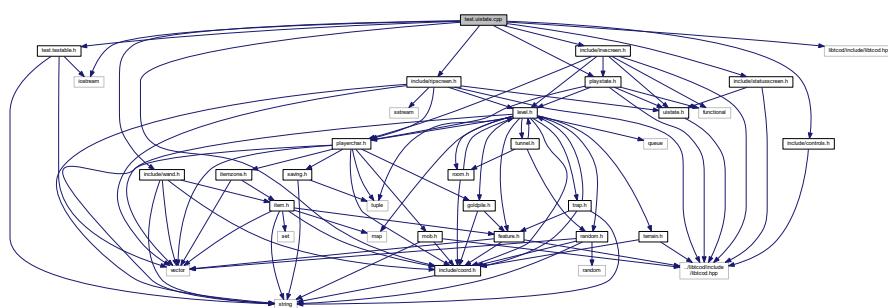
December 06, 2016

## 5.98 test.uistate.cpp File Reference

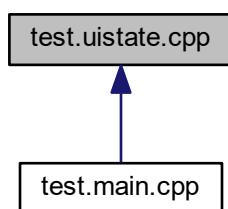
Global members.

```
#include <iostream>
#include "include/controls.h"
#include "include/coord.h"
#include "include/invscreen.h"
#include "include/playstate.h"
#include "include/ripscreen.h"
#include "include/statusscreen.h"
#include "include/wand.h"
#include "libtcod/include/libtcod.hpp"
#include "test.testable.h"
```

Include dependency graph for test.uistate.cpp:



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



## Classes

- class [UIStateTest](#)

### 5.98.1 Detailed Description

Global members.

#### Author

Team Rogue++

#### Date

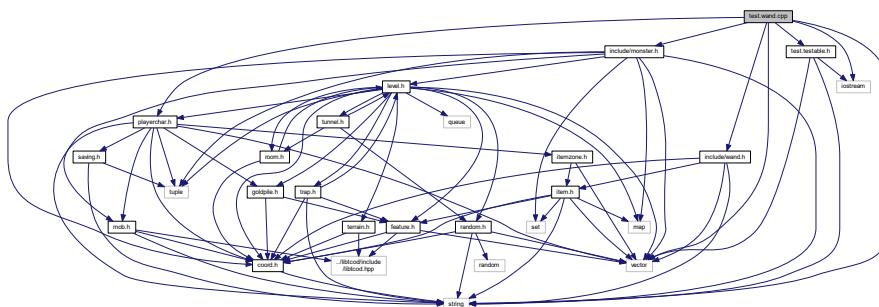
December 06, 2016

## 5.99 test.wand.cpp File Reference

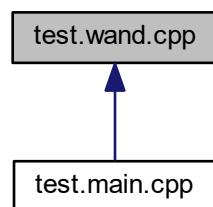
Global members.

```
#include <iostream>
#include <string>
#include <vector>
#include "include/monster.h"
#include "include/playerchar.h"
#include "include/wand.h"
#include "test.testable.h"
```

Include dependency graph for test.wand.cpp:



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



## Classes

- class [WandTest](#)

### 5.99.1 Detailed Description

Global members.

#### Author

Team Rogue++

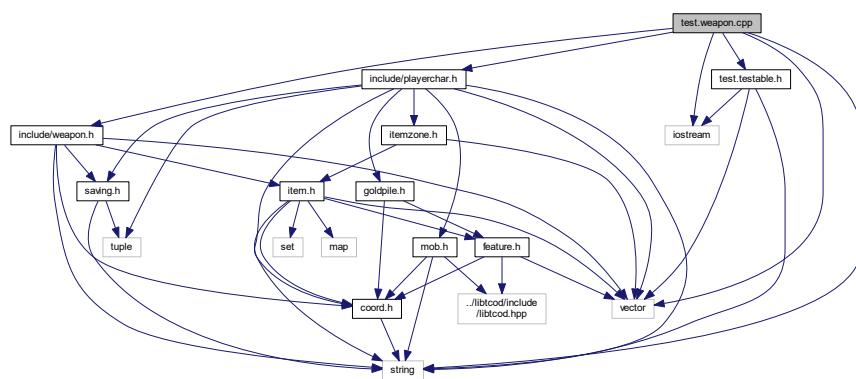
#### Date

December 06, 2016

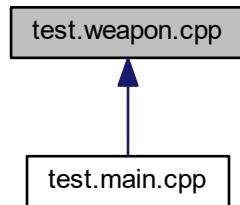
## 5.100 test.weapon.cpp File Reference

Global members.

```
#include <iostream>
#include <string>
#include <vector>
#include "include/playerchar.h"
#include "include/weapon.h"
#include "test.testable.h"
Include dependency graph for test.weapon.cpp:
```



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



## Classes

- class [WeaponTest](#)

### 5.100.1 Detailed Description

Global members.

#### Author

Team Rogue++

#### Date

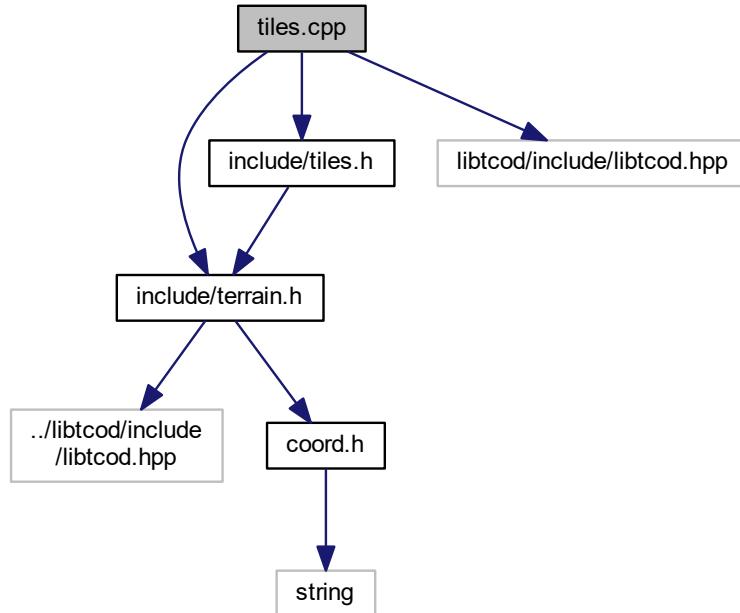
December 06, 2016

## 5.101 tiles.cpp File Reference

Member definitions for the [Corridor](#), [Door](#), [Floor](#), [Wall](#) classes.

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

```
#include "libtcod/include/libtcod.hpp"
Include dependency graph for tiles.cpp:
```



### 5.101.1 Detailed Description

Member definitions for the [Corridor](#), [Door](#), [Floor](#), [Wall](#) classes.

#### Author

Team Rogue++

#### Date

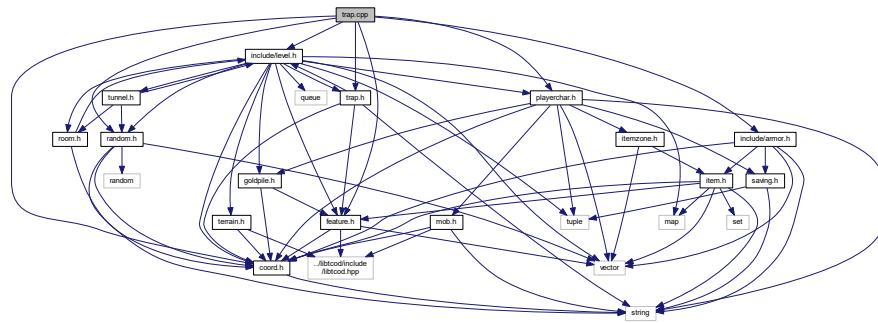
December 06, 2016

## 5.102 trap.cpp File Reference

Member definitions for the [Trap](#) class.

```
#include "include/armor.h"
#include "include/coord.h"
#include "include/feature.h"
#include "include/level.h"
#include "include/playerchar.h"
#include "include/random.h"
```

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



### 5.102.1 Detailed Description

Member definitions for the [Trap](#) class.

#### Author

Team Rogue++

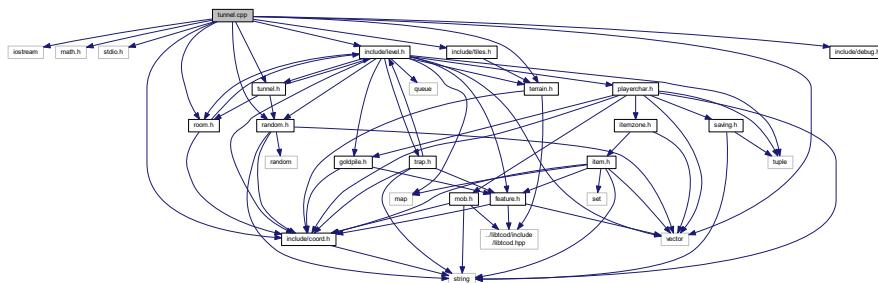
#### Date

December 06, 2016

## 5.103 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/debug.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:
```



### 5.103.1 Detailed Description

Member definitions for the [Tunnel](#) class.

#### Author

Team Rogue++

#### Date

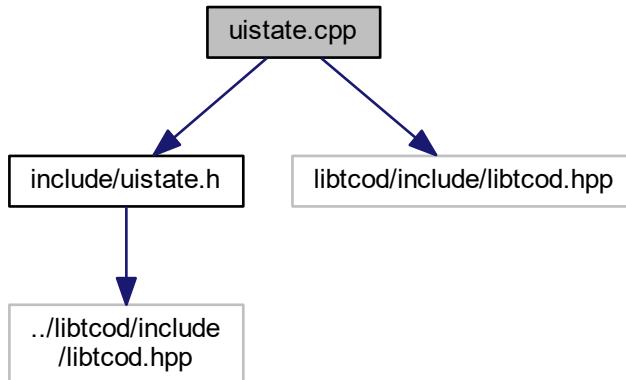
December 06, 2016

## 5.104 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:



### 5.104.1 Detailed Description

Member definitions for the [UIState](#) class.

#### Author

Team Rogue++

#### Date

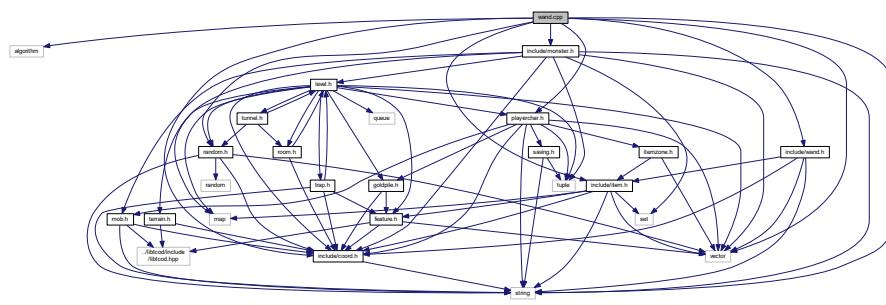
December 06, 2016

## 5.105 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/monster.h"
#include "include/playerchar.h"
#include "include/random.h"
#include "include/wand.h"
```

Include dependency graph for wand.cpp:



### 5.105.1 Detailed Description

Member definitions for the [Wand](#) class.

#### Author

Team Rogue++

#### Date

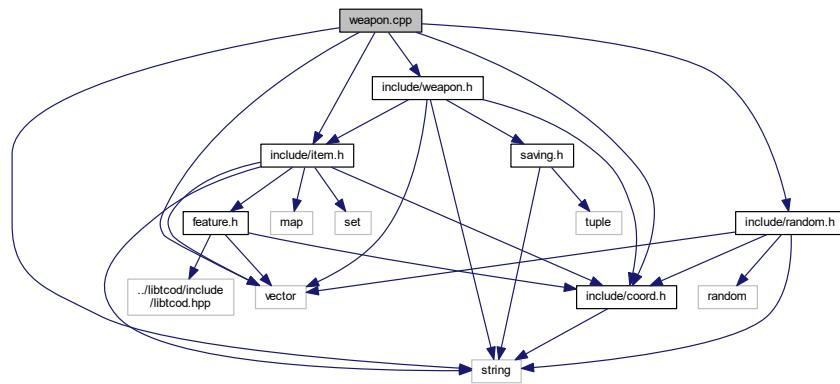
December 06, 2016

## 5.106 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:
```



### 5.106.1 Detailed Description

Member definitions for the [Weapon](#) class.

#### Author

Team Rogue++

#### Date

December 06, 2016

# Index

- ~Feature
  - Feature, 29
- ~Mob
  - Mob, 72
- ~PlayState
  - PlayState, 110
- ~Terrain
  - Terrain, 145
- activate
  - Food, 35
  - Potion, 112
  - Ring, 123
  - Scroll, 136
  - Trap, 152
  - Wand, 161
- activateItem
  - PlayerChar, 94
- add
  - ItemZone, 55
- addExp
  - PlayerChar, 94
- addFeature
  - Level, 58
- addFlag
  - Monster, 81
- addFrozenTurns
  - Monster, 83
- addHeader
  - codeformatter.py, 228
- Amulet, 13
  - Amulet, 14
- amulet.cpp, 167
- AmuletTest, 15
- appendLog
  - PlayerChar, 94
- applyCondition
  - PlayerChar, 95
- applyEffect
  - Item, 48
- Armor, 16
  - Armor, 17
  - getEnchantment, 18
  - getRating, 18
  - setEnchantment, 18
- armor
  - Mob, 77
- armor.cpp, 167
- ArmorTest, 19
- asScreen
- Coord, 21
- attack
  - Monster, 83
  - PlayerChar, 95
- attackConfuse
  - Monster, 83
- attackDrainLife
  - Monster, 83
- attackDropLevel
  - Monster, 84
- attackFreeze
  - Monster, 84
- attackRust
  - Monster, 84
- attackSteal
  - Monster, 84
- attackSting
  - Monster, 85
- bfsDiag
  - Level, 58
- bfsPerp
  - Level, 59
- calculateDamage
  - Monster, 85
  - PlayerChar, 95
- calculateHitChance
  - Monster, 85
  - PlayerChar, 95
- canSee
  - Level, 59
- changeArmor
  - Mob, 73
- changeCurrentHP
  - PlayerChar, 96
- changeCurrentStrength
  - PlayerChar, 96
- changeFoodLife
  - PlayerChar, 96
- changeMaxStrength
  - PlayerChar, 97
- character
  - Terrain, 147
- checked
  - Terrain, 147
- cleanPragmas
  - codeformatter.py, 229
- codeformatter.py
  - addHeader, 228

cleanPragmas, 229  
 formatContent, 229  
 formatFiles, 229  
 sortIncludes, 230  
 trim, 230  
 collectGold  
     PlayerChar, 97  
 contains  
     ItemZone, 55  
     Room, 129  
 Coord, 19  
     asScreen, 21  
     Coord, 21  
     copy, 21  
     isAdjacentTo, 21  
     ORTHO, 23  
     operator!=, 21  
     operator<, 22  
     operator\*, 22  
     operator\*=, 22  
     operator+, 22  
     operator+=, 22  
     operator-, 22  
     operator-=, 22  
     operator==, 23  
     operator[], 23  
     toString, 23  
 coord.cpp, 168  
 CoordTest, 24  
 copy  
     Coord, 21  
 Corridor, 24  
  
 deactivate  
     Ring, 123  
 dig  
     Room, 130  
     Tunnel, 155  
 DirectionPrompt, 26  
     handleInput, 27  
 Door, 27  
 draw  
     HelpScreen, 42  
     InvScreen, 44  
     LogScreen, 67  
     MainMenu, 69  
     RIPScreen, 128  
     StatusScreen, 141  
     SymbolScreen, 142  
 dropItem  
     PlayerChar, 97  
  
 eat  
     PlayerChar, 97  
 equipArmor  
     PlayerChar, 98  
 equipRingLeft  
     PlayerChar, 98  
 equipRingRight

PlayerChar, 98  
 equipWeapon  
     PlayerChar, 98  
 exists  
     Room, 130  
 exp  
     Mob, 77  
  
 fcolor  
     Feature, 31  
     Mob, 77  
 Feature, 28  
     ~Feature, 29  
     fcolor, 31  
     Feature, 29  
     getFColor, 30  
     getLocation, 30  
     getSymbol, 30  
     getVisible, 30  
     possibleColors, 31  
     setLocation, 31  
     setVisible, 31  
     visible, 31  
 feature.cpp, 169  
 FeatureTest, 32  
 Floor, 33  
 Food, 34  
     activate, 35  
     Food, 35  
 food.cpp, 170  
 FoodTest, 36  
 formatContent  
     codeformatter.py, 229  
 formatFiles  
     codeformatter.py, 229  
  
 Generator, 36  
     intFromRange, 37  
     nDx, 37  
     rand, 37  
     randBool, 37  
 getAdjPassable  
     Level, 59  
 getArmor  
     PlayerChar, 99  
 getArmorRating  
     Mob, 73  
     Monster, 85  
 getCarryChance  
     Monster, 86  
 getChance  
     Weapon, 165  
 getCharges  
     Wand, 161  
 getClassName  
     Item, 49  
 getColor  
     Terrain, 145  
 getContents

ItemZone, 55  
getContext  
    Item, 49  
getCurWeight  
    ItemZone, 55  
getDamage  
    Weapon, 165  
getDelay  
    Mob, 73  
    Monster, 86  
    PlayerChar, 99  
getDexterity  
    PlayerChar, 99  
getDisplayName  
    Item, 49  
getEnchantment  
    Armor, 18  
getEnchantments  
    Weapon, 165  
getExperience  
    Mob, 73  
getColor  
    Feature, 30  
    Mob, 73  
getFeatures  
    Level, 60  
getFoodLife  
    PlayerChar, 99  
getFoodStatus  
    PlayerChar, 99  
getGold  
    PlayerChar, 100  
getHP  
    Mob, 74  
getInventory  
    PlayerChar, 100  
getItem  
    ItemZone, 55  
getLevel  
    Mob, 74  
    PlayerChar, 100  
getLocation  
    Feature, 30  
    Mob, 74  
getLog  
    PlayerChar, 100  
getMaxHP  
    Mob, 74  
getMaxStrength  
    PlayerChar, 100  
getMaxWeight  
    ItemZone, 56  
getMobs  
    Level, 60  
getName  
    Item, 49  
    Mob, 74  
    Monster, 86  
getNearestGold  
    Level, 60  
getQuantity  
    GoldPile, 39  
getRandomEmptyPosition  
    Level, 60  
getRating  
    Armor, 18  
getRings  
    PlayerChar, 101  
getRooms  
    Level, 61  
getSaveFlag  
    PlayerChar, 101  
getSearchRadius  
    PlayerChar, 101  
getSightRadius  
    PlayerChar, 101  
getSize  
    ItemZone, 56  
getStrength  
    PlayerChar, 102  
getSymbol  
    Feature, 30  
    Mob, 75  
    Terrain, 145  
getSymbolsForLevel  
    Monster, 86  
getSymbolsForTreasure  
    Monster, 87  
getType  
    Item, 49  
getVisibility  
    Terrain, 146  
getVisible  
    Feature, 30  
getWeapon  
    PlayerChar, 102  
getWeight  
    Item, 50  
GoldPile, 38  
    getQuantity, 39  
    GoldPile, 39  
GoldPileTest, 40  
goldpile.cpp, 171  
handleInput  
    DirectionPrompt, 27  
    HelpScreen, 42  
    InvScreen, 44  
    LogScreen, 67  
    MainMenu, 69  
    PlayState, 110  
    QuickThrow, 115  
    QuickUse, 117  
    QuickZap, 118  
    QuitPrompt2, 120  
    RIPScreen, 128  
    RingRemovePrompt, 125

StatusScreen, 141  
 SymbolScreen, 142  
 hasAmulet  
     PlayerChar, 102  
 hasCondition  
     PlayerChar, 102  
 hasEffect  
     Item, 50  
 HelpScreen, 40  
     draw, 42  
     handleInput, 42  
     HelpScreen, 42  
 helpscreen.cpp, 172  
 hit  
     Mob, 75  
     Monster, 87  
     PlayerChar, 103  
 include/amulet.h, 173  
 include/armor.h, 175  
 include/controls.h, 176  
 include/coord.h, 178  
 include/debug.h, 179  
 include/feature.h, 179  
 include/food.h, 180  
 include/globals.h, 181  
 include/goldpile.h, 183  
 include/helpscreen.h, 184  
 include/invscreen.h, 185  
 include/item.h, 187  
 include/itemzone.h, 188  
 include/level.h, 189  
 include/logscreen.h, 190  
 include/mainmenu.h, 191  
 include/mastercontroller.h, 193  
 include/mob.h, 194  
 include/monster.h, 195  
 include/playerchar.h, 196  
 include/playstate.h, 197  
 include/potion.h, 198  
 include/random.h, 199  
 include/ring.h, 201  
 include/ripscreen.h, 202  
 include/room.h, 203  
 include/savescreen.h, 205  
 include/saving.h, 206  
 include scroll.h, 207  
 include/stairs.h, 208  
 include/statusscreen.h, 210  
 include/symbolscreen.h, 211  
 include/terrain.h, 212  
 include/tiles.h, 214  
 include/trap.h, 215  
 include/tunnel.h, 216  
 include/uistate.h, 217  
 include/wand.h, 218  
 include/weapon.h, 219  
 include/wizard.h, 221  
 initializeScrollNames  
     Scroll, 136  
 intFromRange  
     Generator, 37  
 InvScreen, 42  
     draw, 44  
     handleInput, 44  
     InvScreen, 44  
 invscreen.cpp, 221  
 isAdjacentTo  
     Coord, 21  
 isAwake  
     Monster, 87  
 isCursed  
     Item, 50  
 isDead  
     Mob, 75  
 isIdentified  
     Item, 50  
 isMelee  
     Weapon, 165  
 isPassable  
     Terrain, 146  
 isSeen  
     Terrain, 146  
 isStackable  
     Item, 51  
 isThrowable  
     Item, 51  
 isVisible  
     Monster, 87  
 Item, 45  
     applyEffect, 48  
     getClassName, 49  
     getContext, 49  
     getDisplayName, 49  
     getName, 49  
     getType, 49  
     getWeight, 50  
     hasEffect, 50  
     isCursed, 50  
     isIdentified, 50  
     isStackable, 51  
     isThrowable, 51  
     Item, 47, 48  
     operator<, 51  
     operator==, 51  
     removeEffect, 52  
     setContext, 52  
     setIdentified, 52  
     shuffleNameVector, 52  
 item.cpp, 222  
 ItemTest, 53  
 ItemZone, 54  
     add, 55  
     contains, 55  
     getContents, 55  
     getCurrWeight, 55  
     getItem, 55

getMaxWeight, 56  
getSize, 56  
ItemZone, 55  
ItemZoneTest, 56

Level, 57  
addFeature, 58  
bfsDiag, 58  
bfsPerp, 59  
canSee, 59  
getAdjPassable, 59  
getFeatures, 60  
getMobs, 60  
getNearestGold, 60  
getRandomEmptyPosition, 60  
getRooms, 61  
monsterAt, 61  
popTurnClock, 61  
pushMob, 61  
registerMob, 63  
removeFeature, 63  
removeMob, 63  
throwLocation, 63

level  
    Mob, 77  
    PlayState, 110

level.cpp, 223

LevelGenTest, 64

LevelTest, 65

location  
    Mob, 77

LogScreen, 66  
    draw, 67  
    handleInput, 67

logscreen.cpp, 224

MAX\_TYPE  
    Trap, 153

main.cpp, 225

MainMenu, 68  
    draw, 69  
    handleInput, 69  
    MainMenu, 69

mainmenu.cpp, 226

Mapped  
    Terrain, 144

MasterController, 69

mastercontroller.cpp, 227

maxHP  
    Mob, 77

misc/codeformatter.py, 228

misc/stringfinder.py, 230

Mob, 70  
    ~Mob, 72  
    armor, 77  
    changeArmor, 73  
    exp, 77  
    fcolor, 77  
    getArmorRating, 73

getDelay, 73  
getExperience, 73  
getFColor, 73  
getHP, 74  
getLevel, 74  
getLocation, 74  
getMaxHP, 74  
getName, 74  
getSymbol, 75  
hit, 75  
isDead, 75  
level, 77  
location, 77  
maxHP, 77  
Mob, 72  
moveLocation, 75  
name, 77  
setCurrentHP, 76  
setFColor, 76  
setLocation, 76  
setMaxHP, 76

mob.cpp, 231

MobTest, 78

Monster, 79  
    addFlag, 81  
    addFrozenTurns, 83  
    attack, 83  
    attackConfuse, 83  
    attackDrainLife, 83  
    attackDropLevel, 84  
    attackFreeze, 84  
    attackRust, 84  
    attackSteal, 84  
    attackSting, 85  
    calculateDamage, 85  
    calculateHitChance, 85  
    getArmorRating, 85  
    getCarryChance, 86  
    getDelay, 86  
    getName, 86  
    getSymbolsForLevel, 86  
    getSymbolsForTreasure, 87  
    hit, 87  
    isAwake, 87  
    isVisible, 87  
    Monster, 81  
    randomMonster, 88  
    removeFlag, 88  
    setAwake, 88  
    setVisible, 88  
    turn, 89

monster.cpp, 232

monsterAt  
    Level, 61

MonsterTest, 89

move  
    PlayerChar, 103

moveLocation

Mob, 75

nDx  
Generator, 37

name  
Mob, 77

ORTHO  
Coord, 23

operator!=  
Coord, 21

operator<  
Coord, 22  
Item, 51

operator\*  
Coord, 22

operator\*=  
Coord, 22

operator+  
Coord, 22

operator+=  
Coord, 22

operator-  
Coord, 22

operator-=  
Coord, 22

operator==  
Coord, 23  
Item, 51

operator[]  
Coord, 23

parent  
Terrain, 147

Passability  
Terrain, 144

passable  
Terrain, 147

pickupItem  
PlayerChar, 103

PlayState, 108  
~PlayState, 110  
handleInput, 110  
level, 110  
PlayState, 110  
player, 110

player  
PlayState, 110

PlayerChar, 90  
activateItem, 94  
addExp, 94  
appendLog, 94  
applyCondition, 95  
attack, 95  
calculateDamage, 95  
calculateHitChance, 95  
changeCurrentHP, 96  
changeCurrentStrength, 96  
changeFoodLife, 96

changeMaxStrength, 97  
collectGold, 97  
dropItem, 97  
eat, 97  
equipArmor, 98  
equipRingLeft, 98  
equipRingRight, 98  
equipWeapon, 98  
getArmor, 99  
getDelay, 99  
getDexterity, 99  
getFoodLife, 99  
getFoodStatus, 99  
getGold, 100  
getInventory, 100  
getLevel, 100  
getLog, 100  
getMaxStrength, 100  
getRings, 101  
getSaveFlag, 101  
getSearchRadius, 101  
getSightRadius, 101  
getStrength, 102  
getWeapon, 102  
hasAmulet, 102  
hasCondition, 102  
hit, 103  
move, 103  
pickupItem, 103  
PlayerChar, 94  
quaff, 104  
removeArmor, 104  
removeCondition, 104  
removeRingLeft, 104  
removeRingRight, 104  
removeWeapon, 105  
setDexterity, 105  
setFoodLife, 105  
setGold, 105  
setSaveFlag, 106  
setStrength, 106  
update, 106

PlayerCharTest, 107

playerchar.cpp, 233

playstate.cpp, 233

popTurnClock  
Level, 61

possibleColors  
Feature, 31

Potion, 111  
activate, 112  
Potion, 112

potion.cpp, 235

PotionTest, 113

printInfo  
Room, 130

pushMob  
Level, 61

quaff  
    PlayerChar, 104  
QuickThrow, 114  
    handleInput, 115  
QuickUse  
    handleInput, 117  
QuickUse< T >, 116  
QuickZap, 117  
    handleInput, 118  
QuitPrompt2, 119  
    handleInput, 120  
  
RIPScreen, 126  
    draw, 128  
    handleInput, 128  
    RIPScreen, 128  
rand  
    Generator, 37  
randBool  
    Generator, 37  
random.cpp, 236  
randomMonster  
    Monster, 88  
RandomTest, 120  
randomTrap  
    Trap, 152  
registerMob  
    Level, 63  
removeArmor  
    PlayerChar, 104  
removeCondition  
    PlayerChar, 104  
removeEffect  
    Item, 52  
removeFeature  
    Level, 63  
removeFlag  
    Monster, 88  
removeMob  
    Level, 63  
removeRingLeft  
    PlayerChar, 104  
removeRingRight  
    PlayerChar, 104  
removeWeapon  
    PlayerChar, 105  
Ring, 121  
    activate, 123  
    deactivate, 123  
    Ring, 122, 123  
ring.cpp, 236  
RingRemovePrompt, 124  
    handleInput, 125  
RingTest, 126  
ripscreen.cpp, 237  
Room, 128  
    contains, 129  
    dig, 130  
    exists, 130  
                printInfo, 130  
                touches, 130  
room.cpp, 238  
RoomTest, 131  
SaveScreen, 132  
savescreen.cpp, 239  
saving.cpp, 240  
ScoreItem, 133  
Scroll, 134  
    activate, 136  
    initializeScrollNames, 136  
    Scroll, 135  
scroll.cpp, 241  
ScrollTest, 137  
setAwake  
    Monster, 88  
setContext  
    Item, 52  
setCurrentHP  
    Mob, 76  
setDexterity  
    PlayerChar, 105  
setEnchantment  
    Armor, 18  
setEnchantments  
    Weapon, 165  
setFColor  
    Mob, 76  
setFoodLife  
    PlayerChar, 105  
setGold  
    PlayerChar, 105  
setIdentified  
    Item, 52  
setIsSeen  
    Terrain, 146  
setLocation  
    Feature, 31  
    Mob, 76  
setMaxHP  
    Mob, 76  
setPassable  
    Terrain, 146  
setSaveFlag  
    PlayerChar, 106  
setStrength  
    PlayerChar, 106  
setSymbol  
    Terrain, 147  
setVisible  
    Feature, 31  
    Monster, 88  
shuffleNameVector  
    Item, 52  
sortIncludes  
    codeformatter.py, 230  
Stairs, 138  
stairs.cpp, 241

StairsTest, 139  
StatusScreen, 139  
  draw, 141  
  handleInput, 141  
  StatusScreen, 140  
statusscreen.cpp, 242  
SymbolScreen, 141  
  draw, 142  
  handleInput, 142  
  SymbolScreen, 142  
symbolscreen.cpp, 243  
  
Terrain, 143  
  ~Terrain, 145  
  character, 147  
  checked, 147  
  getColor, 145  
  getSymbol, 145  
  getVisibility, 146  
  isPassable, 146  
  isSeen, 146  
  Mapped, 144  
  parent, 147  
  Passability, 144  
  passable, 147  
  setIsSeen, 146  
  setPassable, 146  
  setSymbol, 147  
  Terrain, 145  
  visible, 148  
terrain.cpp, 244  
TerrainTest, 148  
test.amulet.cpp, 245  
test.armor.cpp, 246  
test.coord.cpp, 247  
test.feature.cpp, 248  
test.food.cpp, 250  
test.goldpile.cpp, 251  
test.item.cpp, 252  
test.itemzone.cpp, 254  
test.level.cpp, 255  
test.levelgen.cpp, 256  
test.main.cpp, 257  
test.mob.cpp, 258  
test.monster.cpp, 259  
test.playerchar.cpp, 260  
test.potion.cpp, 261  
test.random.cpp, 262  
test.ring.cpp, 263  
test.room.cpp, 265  
test.scroll.cpp, 266  
test.stairs.cpp, 267  
test.terrain.cpp, 268  
test.testable.cpp, 270  
test.testable.h, 271  
test.trap.cpp, 272  
test.tunnel.cpp, 273  
test.uistate.cpp, 274  
test.wand.cpp, 275  
  
test.weapon.cpp, 276  
Testable, 150  
throwLocation  
  Level, 63  
tiles.cpp, 277  
toString  
  Coord, 23  
touches  
  Room, 130  
Trap, 151  
  activate, 152  
  MAX\_TYPE, 153  
  randomTrap, 152  
  Trap, 152  
trap.cpp, 278  
TrapTest, 153  
trim  
  codeformatter.py, 230  
Tunnel, 154  
  dig, 155  
  Tunnel, 154  
tunnel.cpp, 279  
TunnelTest, 155  
turn  
  Monster, 89  
  
UIState, 156  
UIStateTest, 157  
uistate.cpp, 280  
update  
  PlayerChar, 106  
  
visible  
  Feature, 31  
  Terrain, 148  
  
Wall, 158  
Wand, 159  
  activate, 161  
  getCharges, 161  
  Wand, 160, 161  
wand.cpp, 281  
WandTest, 162  
Weapon, 162  
  getChance, 165  
  getDamage, 165  
  getEnchantments, 165  
  isMelee, 165  
  setEnchantments, 165  
  Weapon, 164  
weapon.cpp, 281  
WeaponTest, 166