

Bronze Adventures

Generated by Doxygen 1.8.16

1 Documentation	1
1.1 Overview	1
1.2 Instructions	1
1.2.1 Moving	1
1.2.2 Attacking	1
1.2.3 Building	1
1.2.4 Altars	2
1.2.5 Winning	2
1.3 distribution	2
1.4 content	2
2 Programming 3 Exercise project - CourseLibrary	3
2.1 Documentation	3
3 The Bronze Adventures game!	5
3.1 Aditonal note:	5
4 Hierarchical Index	7
4.1 Class Hierarchy	7
5 Class Index	9
5.1 Class List	9
6 Class Documentation	13
6.1 Whiskas::AltarBase Class Reference	13
6.1.1 Detailed Description	14
6.1.2 Member Function Documentation	15
6.1.2.1 getType()	15
6.2 Whiskas::Attackable Class Reference	15
6.2.1 Detailed Description	16
6.2.2 Constructor & Destructor Documentation	16
6.2.2.1 Attackable()	16
6.2.3 Member Function Documentation	17
6.2.3.1 getAttack()	17
6.2.3.2 getAttacked()	17
6.2.3.3 getBoundID()	17
6.2.3.4 getHealth()	18
6.2.3.5 modifyAttack()	18
6.2.3.6 modifyHealth()	18
6.2.3.7 setAttacked()	18
6.3 Course::BaseException Class Reference	19
6.3.1 Detailed Description	20
6.3.2 Constructor & Destructor Documentation	20
6.3.2.1 BaseException()	20

6.3.3 Member Function Documentation	20
6.3.3.1 msg()	20
6.4 Course::BasicWorker Class Reference	21
6.4.1 Detailed Description	22
6.4.2 Constructor & Destructor Documentation	23
6.4.2.1 BasicWorker()	23
6.4.3 Member Function Documentation	23
6.4.3.1 canBePlacedOnTile()	23
6.4.3.2 getType()	24
6.4.3.3 tileWorkAction()	25
6.5 Course::BuildingBase Class Reference	25
6.5.1 Detailed Description	27
6.5.2 Constructor & Destructor Documentation	27
6.5.2.1 BuildingBase()	27
6.5.3 Member Function Documentation	28
6.5.3.1 addHoldMarkers()	28
6.5.3.2 canBePlacedOnTile()	28
6.5.3.3 getProduction()	29
6.5.3.4 getType()	30
6.5.3.5 holdCount()	30
6.6 buildingDialog Class Reference	31
6.6.1 Detailed Description	32
6.7 Course::Coordinate Class Reference	32
6.7.1 Constructor & Destructor Documentation	33
6.7.1.1 Coordinate() [1/3]	33
6.7.1.2 Coordinate() [2/3]	33
6.7.1.3 Coordinate() [3/3]	33
6.7.1.4 ~Coordinate()	34
6.7.2 Member Function Documentation	34
6.7.2.1 neighbour_at()	34
6.7.2.2 neighbours()	35
6.7.2.3 operator"!="()	36
6.7.2.4 operator+()	36
6.7.2.5 operator+=()	37
6.7.2.6 operator-()	38
6.7.2.7 operator-=()	38
6.7.2.8 operator<()	39
6.7.2.9 operator<=()	40
6.7.2.10 operator=()	40
6.7.2.11 operator==()	41
6.7.2.12 operator>()	42
6.7.2.13 operator>=()	43

6.7.2.14 set_x()	43
6.7.2.15 set_y()	44
6.7.2.16 travel()	44
6.7.2.17 x()	45
6.7.2.18 y()	46
6.8 Whiskas::CustomBuildingBase Class Reference	48
6.9 default_coordinate Class Reference	49
6.10 default_gameobjects Class Reference	50
6.11 default_playerbase Class Reference	51
6.12 default_test Class Reference	52
6.13 Whiskas::Desert Class Reference	52
6.13.1 Detailed Description	53
6.13.2 Member Function Documentation	54
6.13.2.1 addBuilding()	54
6.13.2.2 getType()	54
6.14 endDialog Class Reference	55
6.14.1 Detailed Description	55
6.15 Course::Farm Class Reference	56
6.15.1 Detailed Description	57
6.15.2 Constructor & Destructor Documentation	57
6.15.2.1 Farm()	57
6.15.3 Member Function Documentation	58
6.15.3.1 getType()	58
6.16 Course::Forest Class Reference	59
6.16.1 Detailed Description	60
6.16.2 Constructor & Destructor Documentation	60
6.16.2.1 Forest()	60
6.16.3 Member Function Documentation	61
6.16.3.1 addBuilding()	61
6.16.3.2 getType()	62
6.17 Whiskas::gameEventHandler Class Reference	62
6.17.1 Detailed Description	64
6.17.2 Constructor & Destructor Documentation	64
6.17.2.1 gameEventHandler()	64
6.17.3 Member Function Documentation	64
6.17.3.1 endTurn()	64
6.17.3.2 getActiveMinion()	64
6.17.3.3 getActiveTile()	65
6.17.3.4 getTurn()	65
6.17.3.5 handleLeftClick()	65
6.17.3.6 handleMwindowClick()	66
6.17.3.7 handleRightClick()	66

6.17.3.8 modifyResource()	67
6.17.3.9 modifyResources()	67
6.17.3.10 setActiveTile()	68
6.17.3.11 subtractPlayerResources()	68
6.18 Whiskas::gameManager Class Reference	68
6.18.1 Detailed Description	70
6.18.2 Constructor & Destructor Documentation	71
6.18.2.1 gameManager()	71
6.18.3 Member Function Documentation	71
6.18.3.1 addBuilding()	71
6.18.3.2 addMinion()	71
6.18.3.3 addPlayer()	72
6.18.3.4 addTile()	72
6.18.3.5 addTiles()	72
6.18.3.6 attack()	73
6.18.3.7 attackMultiple()	74
6.18.3.8 checkBuildingAvailability()	74
6.18.3.9 checkForEnemies()	75
6.18.3.10 destroyObject()	75
6.18.3.11 getBuildingVector()	76
6.18.3.12 getMinionVector()	76
6.18.3.13 getNexusLocation()	76
6.18.3.14 getPlayerPair()	77
6.18.3.15 getTile() [1/2]	77
6.18.3.16 getTile() [2/2]	77
6.18.3.17 getTiles()	78
6.18.3.18 getTileVector()	78
6.18.3.19 getWinner()	79
6.18.3.20 move()	79
6.18.3.21 selectAttackTarget()	80
6.18.3.22 spawnBuilding()	80
6.18.3.23 spawnMinion()	81
6.19 Course::GameObject Class Reference	81
6.19.1 Detailed Description	83
6.19.2 Constructor & Destructor Documentation	84
6.19.2.1 GameObject() [1/5]	84
6.19.2.2 GameObject() [2/5]	84
6.19.2.3 GameObject() [3/5]	84
6.19.2.4 GameObject() [4/5]	85
6.19.2.5 GameObject() [5/5]	85
6.19.3 Member Function Documentation	85
6.19.3.1 addDescription()	85

6.19.3.2 getCoordinate()	86
6.19.3.3 getCoordinatePtr()	87
6.19.3.4 getDescription()	87
6.19.3.5 getDescriptions()	88
6.19.3.6 getOwner()	88
6.19.3.7 getType()	89
6.19.3.8 hasSameCoordinateAs()	89
6.19.3.9 hasSameOwnerAs()	90
6.19.3.10 lockEventHandler()	91
6.19.3.11 lockObjectManager()	91
6.19.3.12 removeDescription()	92
6.19.3.13 removeDescriptions()	92
6.19.3.14 setCoordinate() [1/2]	93
6.19.3.15 setCoordinate() [2/2]	93
6.19.3.16 setDescription()	94
6.19.3.17 setDescriptions()	94
6.19.3.18 setOwner()	94
6.19.3.19 unsetCoordinate()	95
6.20 Whiskas::GameScene Class Reference	95
6.20.1 Constructor & Destructor Documentation	96
6.20.1.1 GameScene()	97
6.20.2 Member Function Documentation	98
6.20.2.1 drawItem()	98
6.20.2.2 event()	98
6.20.2.3 getLastCoordinate()	99
6.20.2.4 getLastID()	99
6.20.2.5 removeItem()	99
6.20.2.6 setScale()	100
6.20.2.7 setSize()	101
6.20.2.8 updateItem()	101
6.21 Course::Grassland Class Reference	102
6.21.1 Detailed Description	103
6.21.2 Constructor & Destructor Documentation	104
6.21.2.1 Grassland()	104
6.21.3 Member Function Documentation	104
6.21.3.1 getType()	104
6.22 Course::HeadQuarters Class Reference	105
6.22.1 Detailed Description	106
6.22.2 Constructor & Destructor Documentation	107
6.22.2.1 HeadQuarters()	107
6.22.3 Member Function Documentation	107
6.22.3.1 getType()	107

6.22.3.2 onBuildAction()	108
6.23 Course::iGameEventHandler Class Reference	109
6.23.1 Detailed Description	109
6.23.2 Member Function Documentation	109
6.23.2.1 modifyResource()	110
6.23.2.2 modifyResources()	111
6.24 Course::IllegalAction Class Reference	112
6.24.1 Detailed Description	113
6.24.2 Constructor & Destructor Documentation	113
6.24.2.1 IllegalAction()	113
6.24.2.2 ~IllegalAction()	113
6.25 Course::InvalidPointer Class Reference	113
6.25.1 Detailed Description	114
6.25.2 Constructor & Destructor Documentation	115
6.25.2.1 InvalidPointer()	115
6.25.2.2 ~InvalidPointer()	115
6.26 Course::iObjectManager Class Reference	115
6.26.1 Detailed Description	116
6.26.2 Member Function Documentation	116
6.26.2.1 addTiles()	116
6.26.2.2 getTile() [1/2]	116
6.26.2.3 getTile() [2/2]	117
6.26.2.4 getTiles()	117
6.27 Whiskas::Jungle Class Reference	118
6.27.1 Detailed Description	119
6.27.2 Member Function Documentation	119
6.27.2.1 getType()	119
6.28 Course::KeyError Class Reference	120
6.28.1 Detailed Description	121
6.28.2 Constructor & Destructor Documentation	121
6.28.2.1 KeyError()	121
6.28.2.2 ~KeyError()	121
6.29 Whiskas::LeaguePlayer Class Reference	121
6.29.1 Detailed Description	122
6.29.2 Constructor & Destructor Documentation	122
6.29.2.1 LeaguePlayer()	122
6.29.3 Member Function Documentation	123
6.29.3.1 setPlayerItems()	123
6.30 Whiskas::LifePump Class Reference	123
6.30.1 Detailed Description	125
6.30.2 Member Function Documentation	125
6.30.2.1 getType()	126

6.31 Whiskas::MageAltar Class Reference	126
6.31.1 Detailed Description	127
6.31.2 Member Function Documentation	127
6.31.2.1 getType()	128
6.32 Whiskas::MagicChampion Class Reference	128
6.32.1 Detailed Description	129
6.32.2 Member Function Documentation	129
6.32.2.1 getType()	129
6.33 ManagerTest Class Reference	130
6.34 Whiskas::MapItem Class Reference	130
6.34.1 Detailed Description	132
6.34.2 Constructor & Destructor Documentation	132
6.34.2.1 MapItem()	132
6.34.3 Member Function Documentation	132
6.34.3.1 boundingRect()	132
6.34.3.2 getBoundObject()	133
6.34.3.3 isSameObj()	133
6.34.3.4 paint()	133
6.34.3.5 setSize()	134
6.35 MapWindow Class Reference	134
6.35.1 Detailed Description	136
6.35.2 Member Function Documentation	136
6.35.2.1 drawItem()	136
6.35.2.2 initMap	136
6.35.2.3 mousePressEvent()	137
6.35.2.4 openBuildingDialog()	137
6.35.2.5 setGEHandler()	137
6.35.2.6 setGManager()	137
6.35.2.7 setSize()	138
6.36 Whiskas::MeleeAltar Class Reference	138
6.36.1 Detailed Description	140
6.36.2 Member Function Documentation	140
6.36.2.1 getType()	141
6.37 Whiskas::MeleeChampion Class Reference	141
6.37.1 Detailed Description	142
6.37.2 Member Function Documentation	142
6.37.2.1 getType()	142
6.37.2.2 modifyHealth()	143
6.38 Whiskas::Minion Class Reference	143
6.38.1 Detailed Description	144
6.38.2 Member Function Documentation	144
6.38.2.1 getType()	145

6.38.2.2 modifyHealth()	145
6.39 Whiskas::Mountain Class Reference	145
6.39.1 Detailed Description	146
6.39.2 Member Function Documentation	147
6.39.2.1 addBuilding()	147
6.39.2.2 getType()	147
6.40 Whiskas::Nexus Class Reference	148
6.40.1 Detailed Description	149
6.40.2 Member Function Documentation	149
6.40.2.1 getType()	150
6.41 Course::NotEnoughSpace Class Reference	150
6.41.1 Detailed Description	151
6.41.2 Constructor & Destructor Documentation	151
6.41.2.1 NotEnoughSpace()	151
6.41.2.2 ~NotEnoughSpace()	152
6.42 Course::Outpost Class Reference	152
6.42.1 Detailed Description	153
6.42.2 Constructor & Destructor Documentation	154
6.42.2.1 Outpost()	154
6.42.3 Member Function Documentation	154
6.42.3.1 getProduction()	154
6.42.3.2 getType()	155
6.42.3.3 onBuildAction()	155
6.43 Course::OwnerConflict Class Reference	156
6.43.1 Detailed Description	157
6.43.2 Constructor & Destructor Documentation	157
6.43.2.1 OwnerConflict()	157
6.43.2.2 ~OwnerConflict()	158
6.44 Course::PlaceableGameObject Class Reference	158
6.44.1 Detailed Description	159
6.44.2 Constructor & Destructor Documentation	159
6.44.2.1 PlaceableGameObject()	159
6.44.3 Member Function Documentation	160
6.44.3.1 canBePlacedOnTile()	160
6.44.3.2 currentLocationTile()	161
6.44.3.3 getType()	162
6.44.3.4 setLocationTile()	162
6.44.3.5 spacesInTileCapacity()	163
6.45 Course::PlayerBase Class Reference	164
6.45.1 Detailed Description	165
6.45.2 Constructor & Destructor Documentation	165
6.45.2.1 PlayerBase()	165

6.45.3 Member Function Documentation	165
6.45.3.1 addObject()	165
6.45.3.2 addObjects()	166
6.45.3.3 getName()	166
6.45.3.4 getObjects()	167
6.45.3.5 removeObject() [1/2]	167
6.45.3.6 removeObject() [2/2]	167
6.45.3.7 removeObjects() [1/2]	168
6.45.3.8 removeObjects() [2/2]	169
6.45.3.9 setName()	169
6.46 Whiskas::Quarry Class Reference	170
6.46.1 Detailed Description	171
6.46.2 Member Function Documentation	171
6.46.2.1 getType()	172
6.47 Whiskas::RangedAltar Class Reference	172
6.47.1 Detailed Description	173
6.47.2 Member Function Documentation	173
6.47.2.1 getType()	174
6.48 Whiskas::RangedChampion Class Reference	174
6.48.1 Detailed Description	175
6.48.2 Member Function Documentation	175
6.48.2.1 getType()	175
6.49 Resourcemap_operationsTest Class Reference	176
6.50 Whiskas::Sawmill Class Reference	176
6.50.1 Detailed Description	178
6.50.2 Member Function Documentation	178
6.50.2.1 getType()	179
6.51 Course::SimpleGameScene Class Reference	179
6.51.1 Detailed Description	180
6.51.2 Constructor & Destructor Documentation	180
6.51.2.1 SimpleGameScene()	180
6.51.3 Member Function Documentation	181
6.51.3.1 drawItem()	181
6.51.3.2 event()	181
6.51.3.3 getScale()	182
6.51.3.4 getSize()	182
6.51.3.5 removeItem()	182
6.51.3.6 setScale()	183
6.51.3.7 setSize()	184
6.51.3.8 updateItem()	185
6.52 Course::SimpleMapItem Class Reference	186
6.52.1 Detailed Description	187

6.52.2 Constructor & Destructor Documentation	187
6.52.2.1 SimpleMapItem()	187
6.52.3 Member Function Documentation	187
6.52.3.1 boundingRect()	188
6.52.3.2 getBoundObject()	188
6.52.3.3 getSize()	188
6.52.3.4 isSameObj()	188
6.52.3.5 paint()	189
6.52.3.6 setSize()	190
6.53 SizeDialog Class Reference	190
6.53.1 Detailed Description	191
6.54 Whiskas::Spring Class Reference	192
6.54.1 Detailed Description	193
6.54.2 Member Function Documentation	193
6.54.2.1 addBuilding()	193
6.54.2.2 getType()	194
6.55 startdialog Class Reference	194
6.55.1 Detailed Description	195
6.56 testHandler Class Reference	195
6.57 testManager Class Reference	196
6.58 Course::TileBase Class Reference	196
6.58.1 Detailed Description	198
6.58.2 Constructor & Destructor Documentation	198
6.58.2.1 TileBase()	199
6.58.3 Member Function Documentation	199
6.58.3.1 addBuilding()	199
6.58.3.2 addWorker()	200
6.58.3.3 generateResources()	201
6.58.3.4 getBuildingCount()	202
6.58.3.5 getBuildings()	203
6.58.3.6 getType()	203
6.58.3.7 getWorkerCount()	204
6.58.3.8 getWorkers()	204
6.58.3.9 hasSpaceForBuildings()	204
6.58.3.10 hasSpaceForWorkers()	205
6.58.3.11 removeBuilding()	206
6.58.3.12 removeWorker()	206
6.59 Whiskas::Turn Class Reference	207
6.59.1 Detailed Description	208
6.59.2 Constructor & Destructor Documentation	208
6.59.2.1 Turn()	208
6.59.3 Member Function Documentation	208

6.59.3.1 <code>getInTurn()</code>	208
6.59.3.2 <code>getTurnCounter()</code>	208
6.59.3.3 <code>setInTurn()</code>	208
6.60 <code>Whiskas::Unit</code> Class Reference	209
6.60.1 Detailed Description	209
6.60.2 Member Function Documentation	210
6.60.2.1 <code>getMoveValue()</code>	210
6.60.2.2 <code>setMoved()</code>	210
6.61 <code>Course::WorkerBase</code> Class Reference	210
6.61.1 Detailed Description	212
6.61.2 Member Function Documentation	212
6.61.2.1 <code>canBePlacedOnTile()</code>	212
6.61.2.2 <code>doSpecialAction()</code>	213
6.61.2.3 <code>getResourceFocus()</code>	214
6.61.2.4 <code>getType()</code>	214
6.61.2.5 <code>setResourceFocus()</code>	214
6.61.2.6 <code>tileWorkAction()</code>	215
6.62 <code>Course::WorldGenerator</code> Class Reference	215
6.62.1 Detailed Description	216
6.62.2 Member Function Documentation	216
6.62.2.1 <code>addConstructor()</code>	216
6.62.2.2 <code>generateMap()</code>	216
6.62.2.3 <code>getInstance()</code>	217
Index	219

Chapter 1

Documentation

1.1 Overview

The bronze adventures game is a two player game. Players both control Nexuses, which the opposing player must destroy. The players construct buildings using resources. The buildings can either produce more resources, or upgrade the players minions. What are minions you i hear you ask? The minions are movable units that can fight the enemy minions, or destroy enemy buildings!

1.2 Instructions

This section contains information on moving, building and attacking.

1.2.1 Moving

To move a friendly minion, first select it with left click of your mouse. This will popup some text on the right side of the screen, detailing the minion in question. Once you have selected a minion, simply right click on your mouse where you want to go! The destination must be within the minions move value, (Diagonal movement costs 2) and must not contain minions or enemy buildings. Otherwise the move command might fail, or you might attack the enemy!

1.2.2 Attacking

If you wish to attack an enemy minion or building, try to move on top of it. (Movement restrictions apply). If the requirements are met, your unit will deal its attack damage to the opposing unit. Beware, some units have special attacks that can have negative effects for your own units! Some units also take less damage from attacks. If a units health drops down to 0, it will be destroyed and removed from the game.

1.2.3 Building

To build a building, first select "Add building". This will open up a dialog to select the different type of buildings. Select the one you wish to construct. A text label will now appear on the right side of your screen. It will detail the cost, production and placement rules of the building. If you are satisfied with your selection, left click on the tile you wish to place your building. Then select confirm build action button. Assuming you had the required resources, and tile placement, the building will now be constructed. Take note however, that only 1 building can be placed on a given tile.

1.2.4 Altars

You may have noticed special buildings called Altars. These can be constructed with the same rules as normal buildings, but they have a special action at the end of the turn. If a minion ends its turn under an altar, it will be upgraded to a champion. Melee champions are tanky, ranged champions are fast, and magic champions have a special Area of effect attack. This upgrade costs no resources. After an upgrade, the altar will go on cooldown for a set amount of turns. After the cooldown has expired, the altar can upgrade another minion.

1.2.5 Winning

To win the game, simply destroy the enemy Nexus to instantly win the game, and be declared master of all bronze players! (So silver)

1.3 distribution

Safwaneb was in charge of all things graphical. All pictures, and drawing objects/tiles were his responsibility. Several buildings are also made by Safwaneb. Jussi was in charge of the game mechanics. Moving and attacking and the minion, champions and altars were his doing.

1.4 content

- Moving
- Attacking
- Special Altar buildings
- Music (Currently only on Windows)

Chapter 2

Programming 3 Exercise project - CourseLibrary

This repository contains course-side code for the Programming 3 Exercise project.

You can report bugs and request features: <https://course-gitlab.tuni.fi/tie-02402-ohj3-2019-2020/courselibrary/issues>

2.1 Documentation

You can generate the class diagram with doxygen and view it by opening Doxy-documentation/html/index.html Note that the index-page contains a copy of <https://plus.tuni.fi/tie-0240x/autumn-2019/modules-00/harjoitusty%C3%B6/> so some elements aren't visible if you aren't logged in to plussa.

Updated 18.9.2019

Chapter 3

The Bronze Adventures game!

ALL DOCUMENTATION CAN BE FOUND IN THE FOLDER DOCUMENTATION! Meant to be read in html format, but a pdf is also included

3.1 Aditonal note:

Music is currently disabled. It can be enabled by removing comment markers found in mapwindow.cc constructor. It is clearly marked. It is disapled to ensure compatibility between windwows and the Uni Tiger vnc linux desktop. The music should work on windows, and on linux machines that support QT multimedia.

Chapter 4

Hierarchical Index

4.1 Class Hierarchy

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

Whiskas::Attackable	15
Whiskas::CustomBuildingBase	48
Whiskas::AltarBase	13
Whiskas::MageAltar	126
Whiskas::MeleeAltar	138
Whiskas::RangedAltar	172
Whiskas::Lifepump	123
Whiskas::Nexus	148
Whiskas::Quarry	170
Whiskas::Sawmill	176
Whiskas::Minion	143
Whiskas::MagicChampion	128
Whiskas::MeleeChampion	141
Whiskas::RangedChampion	174
Course::Coordinate	32
exception	
Course::BaseException	19
Course::IllegalAction	112
Course::NotEnoughSpace	150
Course::OwnerConflict	156
Course::InvalidPointer	113
Course::KeyError	120
Course::GameObject	81
Course::PlaceableGameObject	158
Course::BuildingBase	25
Course::Farm	56
Course::HeadQuarters	105
Course::Outpost	152
Whiskas::CustomBuildingBase	48
Course::WorkerBase	210
Course::BasicWorker	21
Whiskas::Minion	143
Course::TileBase	196
Course::Forest	59

Course::Grassland	102
Whiskas::Desert	52
Whiskas::Jungle	118
Whiskas::Mountain	145
Whiskas::Spring	192
Course::iGameEventHandler	109
Whiskas::gameEventHandler	62
Course::iObjectManager	115
Whiskas::gameManager	68
Course::PlayerBase	164
Whiskas::LeaguePlayer	121
QDialog	
buildingDialog	31
endDialog	55
SizeDialog	190
startdialog	194
QGraphicsItem	
Course::SimpleMapItem	186
Whiskas::MapItem	130
QGraphicsScene	
Course::SimpleGameScene	179
Whiskas::GameScene	95
QMainWindow	
MapWindow	134
QObject	
default_coordinate	49
default_gameobjects	50
default_playerbase	51
default_test	52
ManagerTest	130
Resourcemap_operationsTest	176
testHandler	195
testManager	196
Whiskas::Turn	207
Whiskas::Unit	209
Whiskas::Minion	143
Course::WorldGenerator	215

Chapter 5

Class Index

5.1 Class List

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

Whiskas::AltarBase	Inheritacne class for altar buildings	13
Whiskas::Attackable	Information regarding attacking and dealing damage	15
Course::BaseException	The Exception class is a base-class for custom exceptions in project	19
Course::BasicWorker	"normal worker" in the game	21
Course::BuildingBase	Base-class for different buildings in the game	25
buildingDialog	Dialog for selecting the building to be built	31
Course::Coordinate	32
Whiskas::CustomBuildingBase	48
default_coordinate	49
default_gameobjects	50
default_playerbase	51
default_test	52
Whiskas::Desert	Desert in the gameworld. The desert supports melee altars	52
endDialog	Shows the end message	55
Course::Farm	Farm-building in the game	56
Course::Forest	Forest in the gameworld	59
Whiskas::gameEventHandler	Handles events the game can have. Most notably, the mouse presses. NOTE: The first few methods that have unused variables are there for the reason that course side documentation said that those are required by the course side code. We dont use them but we didnt know if we could remove them	62
Whiskas::gameManager	Class that manages the game logic. It has access to almost all the classes included in this game	68
Course::GameObject	Base-class that contain's general information on different Objects in game	81

Whiskas::GameScene	95
Course::Grassland	
Grassland in the gameworld	102
Course::HeadQuarters	
Player's HeadQuarters-building	105
Course::iGameEventHandler	
Interface which the Course-side code uses to interact with the GameEventHandler implemented by the students	109
Course::IllegalAction	
The <code>IllegalAction</code> exception is usually used in cases, where an illegal game action was attempted	112
Course::InvalidPointer	
The <code>InvalidPointer</code> exception is usually used in cases, where data can't be accessed through a pointer	113
Course::iObjectManager	
Interface which the Course-side code uses to interact with the ObjectManager implemented by the students	115
Whiskas::Jungle	
Jungle in the gameworld. The jungle supports ranged altars and sawmills	118
Course::KeyError	
Exception-class for cases where the used key is invalid	120
Whiskas::LeaguePlayer	
The playerClass for this game	121
Whiskas::LifePump	
LifePump-building in the game	123
Whiskas::MageAltar	
Altar, that upgrades minions to mages	126
Whiskas::MagicChampion	
Subminion, has a higher dmg and an area attack	128
ManagerTest	
Whiskas::MapItem	
Custom QGraphicsItem that acts as a single GameObject's graphical element	130
MapWindow	
Manages the UI interface	134
Whiskas::MeleeAltar	
Altar that upgrades minions to melee champs	138
Whiskas::MeleeChampion	
Minion subtype. Has a melee attack of higher dmg and better defenses	141
Whiskas::Minion	
Subtype of worker It can be upgraded to champions using altars Is attackable and a unit (so it can attack and move)	143
Whiskas::Mountain	
Mountain in the gameworld. The mountain supports melee altars and quarries	145
Whiskas::Nexus	
Nexus-building in the game	148
Course::NotEnoughSpace	
Exception-class for errors where GameObjects are being placed onto Tiles with no space available	150
Course::Outpost	
Player's Outpost-building	152
Course::OwnerConflict	
Exception-class for errors where an operation is conflicting with a <code>GameObject</code> 's ownership	156
Course::PlaceableGameObject	
GameObjects that can be placed on Tile Objects	158
Course::PlayerBase	
Base class for classes used to describe a player in game	164
Whiskas::Quarry	
Quarry-building in the game	170

Whiskas::RangedAltar	Altar that upgrades minions to ranged units	172
Whiskas::RangedChampion	Has a ranged attack and higher movement	174
Resourcemap_operationsTest	176
Whiskas::Sawmill	Sawmill-building in the game	176
Course::SimpleGameScene	The SimpleGameScene is a custom QGraphicsScene that shows a simple rendering of the game map	179
Course::SimpleMapItem	Custom QGraphicsItem that acts as a single GameObject 's graphical element	186
SizeDialog	Sets the mapSize	190
Whiskas::Spring	Spring in the gameworld. The spring supports mage altars and lifepumps	192
startdialog	Selector dialog for the game start	194
testHandler	195
testManager	196
Course::TileBase	Base-class for different Tile-objects in the game. 196	
Whiskas::Turn	Handles the turn event. Updates the player inventories, resets attacked and moved for units . . .	207
Whiskas::Unit	Handles the move values	209
Course::WorkerBase	Abstract base-class for Worker-objects	210
Course::WorldGenerator	Singleton for generating tiles for the game	215

Chapter 6

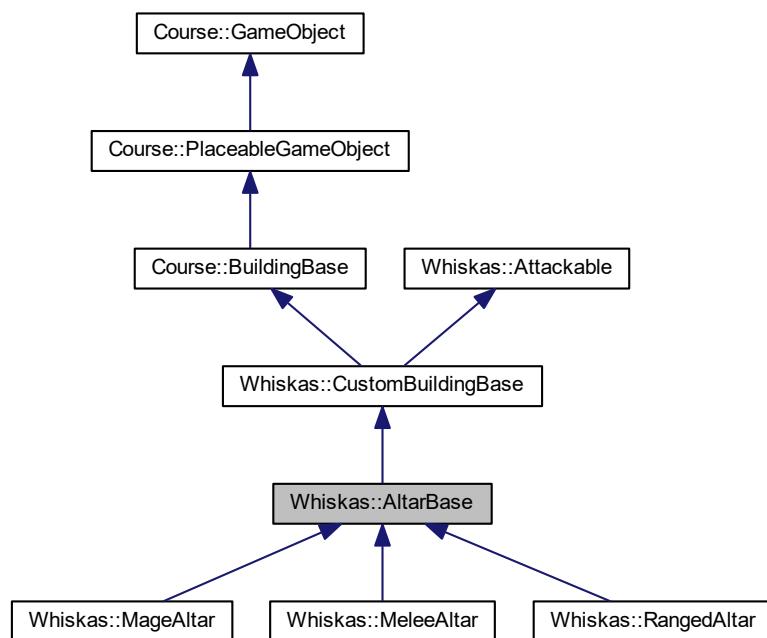
Class Documentation

6.1 Whiskas::AltarBase Class Reference

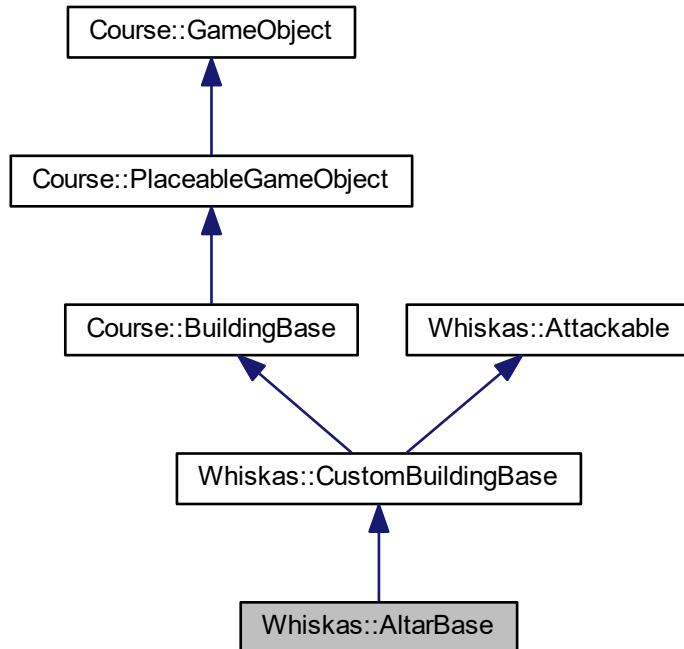
The [AltarBase](#) class is the inheritance class for altar buildings.

```
#include <AltarBase.h>
```

Inheritance diagram for Whiskas::AltarBase:



Collaboration diagram for Whiskas::AltarBase:



Public Member Functions

- `AltarBase (const std::shared_ptr< gameEventHandler > &eventhandler, const std::shared_ptr< gameManager > &objectmanager, const std::shared_ptr< Course::PlayerBase > &owner, const int &tilespaces=1, const AdvancedResourceMap &buildcost={}, const AdvancedResourceMap &production={}, int health=5, int attack=0)`
- `std::string getType () const override`
- `virtual void upgradeMinion ()=0`
upgradeMinion upgrades a minion on the tile when called.
- `void doSpecialAction () override`
doSpecialAction calls upgrade minion, since all buildings have this method

Protected Attributes

- `std::shared_ptr< gameManager > manager_`
- `std::shared_ptr< gameEventHandler > handler_`
- `std::shared_ptr< Course::PlayerBase > owner_`

Additional Inherited Members

6.1.1 Detailed Description

The `AltarBase` class is the inheritance class for altar buildings.

6.1.2 Member Function Documentation

6.1.2.1 `getType()`

```
std::string Whiskas::AltarBase::getType ( ) const [override], [virtual]
```

Reimplemented from [Course::BuildingBase](#).

Reimplemented in [Whiskas::MeleeAltar](#), [Whiskas::MageAltar](#), and [Whiskas::RangedAltar](#).

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

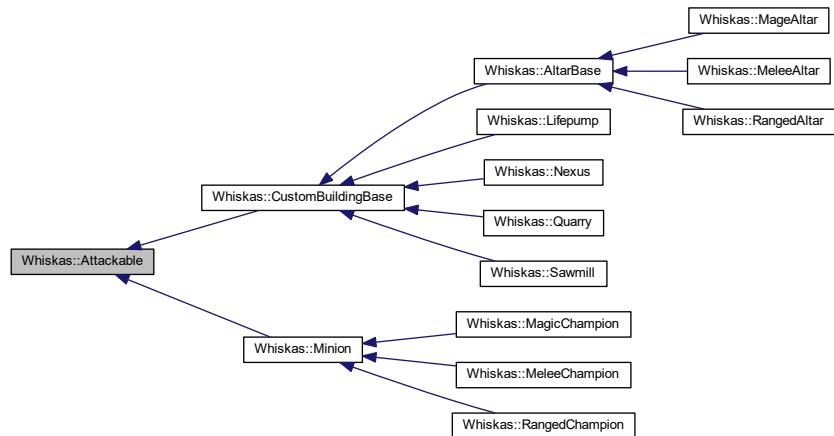
- E:/Gitin repo/ohjelmointi 3/whiskas/Game/buildings/AltarBase.h
- E:/Gitin repo/ohjelmointi 3/whiskas/Game/buildings/AltarBase.cpp

6.2 Whiskas::Attackable Class Reference

The [Attackable](#) class contains information regarding attacking and dealing damage.

```
#include <attackable.h>
```

Inheritance diagram for Whiskas::Attackable:



Public Member Functions

- [Attackable](#) (int health, int attack, int numberOfAttacks, int boundID)
Attackable Constructor.
- int [getHealth](#) ()
getHealth
- int [getAttack](#) ()
getAttack
- int [getBoundID](#) ()
getBoundID
- void [modifyAttack](#) (int AModifier)
Increases the damage of the unit by the modifier.
- bool [getAttacked](#) ()
getAttacked
- virtual bool [modifyHealth](#) (int hModifier)
modifyHealth
- virtual void [setAttacked](#) (bool hasAttacked)
setAttacked tries to modify the boolean HasAttacked_;

Protected Attributes

- int **healthValue_**
- int **attackValue_**
- int **boundID_**
- int **numberOfAttacks_**
- int **numberOfAttacksDone_** =0
- bool **HasAttacked_** =false

6.2.1 Detailed Description

The [Attackable](#) class contains information regarding attacking and dealing damage.

6.2.2 Constructor & Destructor Documentation

6.2.2.1 [Attackable\(\)](#)

```
Whiskas::Attackable::Attackable (
    int health,
    int attack,
    int numberOfAttacks,
    int boundID )
```

[Attackable](#) Constructor.

Parameters

<i>health</i>	0=Dead
<i>attack</i>	damage
<i>numberOfAttacks</i>	how many times the unit can attack in a turn
<i>boundID</i>	id of the object this is bound to

6.2.3 Member Function Documentation

6.2.3.1 getAttack()

```
int Whiskas::Attackable::getAttack ( )
```

getAttack

Returns

AttackValue

6.2.3.2 getAttacked()

```
bool Whiskas::Attackable::getAttacked ( )
```

getAttacked

Returns

true if he unit has attacked this turn. False otherwise

6.2.3.3 getBoundID()

```
int Whiskas::Attackable::getBoundID ( )
```

getBoundID

Returns

ID of the object this is bound to

6.2.3.4 `getHealth()`

```
int Whiskas::Attackable::getHealth ( )
```

getHealth

Returns

Current HP

6.2.3.5 `modifyAttack()`

```
void Whiskas::Attackable::modifyAttack ( int AModifier )
```

Increases the damage of the unit by the modifier.

Parameters

<i>AModifier</i>

6.2.3.6 `modifyHealth()`

```
bool Whiskas::Attackable::modifyHealth ( int hModifier ) [virtual]
```

modifyHealth

Parameters

<i>hModifier</i>

Returns

true if dead, false if alive

Reimplemented in [Whiskas::Minion](#), and [Whiskas::MeleeChampion](#).

6.2.3.7 `setAttacked()`

```
void Whiskas::Attackable::setAttacked ( bool hasAttacked ) [virtual]
```

setAttacked tries to modify the boolean HasAttacked_;

Parameters

hasAttacked	
-------------	--

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

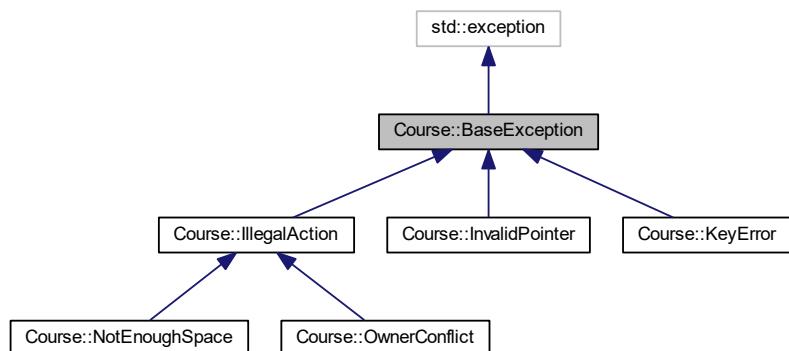
- E:/Gitin repo/ohjelmointi 3/whiskas/Game/minion/attackable.h
- E:/Gitin repo/ohjelmointi 3/whiskas/Game/minion/attackable.cpp

6.3 Course::BaseException Class Reference

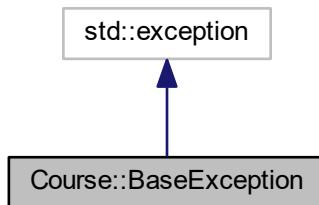
The Exception class is a base-class for custom exceptions in project.

```
#include <baseexception.h>
```

Inheritance diagram for Course::BaseException:



Collaboration diagram for Course::BaseException:



Public Member Functions

- **BaseException** (const std::string &**msg**= "")
Exception Constructor.
- virtual ~**BaseException** ()=default
~Exception Default destructor
- virtual std::string **msg** () const
msg Returns the message being stored in Exception.

6.3.1 Detailed Description

The Exception class is a base-class for custom exceptions in project.

6.3.2 Constructor & Destructor Documentation

6.3.2.1 BaseException()

```
Course::BaseException::BaseException (
    const std::string & msg = "" ) [inline], [explicit]
```

Exception Constructor.

Parameters

msg	std::string describing the reason for exception.
------------	--

6.3.3 Member Function Documentation

6.3.3.1 msg()

```
virtual std::string Course::BaseException::msg ( ) const [inline], [virtual]
```

msg Returns the message being stored in Exception.

Returns

Stored message or empty string.

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

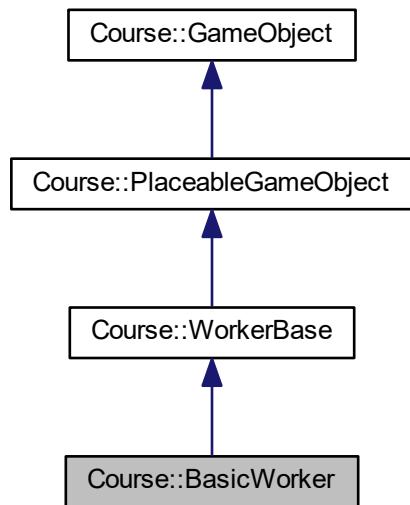
- E:/Gitin repo/ohjelmointi 3/whiskas/Course/CourseLib/exceptions/baseexception.h

6.4 Course::BasicWorker Class Reference

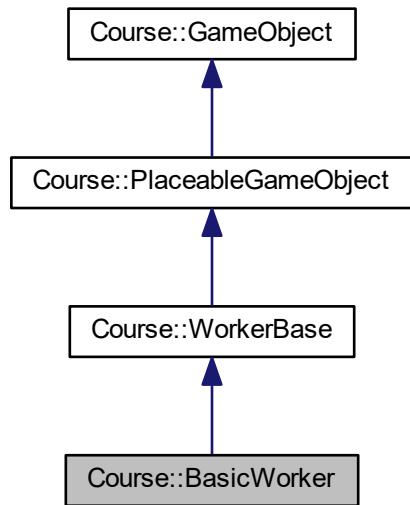
The [BasicWorker](#) class represents a "normal worker" in the game.

```
#include <basicworker.h>
```

Inheritance diagram for Course::BasicWorker:



Collaboration diagram for Course::BasicWorker:



Public Member Functions

- `BasicWorker ()=delete`
Disabled parameterless constructor.
- `BasicWorker (const std::shared_ptr< iGameEventHandler > &eventhandler, const std::shared_ptr< iObjectManager > &objectmanager, const std::shared_ptr< PlayerBase > &owner, const int &tilespaces=1, const ResourceMap &cost=ConstResourceMaps::BW_RECRUITMENT_COST, const ResourceMapDouble &efficiency=ConstResourceMaps::BW_WORKER_EFFICIENCY)`
Constructor for the class.
- `virtual ~BasicWorker ()=default`
Default destructor.
- `virtual std::string getType () const override`
getType Returns a string describing objects type. This should be overriden in each inherited class. Makes checking object's type easier for students.
- `virtual bool canBePlacedOnTile (const std::shared_ptr< TileBase > &target) const override`
Check if the worker can be placed on the Tile according to it's placement rule. Only rule is that the Tile must have same owner as the worker.
- `virtual void doSpecialAction () override`
Performs the Worker's default action.
- `virtual const ResourceMapDouble tileWorkAction () override`
Returns Worker's efficiency at resource production. Worker consumes FOOD and MONEY. Resource consumption and resource focus determine final multiplier that is based on WORKER_EFFICIENCY.

Additional Inherited Members

6.4.1 Detailed Description

The `BasicWorker` class represents a "normal worker" in the game.

Worker has following production-efficiency:

- Money - 0.25
- Food - 1.00
- Wood - 0.75
- Stone - 0.50
- Ore - 0.50
 BasicWorkers consume Food and money.
- 1 Food - Or `BasicWorker` refuses to work.
- 1 Money - Or `BasicWorker` works at 50% efficiency.
- Resourcefocus adds 25% efficiency for the focused resource, even if the worker is refusing work.

6.4.2 Constructor & Destructor Documentation

6.4.2.1 BasicWorker()

```
Course::BasicWorker::BasicWorker (
    const std::shared_ptr< iGameEventHandler > & eventhandler,
    const std::shared_ptr< iObjectManager > & objectmanager,
    const std::shared_ptr< PlayerBase > & owner,
    const int & tilespaces = 1,
    const ResourceMap & cost = ConstResourceMaps::BW_RECRUITMENT_COST,
    const ResourceMapDouble & efficiency = ConstResourceMaps::BW_WORKER_EFFICIENCY )
```

Constructor for the class.

Parameters

<i>eventhandler</i>	points to the student's GameEventHandler.
<i>owner</i>	points to the owning player.
<i>descriptions</i>	contains descriptions and flavor texts.

6.4.3 Member Function Documentation

6.4.3.1 canBePlacedOnTile()

```
bool Course::BasicWorker::canBePlacedOnTile (
    const std::shared_ptr< TileBase > & target ) const [override], [virtual]
```

Check if the worker can be placed on the Tile according to it's placement rule. Only rule is that the Tile must have same owner as the worker.

Parameters

<i>target</i>	is the Tile that worker is being placed on.
---------------	---

Returns

True - If baseclass' method return true and target Tile has space for worker. False - If both conditions aren't met.

Note

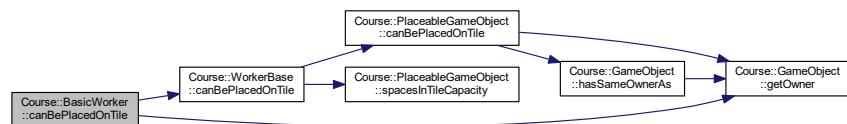
Override to change placement rules for derived worker.

Postcondition

Exception guarantee: Basic

Reimplemented from [Course::PlaceableGameObject](#).

Here is the call graph for this function:

**6.4.3.2 getType()**

```
std::string Course::BasicWorker::getType ( ) const [override], [virtual]
```

getType Returns a string describing objects type. This should be overriden in each inherited class. Makes checking object's type easier for students.

Returns

std::string that represents Object's type.

Postcondition

Exception guarantee: No-throw

Note

You can use this in e.g. debugging and similar printing.

Reimplemented from [Course::PlaceableGameObject](#).

6.4.3.3 tileWorkAction()

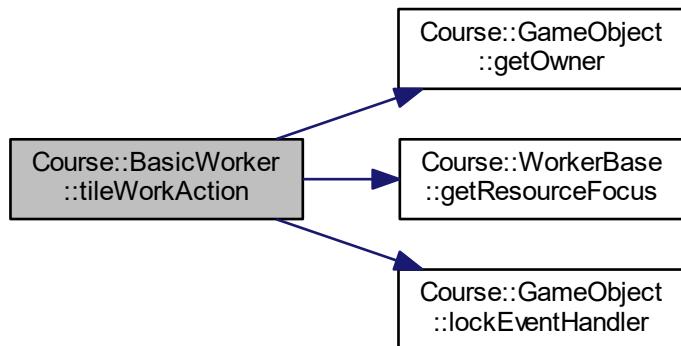
```
const ResourceMapDouble Course::BasicWorker::tileWorkAction ( ) [override], [virtual]
```

Returns Worker's efficiency at resource production. Worker consumes FOOD and MONEY. Resource consumption and resource focus determine final multiplier that is based on WORKER_EFFICIENCY.

Returns

Reimplemented from [Course::WorkerBase](#).

Here is the call graph for this function:



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

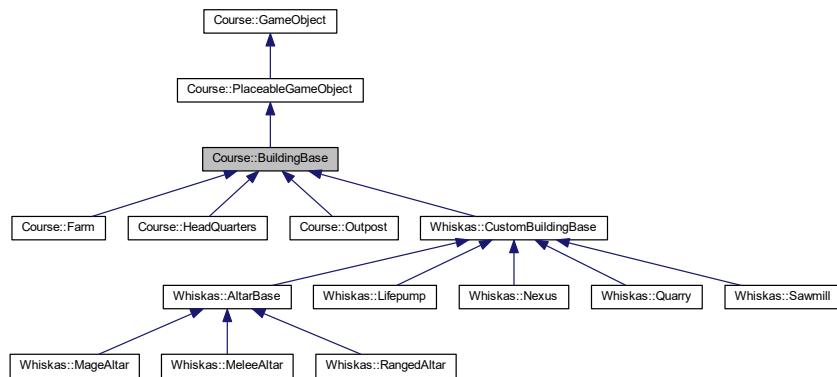
- E:/Gitin repo/ohjelmointi 3/whiskas/Course/CourseLib/workers/basicworker.h
- E:/Gitin repo/ohjelmointi 3/whiskas/Course/CourseLib/workers/basicworker.cpp

6.5 Course::BuildingBase Class Reference

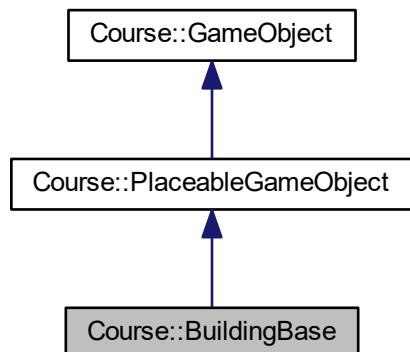
The [BuildingBase](#) class is a base-class for different buildings in the game.

```
#include <buildingbase.h>
```

Inheritance diagram for Course::BuildingBase:



Collaboration diagram for Course::BuildingBase:



Public Member Functions

- **BuildingBase ()=delete**
Disabled parameterless constructor.
- **BuildingBase (const std::shared_ptr< iGameEventHandler > &eventhandler, const std::shared_ptr< iObjectManager > &objectmanager, const std::shared_ptr< PlayerBase > &owner, const int &tilespaces=1, const ResourceMap &buildcost={}, const ResourceMap &production={})**
Constructor for the class.
- **~BuildingBase () override=default**
Default destructor.
- **std::string getType () const override**
getType Returns a string describing objects type. This should be overriden in each inherited class. Makes checking object's type easier for students.
- **virtual void doSpecialAction ()**

- `virtual void onBuildAction ()`
Performs building's default action.
- `virtual ResourceMap getProduction ()`
Performs building's possible action after construction.
- `virtual void addHoldMarkers (int amount) final`
Adds the amount to hold-markers.
- `virtual int holdCount () const final`
Returns the amount of hold-markers.
- `bool canBePlacedOnTile (const std::shared_ptr< TileBase > &target) const override`
Returns boolean based on wheter the building can or can't be placed on a Tile-object.

Public Attributes

- `const ResourceMap BUILD_COST`
- `const ResourceMap PRODUCTION_EFFECT`

Additional Inherited Members

6.5.1 Detailed Description

The `BuildingBase` class is a base-class for different buildings in the game.

- Can increase base-production for a Tile.
- Can call functions from GameEventHandler.

Buildings can have hold-markers that prevent normal operation.

6.5.2 Constructor & Destructor Documentation

6.5.2.1 BuildingBase()

```
Course::BuildingBase::BuildingBase (
    const std::shared_ptr< iGameEventHandler > & eventhandler,
    const std::shared_ptr< iObjectManager > & objectmanager,
    const std::shared_ptr< PlayerBase > & owner,
    const int & tilespaces = 1,
    const ResourceMap & buildcost = {},
    const ResourceMap & production = {} ) [explicit]
```

Constructor for the class.

Parameters

<i>eventhandler</i>	points to the student's GameEventHandler.
<i>owner</i>	points to the owning player.
<i>descriptions</i>	contains descriptions and flavor texts.
<i>tile</i>	points to the tile upon which the building is constructed.
<i>hold</i>	is the initial amount of hold-markers.

Postcondition

Exception guarantee: No guarantee.

Exceptions

<i>OwnerConflict</i>	- if the building conflicts with tile's ownership.
----------------------	--

6.5.3 Member Function Documentation

6.5.3.1 addHoldMarkers()

```
void Course::BuildingBase::addHoldMarkers (
    int amount ) [final], [virtual]
```

Adds the amount to hold-markers.

Note

Negative amounts can be used for subtraction.

Parameters

<i>amount</i>	the amount being added.
---------------	-------------------------

Postcondition

Exception guarantee: No-throw

6.5.3.2 canBePlacedOnTile()

```
bool Course::BuildingBase::canBePlacedOnTile (
    const std::shared_ptr< TileBase > & target ) const [override], [virtual]
```

Returns boolean based on wheter the building can or can't be placed on a Tile-object.

Parameters

<i>target</i>	is a pointer to the target Tile.
---------------	----------------------------------

Returns

True - Base class' method return true and Tile has space for building. False - If both conditions are not met.

Note

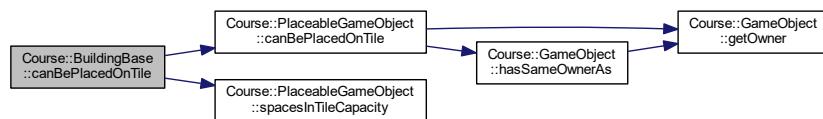
Override to modify placementrules for derived classes.

Postcondition

Exception guarantee: Basic

Reimplemented from [Course::PlaceableGameObject](#).

Here is the call graph for this function:

**6.5.3.3 getProduction()**

```
ResourceMap Course::BuildingBase::getProduction ( ) [virtual]
```

Return's building's production as a ResourceMap.

Note

Used by [TileBase](#) to get buildings production effect on Tile's production.

Reimplemented in [Course::Outpost](#).

6.5.3.4 `getType()`

```
std::string Course::BuildingBase::getType ( ) const [override], [virtual]
```

`getType` Returns a string describing objects type. This should be overriden in each inherited class. Makes checking object's type easier for students.

Returns

`std::string` that represents Object's type.

Postcondition

Exception guarantee: No-throw

Note

You can use this in e.g. debugging and similar printing.

Reimplemented from [Course::GameObject](#).

Reimplemented in [Course::Outpost](#), [Course::HeadQuarters](#), [Course::Farm](#), [Whiskas::Nexus](#), [Whiskas::Quarry](#), [Whiskas::Sawmill](#), [Whiskas::Lifepump](#), [Whiskas::AltarBase](#), [Whiskas::MeleeAltar](#), [Whiskas::MageAltar](#), and [Whiskas::RangedAltar](#).

6.5.3.5 `holdCount()`

```
int Course::BuildingBase::holdCount ( ) const [final], [virtual]
```

Returns the amount of hold-markers.

Returns

value in `m_hold`.

Postcondition

Exception guarantee: No-throw

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

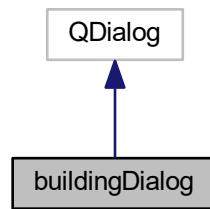
- E:/Gitin repo/ohjelmointi 3/whiskas/Course/CourseLib/buildings/buildingbase.h
- E:/Gitin repo/ohjelmointi 3/whiskas/Course/CourseLib/buildings/buildingbase.cpp

6.6 buildingDialog Class Reference

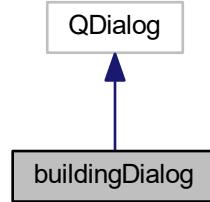
The [buildingDialog](#) class is a dialog for selecting the building to be built.

```
#include <buildingdialog.h>
```

Inheritance diagram for buildingDialog:



Collaboration diagram for buildingDialog:



Signals

- void [buildingType](#) (std::string)
buildingType send the buildingtype to MainWindow

Public Member Functions

- [buildingDialog](#) (QWidget *parent=nullptr)

6.6.1 Detailed Description

The `buildingDialog` class is a dialog for selecting the building to be built.

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

- E:/Gitin repo/ohjelmointi 3/whiskas/Game/dialogs/buildingdialog.h
- E:/Gitin repo/ohjelmointi 3/whiskas/Game/dialogs/buildingdialog.cpp

6.7 Course::Coordinate Class Reference

Public Member Functions

- `Coordinate (int x, int y)`
`Constructor for x, y coordinates.`
- `Coordinate (const Coordinate &original)`
`Copy-constructor.`
- `Coordinate (const Coordinate &original, Direction direction, int steps=1)`
`Copy-constructor where the new Coordinate is moved.`
- `~Coordinate ()=default`
`Default destructor.`
- `int x () const`
`x Returns the x-value`
- `int y () const`
`y Returns the y-value`
- `void set_x (int x)`
`set_x Sets the x-value`
- `void set_y (int y)`
`set_y Sets the y-value`
- `void travel (Direction direction, int steps=1)`
`travel Increments the coordinate values to given direction. Optionally can give how many increments are done.`
- `Coordinate neighbour_at (Direction direction, int steps=1) const`
`neighbour_at Returns a new Coordinate object instead of changing the current one`
- `std::vector< Coordinate > neighbours (const int &radius=1) const`
`neighbours Returns a Coordinate for each direction. The created vector has the Coordinates in order of the default direction values`
- `Coordinate operator+ (const Coordinate &other) const`
`operator + Sums two coordinates' x and y values together and returns a new Coordinate with the summed values.`
- `Coordinate & operator+= (const Coordinate &other)`
`operator += Adds other coordinate's x and y values to this.`
- `Coordinate operator- (const Coordinate &other) const`
`operator - Subtracts right side coordinate's x and y values from this x and y values. Returns a new Coordinate with new x and y values`
- `Coordinate & operator-= (const Coordinate &other)`
`operator -= Subtracts other coordinate's x and y values from this`
- `Coordinate & operator= (const Coordinate &other)`
`operator = Assigns other Coordinate's x and y values to this.`
- `bool operator== (const Coordinate &other) const`
`operator == Checks if a coordinate has same x and y values as this`

- bool `operator!=` (const `Coordinate` &other) const
`operator ==` Checks if a coordinate doesn't have same x and y values as this
- bool `operator<` (const `Coordinate` &other) const
`operator <` Checks if this has lower value than other. Decision is based on x-value. If the x-values are equal, decision is based on y-value
- bool `operator>` (const `Coordinate` &other) const
`operator <` Checks if this has higher value than other. Decision is based on x-value. If the x-values are equal, decision is based on y-value
- bool `operator<=` (const `Coordinate` &other) const
`operator <=` logical-OR with == and < operators
- bool `operator>=` (const `Coordinate` &other) const
`operator <=` logical-OR with == and > operators

6.7.1 Constructor & Destructor Documentation

6.7.1.1 Coordinate() [1/3]

```
Course::Coordinate::Coordinate (
    int x,
    int y )
```

Constructor for x, y coordinates.

Parameters

<code>x</code>	X-coordinate
<code>y</code>	Y-coordinate

6.7.1.2 Coordinate() [2/3]

```
Course::Coordinate::Coordinate (
    const Coordinate & original )
```

Copy-constructor.

Parameters

<code>original</code>	Reference to a <code>Coordinate</code> that is being copied
-----------------------	---

6.7.1.3 Coordinate() [3/3]

```
Course::Coordinate::Coordinate (
```

```
const Coordinate & original,
Direction direction,
int steps = 1 )
```

Copy-constructor where the new [Coordinate](#) is moved.

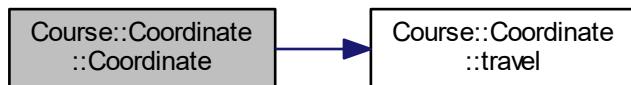
Parameters

<i>original</i>	Reference to Coordinate that is being copied
<i>direction</i>	What direction is the movement applied
<i>steps</i>	How many steps towards that direction is taken, default = 1

Postcondition

Exception guarantee: No-throw

Here is the call graph for this function:



6.7.1.4 ~Coordinate()

```
Course::Coordinate::~Coordinate () [default]
```

Default destructor.

Postcondition

Exception guarantee: Strong

6.7.2 Member Function Documentation

6.7.2.1 neighbour_at()

```
Coordinate Course::Coordinate::neighbour_at (
    Direction direction,
    int steps = 1 ) const
```

`neighbour_at` Returns a new [Coordinate](#) object instead of changing the current one

Parameters

<i>direction</i>	Direction in x,y -coordinate system
<i>steps</i>	How many increments of the values are done.

Returns

[Coordinate](#) object that was created

Postcondition

Exception guarantee: No-throw

6.7.2.2 neighbours()

```
std::vector< Coordinate > Course::Coordinate::neighbours (
    const int & radius = 1 ) const
```

neighbours Returns a [Coordinate](#) for each direction. The created vector has the Coordinates in order of the default direction values

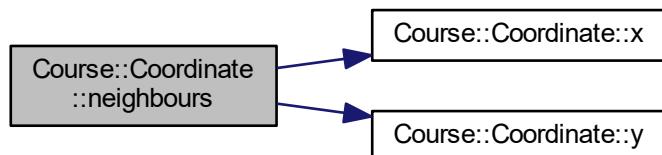
Returns

vector of Coordinates in the order of directions.

Postcondition

Exception guarantee: Strong

Here is the call graph for this function:



Here is the caller graph for this function:



6.7.2.3 operator"!=()

```
bool Course::Coordinate::operator!= (
    const Coordinate & other ) const
```

operator == Checks if a coordinate doesn't have same x and y values as this

Parameters

<i>other</i>	The other Coordinate
--------------	--------------------------------------

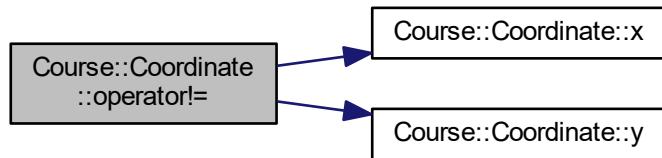
Returns

True - if the coordinates don't have same values.

Postcondition

Exception guarantee: No-throw

Here is the call graph for this function:



6.7.2.4 operator+()

```
Coordinate Course::Coordinate::operator+ (
    const Coordinate & other ) const
```

operator + Sums two coordinates' x and y values together and returns a new [Coordinate](#) with the summed values.

Parameters

<i>other</i>	The other Coordinate
--------------	--------------------------------------

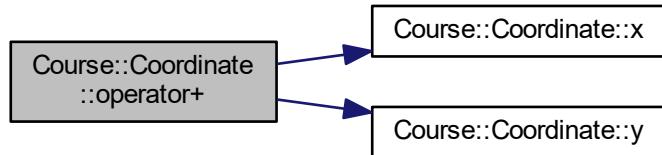
Returns

[Coordinate](#) with $x = x(\text{this}) + x(\text{other})$ and $y = y(\text{this}) + y(\text{other})$

Postcondition

Exception guarantee: No-throw

Here is the call graph for this function:



6.7.2.5 operator+=()

```
Coordinate & Course::Coordinate::operator+= ( const Coordinate & other )
```

`operator +=` Adds other coordinate's x and y values to this.

Parameters

<code>other</code>	The other coordinate
--------------------	----------------------

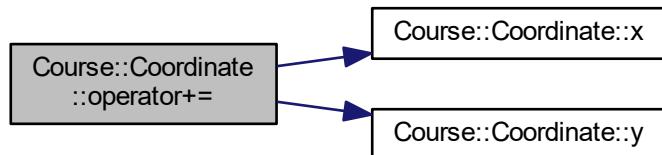
Returns

Reference to self.

Postcondition

Exception guarantee: No-throw

Here is the call graph for this function:



6.7.2.6 operator-()

```
Coordinate Course::Coordinate::operator- (
    const Coordinate & other ) const
```

operator - Subtracts right side coordinate's x and y values from this x and y values. Returns a new [Coordinate](#) with new x and y values

Parameters

<i>other</i>	The other coordinate
--------------	----------------------

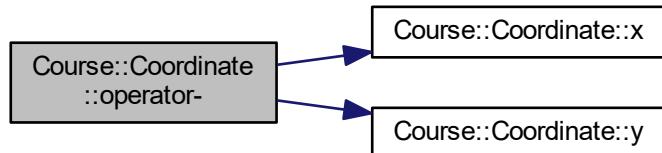
Returns

[Coordinate](#) with $x = x(\text{this}) - x(\text{other})$ and $y = y(\text{this}) - y(\text{other})$

Postcondition

Exception guarantee: No-throw

Here is the call graph for this function:



6.7.2.7 operator-=(())

```
Coordinate & Course::Coordinate::operator-= (
    const Coordinate & other )
```

operator -= Subtracts other coordinate's x and y values from this

Parameters

<i>other</i>	The other coordinate
--------------	----------------------

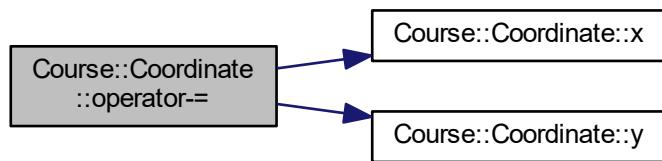
Returns

Reference to self.

Postcondition

Exception guarantee: No-throw

Here is the call graph for this function:



6.7.2.8 operator<()

```
bool Course::Coordinate::operator< (
    const Coordinate & other ) const
```

`operator <` Checks if this has lower value than `other`. Decision is based on `x`-value. If the `x`-values are equal, decision is based on `y`-value

Parameters

<code>other</code>	The other Coordinate
--------------------	--------------------------------------

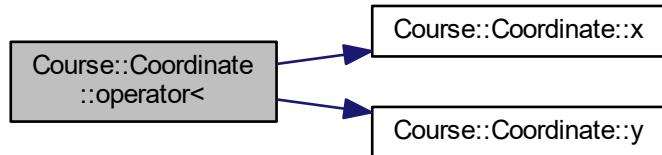
Returns

True - (`x(this) < x(other)`) or (`x(this) == x(other) + y(this) < y(other)`)

Postcondition

Exception guarantee: No-throw

Here is the call graph for this function:



6.7.2.9 operator<=()

```
bool Course::Coordinate::operator<= (
    const Coordinate & other ) const
```

operator <= logical-OR with == and < operators

Parameters

<i>other</i>	The other Coordinate
--------------	--------------------------------------

Postcondition

Exception guarantee: No-throw

6.7.2.10 operator=(())

```
Coordinate & Course::Coordinate::operator= (
    const Coordinate & other )
```

operator = Assigns other [Coordinate](#)'s x and y values to this.

Parameters

<i>other</i>	The other Coordinate
--------------	--------------------------------------

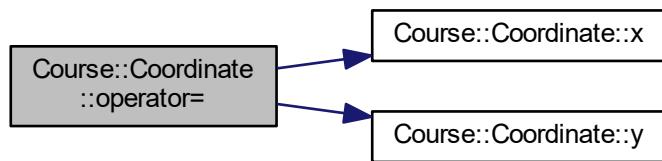
Returns

Reference to self.

Postcondition

Exception guarantee: No-throw

Here is the call graph for this function:



6.7.2.11 operator==()

```
bool Course::Coordinate::operator== (  
    const Coordinate & other ) const
```

operator == Checks if a coordinate has same x and y values as this

Parameters

<i>other</i>	The other Coordinate
--------------	--------------------------------------

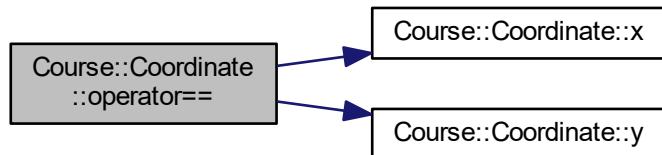
Returns

True - if the coordinates have same values.

Postcondition

Exception guarantee: No-throw

Here is the call graph for this function:

**6.7.2.12 operator>()**

```
bool Course::Coordinate::operator> (
    const Coordinate & other ) const
```

`operator <` Checks if this has higher value than other. Decision is based on x-value. If the x-values are equal, decision is based on y-value

Parameters

<code>other</code>	The other Coordinate
--------------------	--------------------------------------

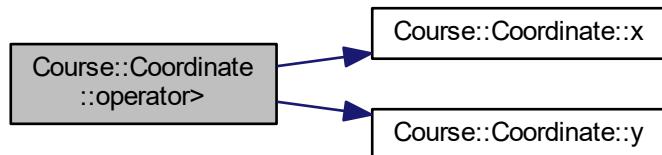
Returns

True - ($x(\text{this}) > x(\text{other})$) or ($x(\text{this}) == x(\text{other}) + y(\text{this}) > y(\text{other})$)

Postcondition

Exception guarantee: No-throw

Here is the call graph for this function:



6.7.2.13 operator>=()

```
bool Course::Coordinate::operator>= (
    const Coordinate & other ) const
```

operator <= logical-OR with == and > operators

Parameters

<i>other</i>	The other Coordinate
--------------	----------------------

Postcondition

Exception guarantee: No-throw

6.7.2.14 set_x()

```
void Course::Coordinate::set_x (
    int x )
```

set_x Sets the x-value

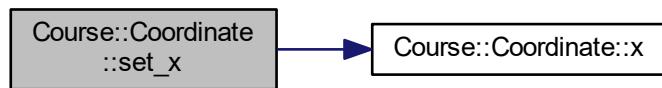
Parameters

<i>x</i>	New x-value
----------	-------------

Postcondition

Exception guarantee: No-throw

Here is the call graph for this function:



6.7.2.15 set_y()

```
void Course::Coordinate::set_y (
    int y )
```

`set_y` Sets the y-value

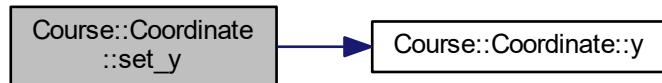
Parameters

<code>y</code>	New y-value
----------------	-------------

Postcondition

Exception guarantee: No-throw

Here is the call graph for this function:



6.7.2.16 travel()

```
void Course::Coordinate::travel (
    Direction direction,
    int steps = 1 )
```

`travel` Increments the coordinate values to given direction. Optionally can give how many increments are done.

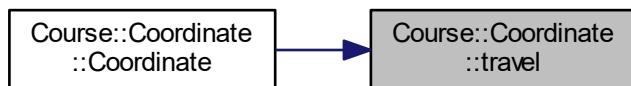
Parameters

<code>direction</code>	Direction in x,y -coordinate system
<code>steps</code>	How many increments of the values are done.

Postcondition

Exception guarantee: No-throw

Here is the caller graph for this function:

**6.7.2.17 x()**

```
int Course::Coordinate::x ( ) const
```

x Return's the x-value

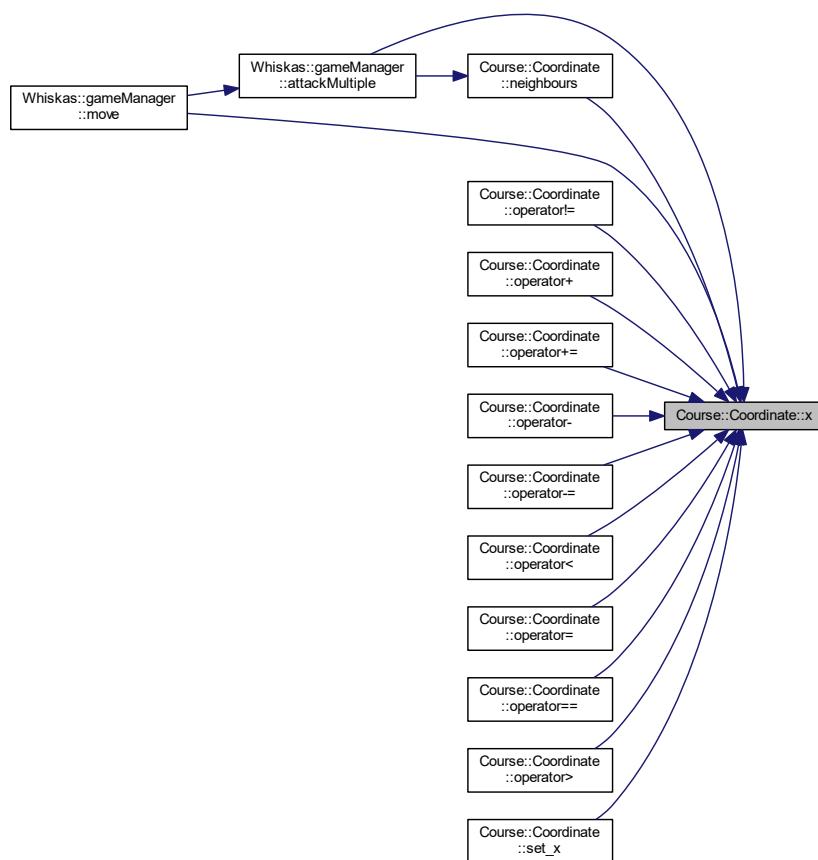
Returns

x-coordinate

Postcondition

Exception guarantee: No-throw

Here is the caller graph for this function:

**6.7.2.18 y()**

```
int Course::Coordinate::y ( ) const
```

`y` Return's the y-value

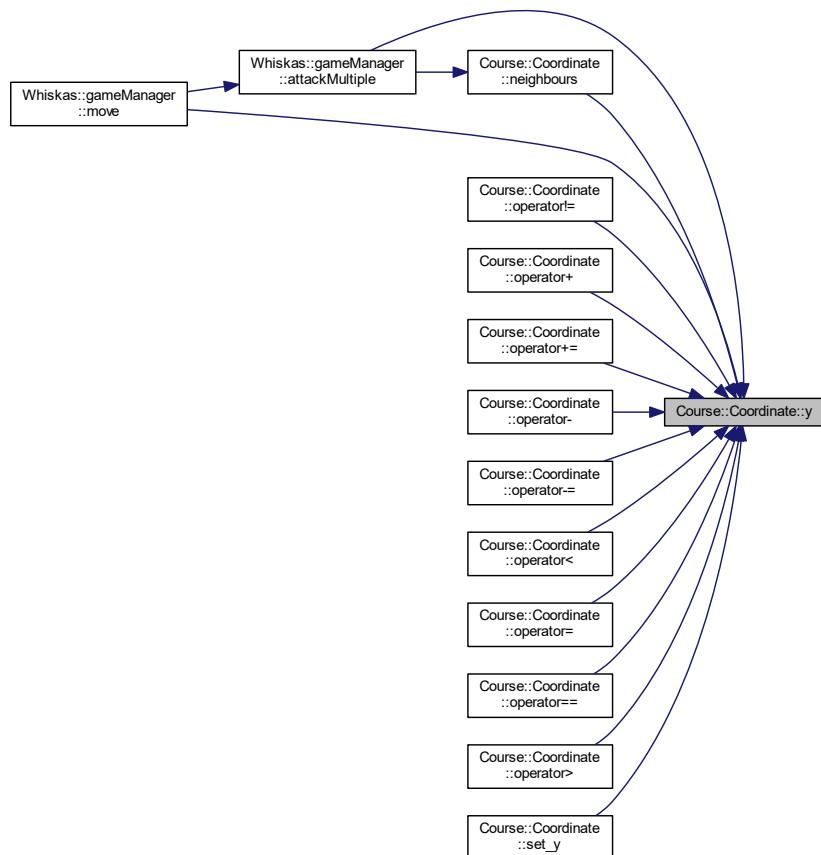
Returns

y-coordinate

Postcondition

Exception guarantee: No-throw

Here is the caller graph for this function:

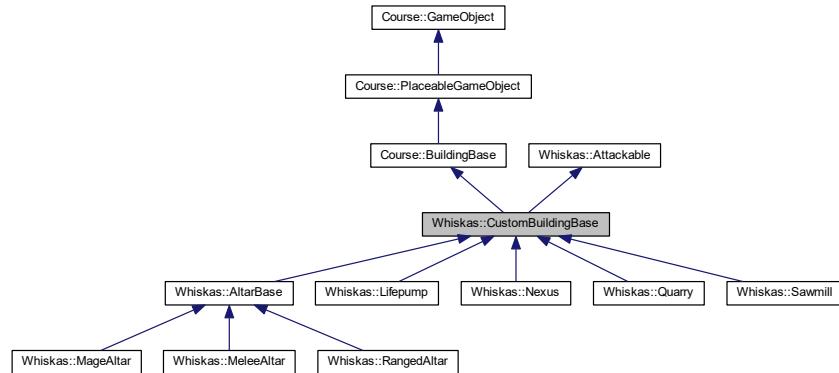


The documentation for this class was generated from the following files:

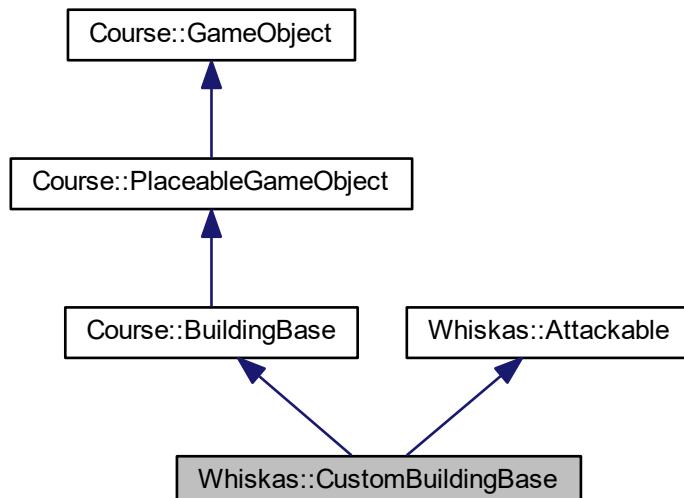
- E:/Gitin repo/ohjelmointi 3/whiskas/Course/CourseLib/core/coordinate.h
- E:/Gitin repo/ohjelmointi 3/whiskas/Course/CourseLib/core/coordinate.cpp

6.8 Whiskas::CustomBuildingBase Class Reference

Inheritance diagram for Whiskas::CustomBuildingBase:



Collaboration diagram for Whiskas::CustomBuildingBase:



Public Member Functions

- **CustomBuildingBase** (const std::shared_ptr< gameEventHandler > &eventhandler, const std::shared_ptr< gameManager > &objectmanager, const std::shared_ptr< Course::PlayerBase > &owner, const int &tilespaces=2, const AdvancedResourceMap &buildcost={}, const AdvancedResourceMap &production={}, int health=3, int attack=0)
- virtual AdvancedResourceMap **getAdvancedProduction** ()
Returns building's production as AdvancedResourceMap.

- virtual AdvancedResourceMap [getAdvancedCost \(\)](#)
Returns building's cost as AdvancedResourceMap.
- virtual int [getCooldown \(\)](#)
returns altar's cooldown

Protected Attributes

- AdvancedResourceMap **production_**
- AdvancedResourceMap **buildcost_**
- int **cooldown_** = -1

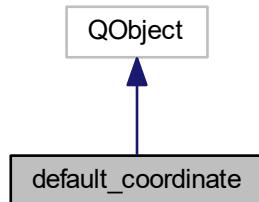
Additional Inherited Members

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

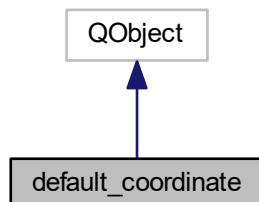
- E:/Gitin repo/ohjelmointi 3/whiskas/Game/buildings/custombuildingbase.h
- E:/Gitin repo/ohjelmointi 3/whiskas/Game/buildings/custombuildingbase.cpp

6.9 default_coordinate Class Reference

Inheritance diagram for default_coordinate:



Collaboration diagram for default_coordinate:

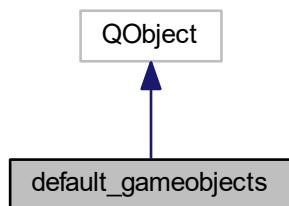


The documentation for this class was generated from the following file:

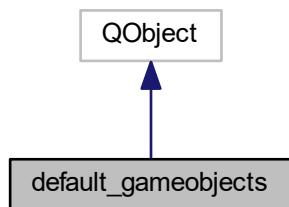
- E:/Gitin repo/ohjelmointi 3/whiskas/Course/UnitTests/coordinate_tests/tst_default_coordinate.cpp

6.10 default_gameobjects Class Reference

Inheritance diagram for default_gameobjects:



Collaboration diagram for default_gameobjects:

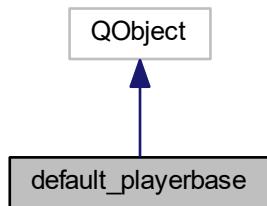


The documentation for this class was generated from the following file:

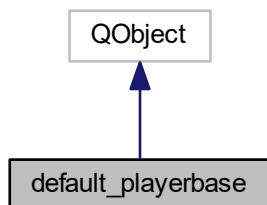
- E:/Gitin repo/ohjelmointi 3/whiskas/Course/UnitTests/gameobject_tests/tst_default_gameobjects.cpp

6.11 default_playerbase Class Reference

Inheritance diagram for default_playerbase:



Collaboration diagram for default_playerbase:

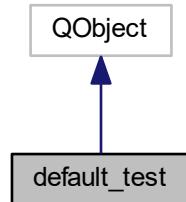


The documentation for this class was generated from the following file:

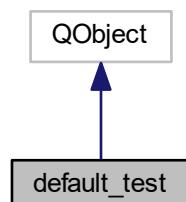
- E:/Gitin repo/ohjelmointi 3/whiskas/Course/UnitTests/playerbase_tests/tst_default_playerbase.cpp

6.12 default_test Class Reference

Inheritance diagram for default_test:



Collaboration diagram for default_test:



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

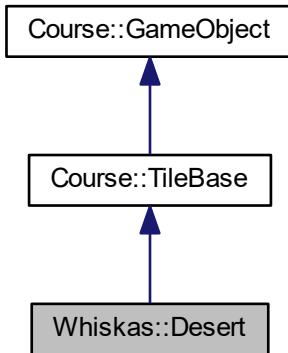
- E:/Gitin repo/ohjelmointi 3/whiskas/Course/UnitTests/placeablegameobject_tests/tst_default_test.cpp

6.13 Whiskas::Desert Class Reference

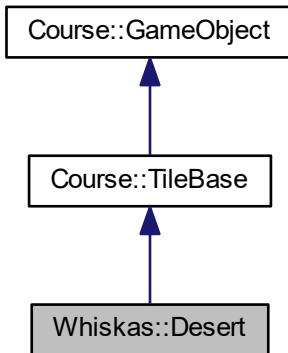
The [Desert](#) class represents the desert in the gameworld. The desert supports melee altars.

```
#include <desert.h>
```

Inheritance diagram for Whiskas::Desert:



Collaboration diagram for Whiskas::Desert:



Public Member Functions

- `Desert (const Course::Coordinate &location, const std::shared_ptr< Course::iGameEventHandler > &eventhandler, const std::shared_ptr< Course::iObjectManager > &objectmanager, const unsigned int &max_build=1, const unsigned int &max_work=1, const Course::ResourceMap &production={})`
- `virtual std::string getType () const override`
- `void addBuilding (const std::shared_ptr< Course::BuildingBase > &building) override`

Adds a new Building-object to the tile.

Additional Inherited Members

6.13.1 Detailed Description

The [Desert](#) class represents the desert in the gameworld. The desert supports melee altars.

6.13.2 Member Function Documentation

6.13.2.1 addBuilding()

```
void Whiskas::Desert::addBuilding (
    const std::shared_ptr< Course::BuildingBase > & building ) [override], [virtual]
```

Adds a new Building-object to the tile.

Phases:

1. Tile checks if it has space for the building.
2. Building checks whether it can be placed on this tile.
3. Building is added to this Tile.
4. Tile update's Building's location.

Parameters

<i>building</i>	A pointer to the Building that is being added.
-----------------	--

Postcondition

Exception guarantee: Basic

Exceptions

<i>InvalidPointer</i>	- If the building's pointer doesn't point to anything or ObjectManager doesn't return valid shared_ptr to this tile.
<i>IllegalMove</i>	- Any IllegalException can be thrown by a Tile or Building if it breaks a placement rule.
<i>NotEnoughSpace</i>	(IllegalMove) - If the tile doesn't have enough space for the Building.

Reimplemented from [Course::TileBase](#).

6.13.2.2 getType()

```
std::string Whiskas::Desert::getType ( ) const [override], [virtual]
```

Reimplemented from [Course::TileBase](#).

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

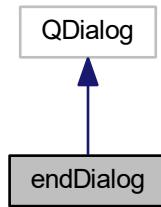
- E:/Gitin repo/ohjelmointi 3/whiskas/Game/tiles/desert.h
- E:/Gitin repo/ohjelmointi 3/whiskas/Game/tiles/desert.cpp

6.14 endDialog Class Reference

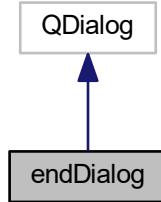
The [endDialog](#) class Shows the end message.

```
#include <enddialog.h>
```

Inheritance diagram for endDialog:



Collaboration diagram for endDialog:



Public Member Functions

- **endDialog** (QWidget *parent=nullptr)
- void **setEndText** (const std::string &text)

6.14.1 Detailed Description

The [endDialog](#) class Shows the end message.

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

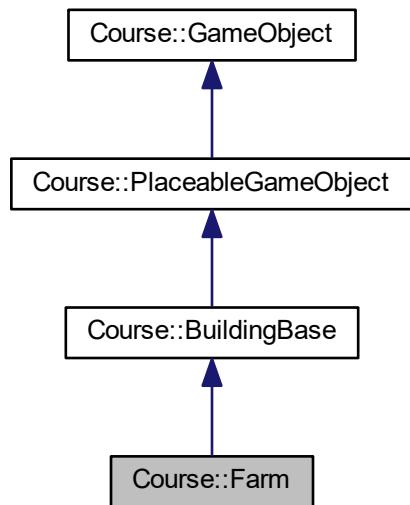
- E:/Gitin repo/ohjelmointi 3/whiskas/Game/dialogs/enddialog.h
- E:/Gitin repo/ohjelmointi 3/whiskas/Game/dialogs/enddialog.cpp

6.15 Course::Farm Class Reference

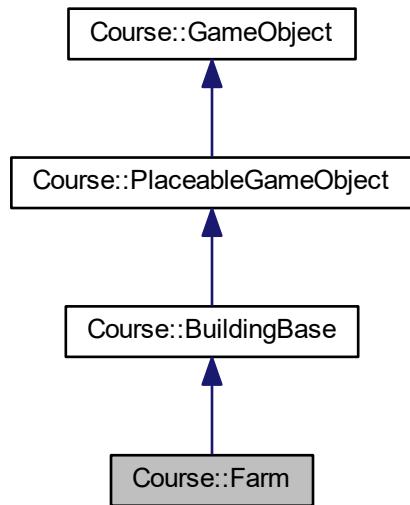
The [Farm](#) class represents a farm-building in the game.

```
#include <farm.h>
```

Inheritance diagram for Course::Farm:



Collaboration diagram for Course::Farm:



Public Member Functions

- `Farm ()=delete`
Disabled parameterless constructor.
- `Farm (const std::shared_ptr< iGameEventHandler > &eventhandler, const std::shared_ptr< iObjectManager > &objectmanager, const std::shared_ptr< PlayerBase > &owner, const int &tilespaces=1, const ResourceMap &buildcost=ConstResourceMaps::FARM_BUILD_COST, const ResourceMap &production=ConstResourceMaps::FARM_PRODUCTION)
Constructor for the class.`
- `~Farm () override=default`
Default destructor.
- `std::string getType () const override`
getType Returns a string describing objects type. This should be overriden in each inherited class. Makes checking object's type easier for students.

Additional Inherited Members

6.15.1 Detailed Description

The `Farm` class represents a farm-building in the game.

The farm adds 2 base-production for food.

6.15.2 Constructor & Destructor Documentation

6.15.2.1 Farm()

```
Course::Farm::Farm (
    const std::shared_ptr< iGameEventHandler > & eventhandler,
    const std::shared_ptr< iObjectManager > & objectmanager,
    const std::shared_ptr< PlayerBase > & owner,
    const int & tilespaces = 1,
    const ResourceMap & buildcost = ConstResourceMaps::FARM_BUILD_COST,
    const ResourceMap & production = ConstResourceMaps::FARM_PRODUCTION ) [explicit]
```

Constructor for the class.

Parameters

<code>eventhandler</code>	points to the student's GameEventHandler.
<code>owner</code>	points to the owning player.
<code>tile</code>	points to the tile upon which the building is constructed.

Postcondition

Exception Guarantee: No guarantee.

Exceptions

<i>OwnerConflict</i>	- if the building conflicts with tile's ownership.
----------------------	--

6.15.3 Member Function Documentation

6.15.3.1 getType()

```
std::string Course::Farm::getType () const [override], [virtual]
```

getType Returns a string describing objects type. This should be overriden in each inherited class. Makes checking object's type easier for students.

Returns

std::string that represents Object's type.

Postcondition

Exception guarantee: No-throw

Note

You can use this in e.g. debugging and similar printing.

Reimplemented from [Course::BuildingBase](#).

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

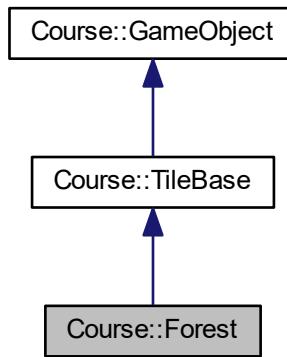
- E:/Gitin repo/ohjelmointi 3/whiskas/Course/CourseLib/buildings/farm.h
- E:/Gitin repo/ohjelmointi 3/whiskas/Course/CourseLib/buildings/farm.cpp

6.16 Course::Forest Class Reference

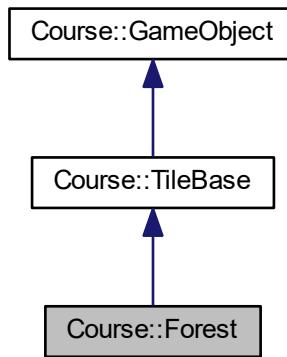
The [Forest](#) class represents [Forest](#) in the gameworld.

```
#include <forest.h>
```

Inheritance diagram for Course::Forest:



Collaboration diagram for Course::Forest:



Public Member Functions

- [Forest \(\)=delete](#)

Disabled parameterless constructor.

- `Forest (const Coordinate &location, const std::shared_ptr< iGameEventHandler > &eventhandler, const std::shared_ptr< iObjectManager > &objectmanager, const unsigned int &max_build=2, const unsigned int &max_work=3, const ResourceMap &production=ConstResourceMaps::FOREST_BP)`
Constructor for the class.
- `virtual ~Forest ()=default`
Default destructor.
- `virtual std::string getType () const override`
getType Returns a string describing objects type. This should be overriden in each inherited class. Makes checking object's type easier for students.
- `void addBuilding (const std::shared_ptr< BuildingBase > &building) override`
Adds a new building-object to the tile. Building in forest adds one hold-marker to the building.

Additional Inherited Members

6.16.1 Detailed Description

The `Forest` class represents `Forest` in the gameworld.

`Forest` has BasicProduction:

- Money = 1
- Food = 3
- Wood = 5
- Stone = 1
- Ore = 0

Building in the forest takes time. So buildings get extra hold-marker.

Tile supports 2 buildings.

6.16.2 Constructor & Destructor Documentation

6.16.2.1 Forest()

```
Course::Forest::Forest (
    const Coordinate & location,
    const std::shared_ptr< iGameEventHandler > & eventhandler,
    const std::shared_ptr< iObjectManager > & objectmanager,
    const unsigned int & max_build = 2,
    const unsigned int & max_work = 3,
    const ResourceMap & production = ConstResourceMaps::FOREST_BP )
```

Constructor for the class.

Parameters

<i>location</i>	is the Coordinate where the Tile is located in the game.
<i>eventhandler</i>	points to the student's GameEventHandler.

6.16.3 Member Function Documentation

6.16.3.1 addBuilding()

```
void Course::Forest::addBuilding (
    const std::shared_ptr< BuildingBase > & building ) [override], [virtual]
```

Adds a new building-object to the tile. Building in forest adds one hold-marker to the building.

Phases:

1. Check that there is space for the building.
2. Call parent's addBuilding
3. Add a HoldMarker for the building.

Postcondition

Exception guarantee: Strong

Exceptions

OwnerConflict	- If the tile's ownership prevents placing the building .
NoSpace	- If the tile doesn't have enough space for the building .

Reimplemented from [Course::TileBase](#).

Here is the call graph for this function:



6.16.3.2 `getType()`

```
std::string Course::Forest::getType () const [override], [virtual]
```

`getType` Returns a string describing objects type. This should be overriden in each inherited class. Makes checking object's type easier for students.

Returns

`std::string` that represents Object's type.

Postcondition

Exception guarantee: No-throw

Note

You can use this in e.g. debugging and similar printing.

Reimplemented from [Course::GameObject](#).

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

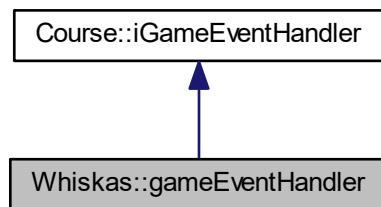
- E:/Gitin repo/ohjelmointi 3/whiskas/Course/CourseLib/tiles/forest.h
- E:/Gitin repo/ohjelmointi 3/whiskas/Course/CourseLib/tiles/forest.cpp

6.17 Whiskas::gameEventHandler Class Reference

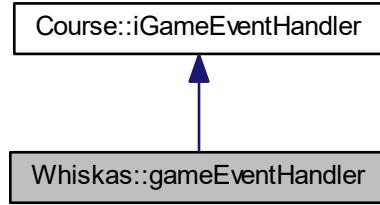
The [gameEventHandler](#) class Handles events the game can have. Most notably, the mouse presses. NOTE: The first few methods that have unused variables are there for the reason that course side documentation said that those are required by the course side code. We dont use them but we didnt know if we could remove them.

```
#include <gameeventhandler.h>
```

Inheritance diagram for Whiskas::gameEventHandler:



Collaboration diagram for Whiskas::gameEventHandler:



Public Member Functions

- `gameEventHandler (std::shared_ptr< Turn > turn, std::shared_ptr< QTextBrowser > textBrowserRight)`
Constructor.
- `bool modifyResources (std::shared_ptr< Course::PlayerBase > player, Course::ResourceMap resources)`
`override`

*modifyResources Does nothing, since the game doesent use Course::resourceMap //NOTE: It was unclear if these methods could be removed, since Igameeventhandler defines these, and it *IS* required by the course side.*
- `bool modifyResource (std::shared_ptr< Course::PlayerBase > player, Course::BasicResource resource, int amount)`
`override`
- `bool subtractPlayerResources (const std::shared_ptr< LeaguePlayer > &player, AdvancedResourceMap costs)`

subtractPlayerResources Reduces items from the player inventory.
- `void handleMwindowClick (const std::shared_ptr< GameScene > &scene, const std::shared_ptr< gameManager > &manager, const QMouseEvent &event)`

handleMwindowClick checks if the click was right, or left
- `void handleLeftClick (const std::shared_ptr< GameScene > &scene, const std::shared_ptr< gameManager > &manager)`

handleLeftClick Handles the left mouse button event, sets active tile and active minion
- `void handleRightClick (const std::shared_ptr< gameManager > &manager, const std::shared_ptr< GameScene > &scene)`

handleRightClick Handles the right mouse button event. Moves minions
- `std::shared_ptr< Course::TileBase > getActiveTile ()`

getActiveTile Gets the last tile that was clicked
- `void setActiveTile (std::shared_ptr< Course::TileBase > activeTile)`

setActiveTile Sets the active tile to be a desired tile
- `std::shared_ptr< Minion > getActiveMinion ()`

getActiveMinion returns the last active minion
- `std::shared_ptr< Turn > getTurn ()`

getTurn
- `void endTurn (const std::shared_ptr< gameManager > &manager, const std::shared_ptr< gameEventHandler > &handler)`

endTurn Calls spawn minions on a desired turn

6.17.1 Detailed Description

The `gameEventHandler` class Handles events the game can have. Most notably, the mouse presses. NOTE: The first few methods that have unused variables are there for the reason that course side documentation said that those are required by the course side code. We dont use them but we didnt know if we could remove them.

6.17.2 Constructor & Destructor Documentation

6.17.2.1 `gameEventHandler()`

```
Whiskas::gameEventHandler::gameEventHandler (
    std::shared_ptr< Turn > turn,
    std::shared_ptr< QTextBrowser > textBrowserRight )
```

Constructor.

Parameters

<code>turn</code>	For easier access to the turn class
<code>textBrowserRight</code>	for updating the right textBrowser

6.17.3 Member Function Documentation

6.17.3.1 `endTurn()`

```
void Whiskas::gameEventHandler::endTurn (
    const std::shared_ptr< gameManager > & manager,
    const std::shared_ptr< gameEventHandler > & handler )
```

`endTurn` Calls spawn minions on a desired turn

Parameters

<code>manager</code>	GameManager
<code>handler</code>	HandlerInstance

6.17.3.2 `getActiveMinion()`

```
std::shared_ptr< Minion > Whiskas::gameEventHandler::getActiveMinion ( )
```

`getActiveMinion` returns the last active minion

Returns

ptr to active minion

6.17.3.3 getActiveTile()

```
std::shared_ptr< Course::TileBase > Whiskas::gameEventHandler::getActiveTile ( )
```

getActiveTile Gets the last tile that was clicked

Returns

The tile Clicked on

6.17.3.4 getTurn()

```
std::shared_ptr< Turn > Whiskas::gameEventHandler::getTurn ( )
```

getTurn

Returns

The turn instance

6.17.3.5 handleLeftClick()

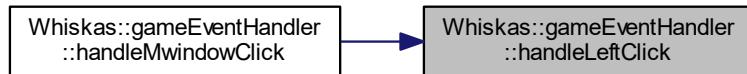
```
void Whiskas::gameEventHandler::handleLeftClick (
    const std::shared_ptr< GameScene > & scene,
    const std::shared_ptr< gameManager > & manager )
```

handleLeftClick Handles the left mouse button event, sets active tile and active minion

Parameters

<i>scene</i>	The game scene
<i>manager</i>	The gameManager

Here is the caller graph for this function:



6.17.3.6 handleMwindowClick()

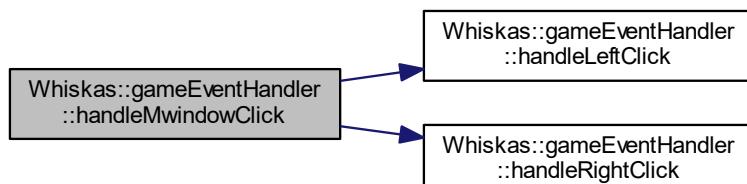
```
void Whiskas::gameEventHandler::handleMwindowClick (
    const std::shared_ptr< GameScene > & scene,
    const std::shared_ptr< gameManager > & manager,
    const QMouseEvent & event )
```

handleMwindowClick checks if the click was right, or left

Parameters

<i>scene</i>	passed on to other methods
<i>manager</i>	passed on to other methods
<i>event</i>	Right or left Click

Here is the call graph for this function:



6.17.3.7 handleRightClick()

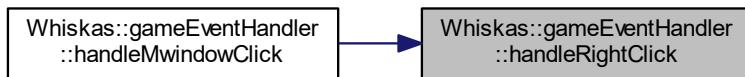
```
void Whiskas::gameEventHandler::handleRightClick (
    const std::shared_ptr< gameManager > & manager,
    const std::shared_ptr< GameScene > & scene )
```

handleRightClick Handles the right mouse button event. Moves minions

Parameters

<i>manager</i>	The gameManager
<i>scene</i>	The gameScene

Here is the caller graph for this function:

**6.17.3.8 modifyResource()**

```
bool Whiskas::gameEventHandler::modifyResource (
    std::shared_ptr< Course::PlayerBase > player,
    Course::BasicResource resource,
    int amount ) [override], [virtual]
```

See above!

Implements [Course::iGameEventHandler](#).

6.17.3.9 modifyResources()

```
bool Whiskas::gameEventHandler::modifyResources (
    std::shared_ptr< Course::PlayerBase > player,
    Course::ResourceMap resources ) [override], [virtual]
```

`modifyResources` Does nothing, since the game doesent use `Course::resourceMap` //NOTE: It was unclear if these methods could be removed, since `Igameeventhandler` defines these, and it *IS" required by the course side.

Parameters

<i>player</i>	does nothing
<i>resources</i>	does even less of anything

Returns

true, but what is the point

Implements [Course::iGameEventHandler](#).

6.17.3.10 setActiveTile()

```
void Whiskas::gameEventHandler::setActiveTile (
    std::shared_ptr< Course::TileBase > activeTile )
```

`setActiveTile` Sets the active tile to be a desired tile

Parameters

<i>activeTile</i>	
-------------------	--

6.17.3.11 subtractPlayerResources()

```
bool Whiskas::gameEventHandler::subtractPlayerResources (
    const std::shared_ptr< LeaguePlayer > & player,
    AdvancedResourceMap costs )
```

`subtractPlayerResources` Reduces items from the player inventory.

Parameters

<i>player</i>	Target player
<i>costs</i>	The amount to be reduced

Returns

true if player had enough items, false otherwise.

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

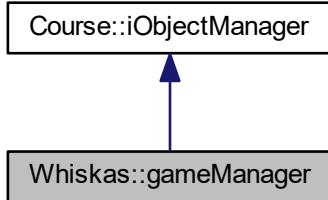
- E:/Gitin repo/ohjelmointi 3/whiskas/Game/handlerandmanager/gameeventhandler.h
- E:/Gitin repo/ohjelmointi 3/whiskas/Game/handlerandmanager/gameeventhandler.cpp

6.18 Whiskas::gameManager Class Reference

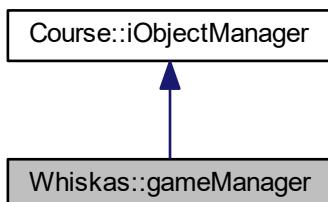
The [gameManager](#) class is a class that manages the game logic. It has access to almost all the classes included in this game.

```
#include <gamemanager.h>
```

Inheritance diagram for Whiskas::gameManager:



Collaboration diagram for Whiskas::gameManager:



Public Member Functions

- `gameManager (std::shared_ptr< GameScene > &m_gamescene, std::shared_ptr< QTextBrowser > browser)`
`gameManager Constructor`
- `void addTiles (const std::vector< std::shared_ptr< Course::TileBase > > &tiles) override`
`Copies the tilesvector to managers own private vector.`
- `void addTile (const std::shared_ptr< Course::TileBase > &tile)`
`Adds a single tile to the private tile vector.`
- `std::shared_ptr< Course::TileBase > getTile (const Course::ObjectId &id) override`
`Return a tile matching the object ID.`
- `std::shared_ptr< Course::TileBase > getTile (const Course::Coordinate &coordinate) override`
`Return a tile matching the Coordinate.`
- `std::vector< std::shared_ptr< Course::TileBase > > getTiles (const std::vector< Course::Coordinate > &coordinates) override`
`getTiles returns all tiles. This is a placeholder since this method is never used anywhere`
- `std::vector< std::shared_ptr< Course::TileBase > > getTileVector ()`
`getTileVector returns all tiles the gamemanager has.`
- `std::vector< std::shared_ptr< CustomBuildingBase > > getBuildingVector ()`

- `std::vector< std::shared_ptr< Minion > > getMinionVector ()`

getBuildingVector returns all the bulding the gamemanager has.
- `std::pair< std::shared_ptr< Whiskas::LeaguePlayer >, std::shared_ptr< Whiskas::LeaguePlayer > > getPlayerPair ()`

Returns a vector of all the minions in excistance. In every multiverse.
- `void addBuilding (std::shared_ptr< CustomBuildingBase > &building)`

addBuilding adds a building pointer to the building vector and all attackables vector. (Since buildings are attackable).
- `void addPlayer (std::pair< std::shared_ptr< Whiskas::LeaguePlayer >, std::shared_ptr< Whiskas::LeaguePlayer > >> &player)`

Copies the player pair for managers own use.
- `void addMinion (const std::shared_ptr< Minion > &Minion)`

addMinion adds a minion ptr to gameManagers own minion vector and attackables vector
- `bool spawnMinion (const std::shared_ptr< gameEventHandler > &handler, const std::shared_ptr< Course::iObjectManager > &manager, const std::shared_ptr< Course::PlayerBase > &owner, const std::shared_ptr< Course::TileBase > &location, const std::string &type)`

spawnMinion Spawns a new minion to desired tile.
- template<typename buildingType>
`bool spawnBuilding (std::shared_ptr< gameEventHandler > handler, std::shared_ptr< gameManager > manager, std::shared_ptr< LeaguePlayer > player)`

spawnBuilding Spawns a new building of specific type. IMPLEMENTATION IN FUNCTIONS.CPP
- `void move (const std::shared_ptr< Minion > &minionToMove, const std::shared_ptr< Course::TileBase > &targetTile)`

move Tries to move a minion to the target tile. Checks for movement range, collision and possible enemies. If enemies are found, tries to attack them.
- `void attack (const std::shared_ptr< Minion > &minionToAttack, const std::shared_ptr< Attackable > &target)`

attack Select minion tries to deal its damage to target attackable. If target hp goes to 0, tries to destroy it
- `bool checkForEnemies (const std::shared_ptr< Minion > &minionTomove, const std::shared_ptr< Course::TileBase > &targetTile)`

checkForEnemies Check if target tile contains minions or buildings of the other player
- `std::shared_ptr< Attackable > selectAttackTarget (const std::shared_ptr< Course::TileBase > &targetTile)`

selectAttackTarget Returns a attackable ptr for someone to attack. Minions have priority.
- `void attackMultiple (const std::shared_ptr< Minion > &minionToAttack, const std::shared_ptr< Course::TileBase > &targetTile)`

attackMultiple Used by the mage minion to attack in an area. Attacks all minions in a radius of 1.
- `void destroyObject (const std::shared_ptr< Attackable > &ObjectToDelete)`

destroyObject Takes an attackable object, and removes it from all databases. (Gamemanager vectors, tileWorkers, Tilebuildings, scene_items...) Also checks if the destroyed object is a Nexus. If it is, win the game.
- `std::shared_ptr< Course::TileBase > getNexusLocation (const std::shared_ptr< LeaguePlayer > &owner)`

getNexusLocation Returns a tile containing the nexus of the desired player
- `std::shared_ptr< LeaguePlayer > getWinner ()`

getWinner Get the winning player
- `bool checkBuildingAvailability (const std::shared_ptr< Course::TileBase > &targetTile, const std::string &type)`

checkBuildingAvailability checks if building can be built on tile

6.18.1 Detailed Description

The `gameManager` class is a class that manages the game logic. It has access to almost all the classes included in this game.

6.18.2 Constructor & Destructor Documentation

6.18.2.1 gameManager()

```
Whiskas::gameManager::gameManager (
    std::shared_ptr< GameScene > & m_gamescene,
    std::shared_ptr< QTextBrowser > browser ) [explicit]
```

gameManager Constructor

Parameters

<i>m_gamescene</i>	for easier access to graphics
<i>browser</i>	For easier management of the textbrowsers

6.18.3 Member Function Documentation

6.18.3.1 addBuilding()

```
void Whiskas::gameManager::addBuilding (
    std::shared_ptr< CustomBuildingBase > & building )
```

addBuilding adds a building pointer to the building vector and all attackables vector. (Since buildings are attackable).

Parameters

<i>building</i>	to be added
-----------------	-------------

6.18.3.2 addMinion()

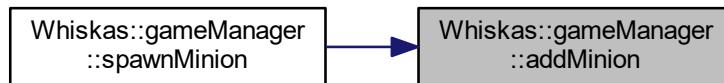
```
void Whiskas::gameManager::addMinion (
    const std::shared_ptr< Minion > & Minion )
```

addMinion adds a minion ptr to gameManagers own minion vector and attackables vector

Parameters

<i>Minion</i>	to be added
---------------	-------------

Here is the caller graph for this function:



6.18.3.3 addPlayer()

```
void Whiskas::gameManager::addPlayer (
    std::pair< std::shared_ptr< Whiskas::LeaguePlayer >, std::shared_ptr< Whiskas::LeaguePlayer >> & player )
```

Copies the player pair for managers own use.

Parameters

<i>player</i>	to be copied
---------------	--------------

6.18.3.4 addTile()

```
void Whiskas::gameManager::addTile (
    const std::shared_ptr< Course::TileBase > & tile )
```

Adds a single tile to the private tile vector.

Parameters

<i>tile</i>	to be added
-------------	-------------

6.18.3.5 addTiles()

```
void Whiskas::gameManager::addTiles (
    const std::vector< std::shared_ptr< Course::TileBase > > & tiles ) [override]
```

Copies the tilesvector to managers own private vector.

Parameters

<code>vector</code>	to be copied
---------------------	--------------

Postcondition

Exception guarantee: strong

6.18.3.6 attack()

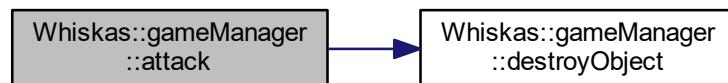
```
void Whiskas::gameManager::attack (
    const std::shared_ptr< Minion > & minionToAttack,
    const std::shared_ptr< Attackable > & target )
```

attack Select minion tries to deal its damage to target attackable. If target hp goes to 0, tries to destroy it

Parameters

<code>minionToAttack</code>	Minion doing the attacking
<code>toAttack</code>	The target

Here is the call graph for this function:



Here is the caller graph for this function:



6.18.3.7 attackMultiple()

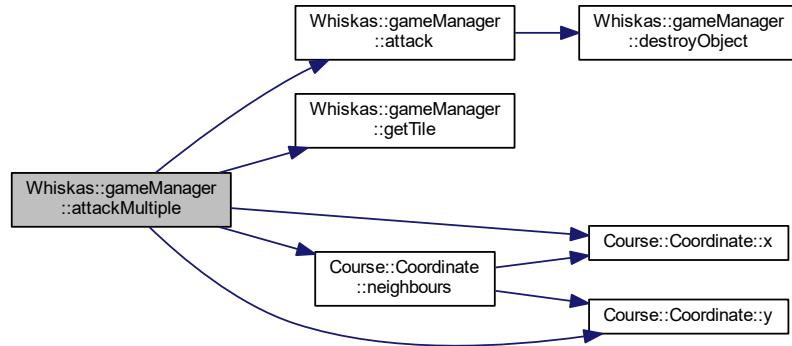
```
void Whiskas::gameManager::attackMultiple (
    const std::shared_ptr< Minion > & minionToAttack,
    const std::shared_ptr< Course::TileBase > & targetTile )
```

attackMultiple Used by the mage minion to attack in an area. Attacks all minions in a radius of 1.

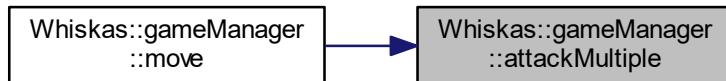
Parameters

<i>minionToAttack</i>	Minion doing the attacking
<i>targetTile</i>	

Here is the call graph for this function:



Here is the caller graph for this function:



6.18.3.8 checkBuildingAvailability()

```
bool Whiskas::gameManager::checkBuildingAvailability (
    const std::shared_ptr< Course::TileBase > & targetTile,
    const std::string & type )
```

checkBuildingAvailability checks if building can be built on tile

Parameters

<i>targetTile</i>	
<i>type</i>	the type of building to be checked

Returns

true if can be built, false if not

6.18.3.9 checkForEnemies()

```
bool Whiskas::gameManager::checkForEnemies (
    const std::shared_ptr< Minion > & minionTomove,
    const std::shared_ptr< Course::TitleBase > & targetTile )
```

checkForEnemies Check if target tile contains minions or buildings of the other player

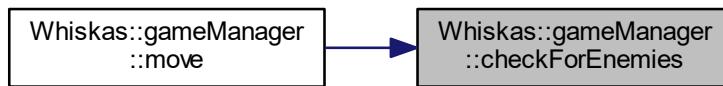
Parameters

<i>minionTomove</i>	Minion doing the moving
<i>targetTile</i>	Destination

Returns

true if enemies, false if no

Here is the caller graph for this function:

**6.18.3.10 destroyObject()**

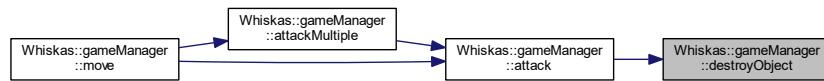
```
void Whiskas::gameManager::destroyObject (
    const std::shared_ptr< Attackable > & ObjectToDestroy )
```

destroyObject Takes an attackable object, and removes it from all databases. (Gamemanager vectors, tileWorkers, Tilebuildings, scene_items...) Also checks if the destroyed object is a Nexus. If it is, win the game.

Parameters

<i>ObjectToDestroy</i>	the object to be destroyed
------------------------	----------------------------

Here is the caller graph for this function:

**6.18.3.11 getBuildingVector()**

```
std::vector< std::shared_ptr< CustomBuildingBase > > Whiskas::gameManager::getBuildingVector( )
```

getBuildingVector returns all the building the gamemanager has.

Returns

Vector containing all the buildings

6.18.3.12 getMinionVector()

```
std::vector< std::shared_ptr< Minion > > Whiskas::gameManager::getMinionVector( )
```

Returns a vector of all the minions in existence. In every multiverse.

Returns

Vector containing all the minions.

6.18.3.13 getNexusLocation()

```
std::shared_ptr< Course::TileBase > Whiskas::gameManager::getNexusLocation(
    const std::shared_ptr< LeaguePlayer > & owner )
```

getNexusLocation Returns a tile containing the nexus of the desired player

Parameters

<code>owner</code>	Desired player
--------------------	----------------

Returns

Tile containing the Nexus

6.18.3.14 getPair()

```
std::pair< std::shared_ptr< Whiskas::LeaguePlayer >, std::shared_ptr< Whiskas::LeaguePlayer > > Whiskas::gameManager::getPair ()
```

getPair Returns both players in the game

Returns

a pair containing both players

6.18.3.15 getTile() [1/2]

```
std::shared_ptr< Course::TileBase > Whiskas::gameManager::getTile (
    const Course::Coordinate & coordinate ) [override], [virtual]
```

Return a tile matching the Coordinate.

Parameters

<code>coordinate</code>	<input type="checkbox"/>
-------------------------	--------------------------

Returns

matching tile

Implements [Course::iObjectManager](#).

6.18.3.16 getTile() [2/2]

```
std::shared_ptr< Course::TileBase > Whiskas::gameManager::getTile (
    const Course::ObjectId & id ) [override], [virtual]
```

Return a tile matching the object ID.

Parameters

<i>ID</i>	
-----------	--

Returns

matching tile

Implements [Course::iObjectManager](#).

Here is the caller graph for this function:

**6.18.3.17 getTiles()**

```
std::vector< std::shared_ptr< Course::TileBase > > Whiskas::gameManager::getTiles (
    const std::vector< Course::Coordinate > & coordinates ) [override], [virtual]
```

getTiles returns all tiles. This is a placeholder since this method is never used anywhere

Parameters

<i>coordinates</i>	unused
--------------------	--------

Returns

all tiles in a vector

Implements [Course::iObjectManager](#).

6.18.3.18 getTileVector()

```
std::vector< std::shared_ptr< Course::TileBase > > Whiskas::gameManager::getTileVector ( )
```

getTileVector returns all tiles the gamemanager has.

Returns

Vector containing all the tiles.

6.18.3.19 getWinner()

```
std::shared_ptr< LeaguePlayer > Whiskas::gameManager::getWinner ( )
```

getWinner Get the winning player

Returns

ptr to the player who won the game

6.18.3.20 move()

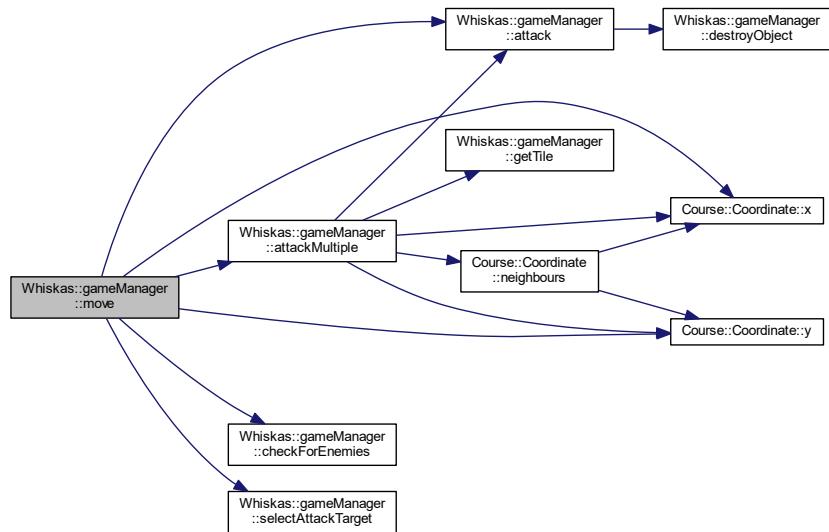
```
void Whiskas::gameManager::move (
    const std::shared_ptr< Minion > & minionToMove,
    const std::shared_ptr< Course::TileBase > & targetTile )
```

move Tries to move a minion to the target tile. Checks for movement range, collision and possible enemies. If enemies are found, tries to attack them.

Parameters

<i>minionToMove</i>	The minion doing the moving
<i>targetTile</i>	Destination

Here is the call graph for this function:



6.18.3.21 selectAttackTarget()

```
std::shared_ptr< Attackable > Whiskas::gameManager::selectAttackTarget (
    const std::shared_ptr< Course::TileBase > & targetTile )
```

selectAttackTarget Returns a attackable ptr for someone to attack. Minions have priority.

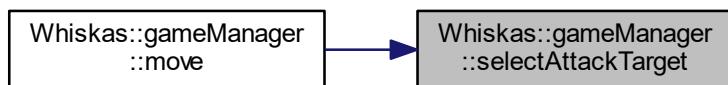
Parameters

<i>targetTile</i>	
-------------------	--

Returns

Attcable ptr if succsefull, nullptr if something goes wrong

Here is the caller graph for this function:



6.18.3.22 spawnBuilding()

```
template<typename buildingType >
bool gameManager::spawnBuilding (
    std::shared_ptr< gameEventHandler > handler,
    std::shared_ptr< gameManager > manager,
    std::shared_ptr< LeaguePlayer > player )
```

spawnBuilding Spawns a new building of specific type. IMPLEMENTATION IN FUNCTIONS.CPP

Parameters

<i>handler</i>	buildings handler
<i>manager</i>	recieves the tile
<i>player</i>	who owns the tile

Returns

false if failed, true otherwise

6.18.3.23 spawnMinion()

```
bool Whiskas::gameManager::spawnMinion (
    const std::shared_ptr< gameEventHandler > & handler,
    const std::shared_ptr< Course::iObjectManager > & manager,
    const std::shared_ptr< Course::PlayerBase > & owner,
    const std::shared_ptr< Course::TileBase > & location,
    const std::string & type )
```

spawnMinion Spawns a new minion to desired tile.

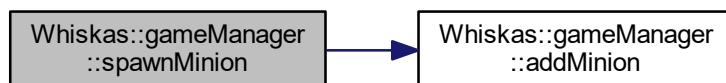
Parameters

<i>handler</i>	minions own handler
<i>manager</i>	Recieves the minion
<i>owner</i>	Who controls the minion
<i>location</i>	where the minion spawns
<i>type</i>	of minion to be spawned(Eg:minion,mage..)

Returns

true if successful, false otherwise

Here is the call graph for this function:



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

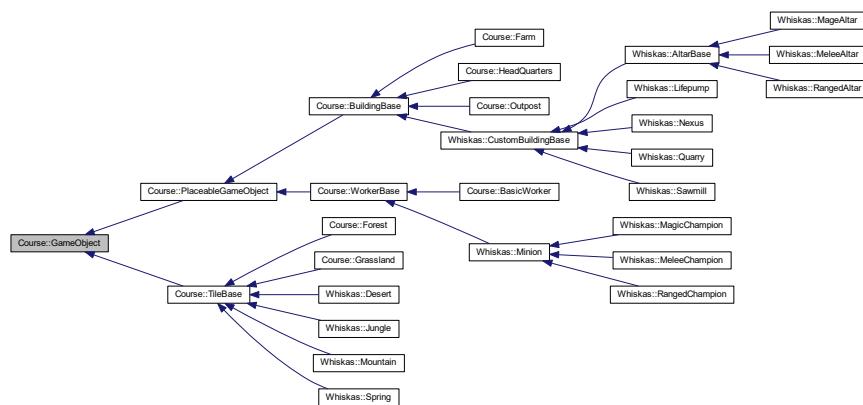
- E:/Gitin repo/ohjelmointi 3/whiskas/Game/handlerandmanager/gamemanager.h
- E:/Gitin repo/ohjelmointi 3/whiskas/Game/functions/functions.cpp
- E:/Gitin repo/ohjelmointi 3/whiskas/Game/handlerandmanager/gamemanager.cpp
- E:/Gitin repo/ohjelmointi 3/whiskas/UnitTests/handlerTest/tst_testhandler.cpp
- E:/Gitin repo/ohjelmointi 3/whiskas/UnitTests/managerTest/tst_testmanager.cpp

6.19 Course::GameObject Class Reference

The [GameObject](#) class is base-class that contains general information on different Objects in game.

```
#include <gameobject.h>
```

Inheritance diagram for Course::GameObject:



Public Member Functions

- `GameObject (const GameObject &original)`
A simple copy-constructor for GameObject.
- `GameObject (const std::shared_ptr< iGameEventHandler > &eventhandler={nullptr}, const std::shared_ptr< iObjectManager > &objectmanager={nullptr})`
Constructor that only specifies GameEventHandler and ObjectManager.
- `GameObject (const std::shared_ptr< PlayerBase > &owner, const std::shared_ptr< iGameEventHandler > &eventhandler={nullptr}, const std::shared_ptr< iObjectManager > &objectmanager={nullptr})`
GameObject constructor that can specify an owner.
- `GameObject (const Coordinate &coordinate, const std::shared_ptr< PlayerBase > &owner, const std::shared_ptr< iGameEventHandler > &eventhandler={nullptr}, const std::shared_ptr< iObjectManager > &objectmanager={nullptr})`
GameObject constructor that can specify a coordinate and an owner.
- `GameObject (const Coordinate &coordinate, const std::shared_ptr< iGameEventHandler > &eventhandler={nullptr}, const std::shared_ptr< iObjectManager > &objectmanager={nullptr})`
GameObject constructor that can specify a coordinate.
- `virtual ~GameObject ()=default`
~GameObject Default destructor.
- `virtual void setOwner (const std::shared_ptr< PlayerBase > &owner) final`
Change GameObject's "owner".
- `virtual void setCoordinate (const std::shared_ptr< Coordinate > &coordinate) final`
Change GameObject's coordinate with a shared pointer to a coordinate.
- `virtual void setCoordinate (const Coordinate &coordinate) final`
Change GameObject's coordinate.
- `virtual void unsetCoordinate () final`
Removes the coordinate from GameObject.
- `virtual void setDescriptions (const DescriptionMap &descriptions) final`
Change GameObject's "descriptions"-map.
- `virtual void addDescription (const std::string &key, const std::string &content) final`
Adds a new description to description map.
- `virtual void setDescription (const std::string &key, const std::string &content) final`
Sets a description for the specified key.
- `virtual std::string getDescription (const std::string &key) const final`

- `virtual void removeDescription (const std::string &key) final`
Returns the content with specified key from the descriptions.
- `virtual void removeDescriptions () final`
Removes all content from descriptions.
- `virtual std::shared_ptr< PlayerBase > getOwner () const final`
Returns GameObject's owner.
- `virtual std::shared_ptr< Coordinate > getCoordinatePtr () const final`
Returns a pointer to a copy of GameObject's coordinate.
- `virtual Coordinate getCoordinate () const final`
Returns GameObject's current coordinate.
- `virtual std::map< std::string, std::string > getDescriptions () const final`
Returns the map of descriptions in GameObject.
- `virtual std::string getType () const`
GetType Returns a string describing objects type. This should be overriden in each inherited class. Makes checking object's type easier for students.
- `virtual bool hasSameOwnerAs (const std::shared_ptr< GameObject > &other) const final`
Function to compare if objects have same owner.
- `virtual bool hasSameCoordinateAs (const std::shared_ptr< GameObject > &other) const final`
has_same_coordinate

Public Attributes

- `const ObjectId ID`
ID is a constant value that can be used to identify GameObjects through ID numbers.

Protected Member Functions

- `virtual std::shared_ptr< iGameEventHandler > lockEventHandler () const final`
This is the primary method for locking GameEventHandler inside different GameObject-classes.
- `virtual std::shared_ptr< iObjectManager > lockObjectManager () const final`
This is the primary method for locking ObjectManager inside different GameObject classes.

6.19.1 Detailed Description

The `GameObject` class is base-class that contain's general information on different Objects in game.

- ID
- Possible owner
- Possible location coordinate
- Metadata in a string->string map
- Pointers to GameEventHandler and ObjectManager

Note

The functions consist mainly of get and set -functions that are used to access and store the information specified above.

6.19.2 Constructor & Destructor Documentation

6.19.2.1 GameObject() [1/5]

```
Course::GameObject::GameObject (
    const GameObject & original )
```

A simple copy-constructor for [GameObject](#).

Parameters

<i>original</i>	is the original GameObject
-----------------	--

6.19.2.2 GameObject() [2/5]

```
Course::GameObject::GameObject (
    const std::shared_ptr< iGameEventHandler > & eventhandler = {nullptr},
    const std::shared_ptr< iObjectManager > & objectmanager = {nullptr} )
```

Constructor that only specifies GameEventHandler and ObjectManager.

Parameters

<i>eventhandler</i>	a shared pointer to Game's GameEventHandler.
<i>objectmanager</i>	a shared pointer to Game's ObjectManager.

6.19.2.3 GameObject() [3/5]

```
Course::GameObject::GameObject (
    const std::shared_ptr< PlayerBase > & owner,
    const std::shared_ptr< iGameEventHandler > & eventhandler = {nullptr},
    const std::shared_ptr< iObjectManager > & objectmanager = {nullptr} )
```

[GameObject](#) constructor that can specify an owner.

Parameters

<i>owner</i>	a shared pointer to player that "owns" the object.
<i>eventhandler</i>	a shared pointer to Game's GameEventHandler.
<i>objectmanager</i>	a shared pointer to Game's ObjectManager.

6.19.2.4 GameObject() [4/5]

```
Course::GameObject::GameObject (
    const Coordinate & coordinate,
    const std::shared_ptr< PlayerBase > & owner,
    const std::shared_ptr< iGameEventHandler > & eventhandler = {nullptr},
    const std::shared_ptr< iObjectManager > & objectmanager = {nullptr} )
```

[GameObject](#) constructor that can specify a coordinate and an owner.

Parameters

<i>coordinate</i>	a shared pointer to coordinates where the object is located.
<i>owner</i>	a shared pointer to player that "owns" the object.
<i>eventhandler</i>	a shared pointer to Game's GameEventHandler.
<i>objectmanager</i>	a shared pointer to Game's ObjectManager.

6.19.2.5 GameObject() [5/5]

```
Course::GameObject::GameObject (
    const Coordinate & coordinate,
    const std::shared_ptr< iGameEventHandler > & eventhandler = {nullptr},
    const std::shared_ptr< iObjectManager > & objectmanager = {nullptr} )
```

[GameObject](#) constructor that can specify a coordinate.

Parameters

<i>coordinate</i>	a shared pointer to coordinates where the object is located.
<i>eventhandler</i>	a shared pointer to Game's GameEventHandler.
<i>objectmanager</i>	a shared pointer to Game's ObjectManager.

6.19.3 Member Function Documentation

6.19.3.1 addDescription()

```
void Course::GameObject::addDescription (
    const std::string & key,
    const std::string & content ) [final], [virtual]
```

Adds a new description to description map.

Parameters

<i>key</i>	Key for the content
<i>content</i>	Content being stored

Postcondition

Exception guarantee: Strong

Exceptions

<i>KeyError</i>	- If the key is already in use in the description-map.
See	std::map::operator[]

6.19.3.2 getCoordinate()

```
Coordinate Course::GameObject::getCoordinate () const [final], [virtual]
```

Returns [GameObject](#)'s current coordinate.

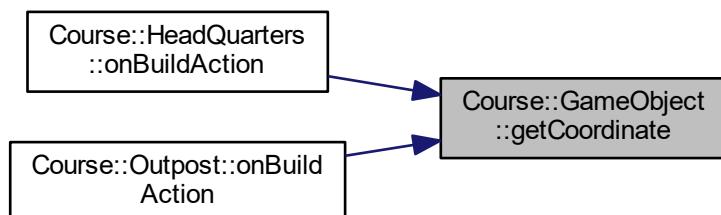
Postcondition

Exception guarantee: Strong

Exceptions

<i>InvalidPointer</i>	- If the GameObject doesn't have a coordinate.
-----------------------	--

Here is the caller graph for this function:



6.19.3.3 getCoordinatePtr()

```
std::shared_ptr< Coordinate > Course::GameObject::getCoordinatePtr () const [final], [virtual]
```

Returns a pointer to a copy of [GameObject](#)'s coordinate.

Returns

Shared-pointer copy of the [GameObject](#)'s coordinate, if the [GameObject](#) has a coordinate. Null-shared-pointer if the [GameObject](#) has no coordinate.

Postcondition

Exception guarantee: Strong

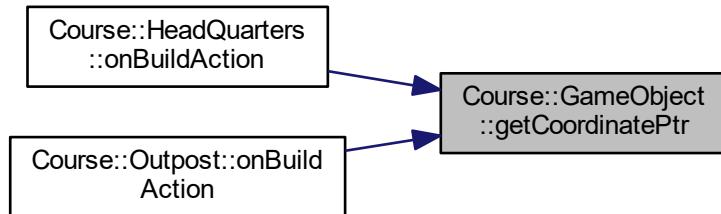
Exceptions

<i>See</i>	<code>std::make_shared</code>
------------	-------------------------------

Note

To change [GameObject](#)'s coordinate you must use `setCoordinate`. This prevent unwanted alterations by accident.

Here is the caller graph for this function:



6.19.3.4 getDescription()

```
std::string Course::GameObject::getDescription (
    const std::string & key ) const [final], [virtual]
```

Returns the content with specified key from the descriptions.

Parameters

<code>key</code>	Key for the content
------------------	---------------------

Postcondition

Exception guarantee: Strong @exceptions [KeyError](#) - if the description for the key is not found.

6.19.3.5 `getDescriptions()`

```
std::map< std::string, std::string > Course::GameObject::getDescriptions ( ) const [final], [virtual]
```

Returns the map of descriptions in [GameObject](#).

Returns

`std::map` of strings referring to strings.

Postcondition

Exception guarantee: No-throw

6.19.3.6 `getOwner()`

```
std::shared_ptr< PlayerBase > Course::GameObject::getOwner ( ) const [final], [virtual]
```

Returns [GameObject](#)'s owner.

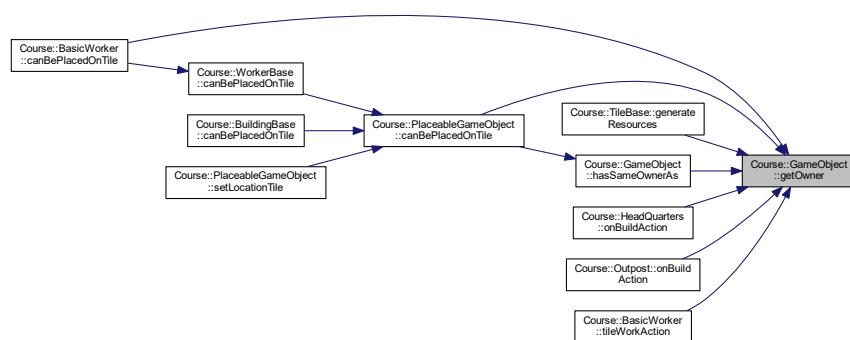
Returns

A shared-pointer to [GameObject](#)'s owner

Postcondition

Exception guarantee: No-throw

Here is the caller graph for this function:



6.19.3.7 getType()

```
std::string Course::GameObject::getType ( ) const [virtual]
```

getType Returns a string describing objects type. This should be overriden in each inherited class. Makes checking object's type easier for students.

Returns

std::string that represents Object's type.

Postcondition

Exception guarantee: No-throw

Note

You can use this in e.g. debugging and similar printing.

Reimplemented in [Course::WorkerBase](#), [Course::TileBase](#), [Course::BuildingBase](#), [Course::Outpost](#), [Course::BasicWorker](#), [Course::PlaceableGameObject](#), [Course::Grassland](#), [Course::HeadQuarters](#), [Course::Forest](#), [Course::Farm](#), [Whiskas::Minion](#), [Whiskas::Nexus](#), [Whiskas::Quarry](#), [Whiskas::Sawmill](#), [Whiskas::LifePump](#), [Whiskas::Jungle](#), [Whiskas::Mountain](#), [Whiskas::Spring](#), [Whiskas::Desert](#), [Whiskas::MeleeChampion](#), [Whiskas::MagicChampion](#), [Whiskas::RangedChampion](#), [Whiskas::AltarBase](#), [Whiskas::MeleeAltar](#), [Whiskas::MageAltar](#), and [Whiskas::RangedAltar](#).

6.19.3.8 hasSameCoordinateAs()

```
bool Course::GameObject::hasSameCoordinateAs ( const std::shared_ptr< GameObject > & other ) const [final], [virtual]
```

has_same_coordinate

Parameters

<i>other</i>	The other GameObject
--------------	--------------------------------------

Returns

True - If coordinates match or both are null
False - If the coordinates don't match

Postcondition

Exception guarantee: Strong

6.19.3.9 hasSameOwnerAs()

```
bool Course::GameObject::hasSameOwnerAs (
    const std::shared_ptr< GameObject > & other ) const [final], [virtual]
```

Function to compare if objects have same owner.

Parameters

<i>other</i>	The other GameObject
--------------	--------------------------------------

Returns

- True - If owners match or both are null False - If owners don't match

Postcondition

Exception guarantee: Strong

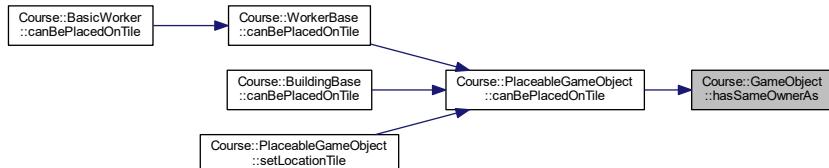
Exceptions

<i>ExpiredPointer</i>	- If any owner weak_ptr has expired.
-----------------------	--------------------------------------

Here is the call graph for this function:



Here is the caller graph for this function:



6.19.3.10 lockEventHandler()

```
std::shared_ptr< iGameEventHandler > Course::GameObject::lockEventHandler ( ) const [final],  
[protected], [virtual]
```

This is the primary method for locking GameEventHandler inside different GameObject-classes.

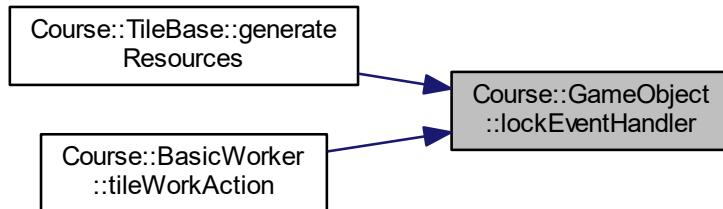
Returns

shared pointer to the GameEventHandler

Postcondition

Exception guarantee: No-throw

Here is the caller graph for this function:



6.19.3.11 lockObjectManager()

```
std::shared_ptr< iObjectManager > Course::GameObject::lockObjectManager ( ) const [final],  
[protected], [virtual]
```

This is the primary method for locking ObjectManager inside different [GameObject](#) classes.

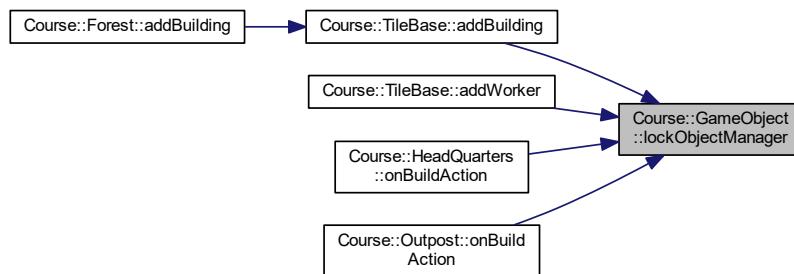
Returns

shared pointer to the ObjectManager

Postcondition

Exception guarantee: No-throw

Here is the caller graph for this function:

**6.19.3.12 removeDescription()**

```
void Course::GameObject::removeDescription (
    const std::string & key ) [final], [virtual]
```

Removes the content with specified key from the descriptions.

Parameters

<i>key</i>	Key for the content
------------	---------------------

Postcondition

Exception guarantee: Strong @exceptions [KeyError](#) - if the description for the key is not found.

6.19.3.13 removeDescriptions()

```
void Course::GameObject::removeDescriptions () [final], [virtual]
```

Removes all content from descriptions.

Postcondition

Exception guarantee: No-throw

6.19.3.14 setCoordinate() [1/2]

```
void Course::GameObject::setCoordinate (
    const Coordinate & coordinate ) [final], [virtual]
```

Change [GameObject](#)'s coordinate.

Parameters

<i>coordinate</i>	The new coordinate.
-------------------	---------------------

Postcondition

Exception guarantee: No-throw

6.19.3.15 setCoordinate() [2/2]

```
void Course::GameObject::setCoordinate (
    const std::shared_ptr< Coordinate > & coordinate ) [final], [virtual]
```

Change [GameObject](#)'s coordinate with a shared pointer to a coordinate.

Parameters

<i>coordinate</i>	A shared-pointer to the new coordinate.
-------------------	---

Postcondition

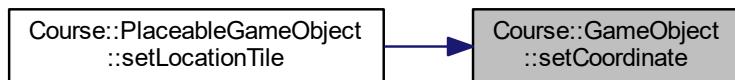
Exception guarantee: No-throw

Note

This creates new [Coordinate](#) based on the coordinate. The [Coordinate](#) can't be modified from outside of the class.

Can be used to unset coordinate with null-shared-pointer.

Here is the caller graph for this function:



6.19.3.16 setDescription()

```
void Course::GameObject::setDescription (
    const std::string & key,
    const std::string & content ) [final], [virtual]
```

Sets a description for the specified key.

Parameters

<i>key</i>	Key for the content
<i>content</i>	Content being stored

Postcondition

Exception guarantee: Strong @exceptions See std::map::operator[]

6.19.3.17 setDescriptions()

```
void Course::GameObject::setDescriptions (
    const DescriptionMap & descriptions ) [final], [virtual]
```

Change [GameObject](#)'s "descriptions"-map.

Parameters

<i>descriptions</i>	The new "descriptions"-map of strings referring to strings
---------------------	--

Postcondition

Exception guarantee: No-throw

6.19.3.18 setOwner()

```
void Course::GameObject::setOwner (
    const std::shared_ptr< PlayerBase > & owner ) [final], [virtual]
```

Change [GameObject](#)'s "owner".

Parameters

<i>owner</i>	a shared pointer to the new "owner".
--------------	--------------------------------------

Postcondition

Exception guarantee: No-throw

6.19.3.19 unsetCoordinate()

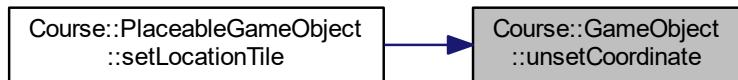
```
void Course::GameObject::unsetCoordinate ( ) [final], [virtual]
```

Removes the coordinate from [GameObject](#).

Postcondition

Exception guarantee: No-throw

Here is the caller graph for this function:

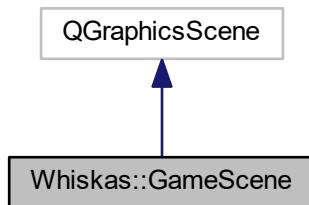


The documentation for this class was generated from the following files:

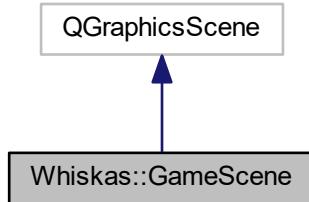
- E:/Gitin repo/ohjelmointi 3/whiskas/Course/CourseLib/core/gameobject.h
- E:/Gitin repo/ohjelmointi 3/whiskas/Course/CourseLib/core/gameobject.cpp

6.20 Whiskas::GameScene Class Reference

Inheritance diagram for Whiskas::GameScene:



Collaboration diagram for Whiskas::GameScene:



Public Member Functions

- **GameScene** (QWidget *qt_parent=nullptr, int width=10, int height=10, int scale=50)
Constructor.
- **~GameScene** ()=default
Default destructor.
- void **setSize** (int width, int height)
Sets the map size.
- void **setScale** (int scale)
sets the size of individual tiles (=scale)
- void **resize** ()
resize recalculates the bounding rectangle
- void **drawItem** (std::shared_ptr< Course::GameObject > obj)
draw a new item to the map.
- void **removeItem** (std::shared_ptr< Course::GameObject > obj)
tries to remove drawn object at the location obj points to. If there's multiple objects, will remove the one that matches obj.
- void **updateItem** (std::shared_ptr< Course::GameObject > obj)
updates the position of obj.
- virtual bool **event** (QEvent *event) override
simple event handler that notifies when objects or the play area is clicked.
- int **getLastID** ()
return the ID of the last object clicked
- std::shared_ptr< Course::Coordinate > **getLastCoordinate** ()
return the coordinate of the last region clicked

6.20.1 Constructor & Destructor Documentation

6.20.1.1 GameScene()

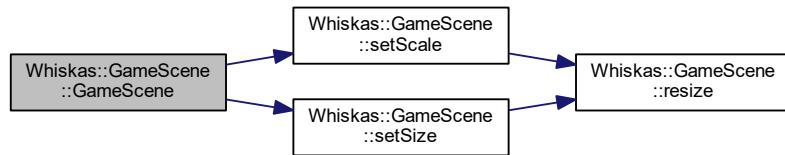
```
Whiskas::GameScene::GameScene (
    QWidget * qt_parent = nullptr,
    int width = 10,
    int height = 10,
    int scale = 50 )
```

Constructor.

Parameters

<i>qt_parent</i>	points to the parent object per Qt's parent-child-system.
<i>width</i>	in tiles for the game map.
<i>height</i>	in tiles for the game map.
<i>scale</i>	is the size in pixels of a single square tile.

Here is the call graph for this function:



6.20.2 Member Function Documentation

6.20.2.1 drawItem()

```
void Whiskas::GameScene::drawItem (
    std::shared_ptr< Course::GameObject > obj )
```

draw a new item to the map.

Parameters

<i>obj</i>	shared ptr to the object
------------	--------------------------

6.20.2.2 event()

```
bool Whiskas::GameScene::event (
    QEvent * event ) [override], [virtual]
```

simple event handler that notifies when objects or the play area is clicked.

Parameters

<i>event</i>	that has happened.
--------------	--------------------

Returns

True: if event was handled in the handler. False: if the event handling was passed over.

6.20.2.3 getLastCoordinate()

```
std::shared_ptr< Course::Coordinate > Whiskas::GameScene::getLastCoordinate ( )
```

return the coordinate of the last region clicked

Returns

coordinate

6.20.2.4 getLastID()

```
int Whiskas::GameScene::getLastID ( )
```

return the ID of the last object clicked

Returns

object's ID

6.20.2.5 removeItem()

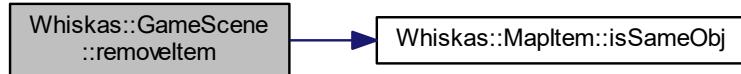
```
void Whiskas::GameScene::removeItem (   
    std::shared_ptr< Course::GameObject > obj )
```

tries to remove drawn object at the location obj points to. If there's multiple objects, will remove the one that matches obj.

Parameters

<i>obj</i>	shared ptr to the object being deleted
------------	--

Here is the call graph for this function:



6.20.2.6 setScale()

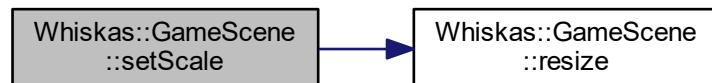
```
void Whiskas::GameScene::setScale ( int scale )
```

sets the size of individual tiles (=scale)

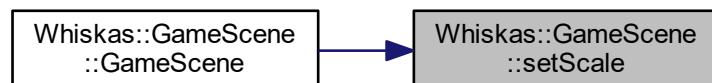
Parameters

scale	in pixels
-------	-----------

Here is the call graph for this function:



Here is the caller graph for this function:



6.20.2.7 setSize()

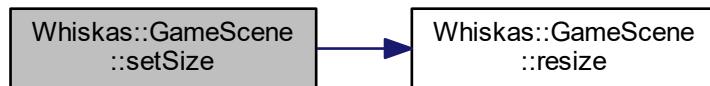
```
void Whiskas::GameScene::setSize (
    int width,
    int height )
```

Sets the map size.

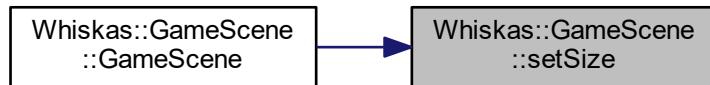
Parameters

<i>width</i>	(number of tiles)
<i>height</i>	(number of tiles)

Here is the call graph for this function:



Here is the caller graph for this function:



6.20.2.8 updateItem()

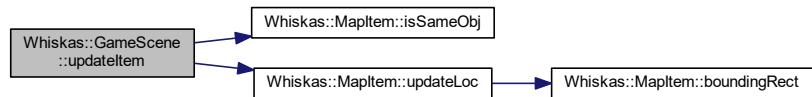
```
void Whiskas::GameScene::updateItem (
    std::shared_ptr< Course::GameObject > obj )
```

updates the position of obj.

Parameters

<i>obj</i>	shared ptr to the obj being updated.
------------	--------------------------------------

Here is the call graph for this function:



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

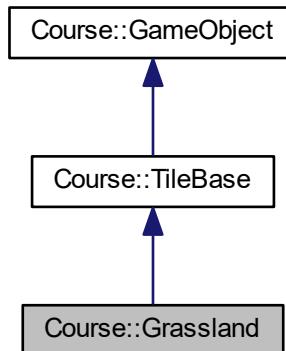
- E:/Gitin repo/ohjelmointi 3/whiskas/Game/graphics/gamescene.h
- E:/Gitin repo/ohjelmointi 3/whiskas/Game/graphics/gamescene.cpp

6.21 Course::Grassland Class Reference

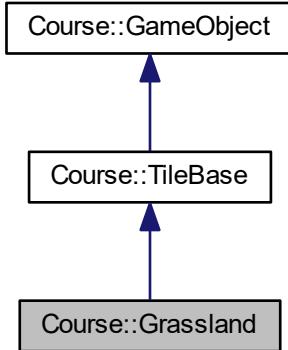
The [Grassland](#) class represents [Grassland](#) in the gameworld.

```
#include <grassland.h>
```

Inheritance diagram for Course::Grassland:



Collaboration diagram for Course::Grassland:



Public Member Functions

- [Grassland \(\)=delete](#)
Disabled parameterless constructor.
- [Grassland \(const Coordinate &location, const std::shared_ptr< iGameEventHandler > &eventhandler, const std::shared_ptr< iObjectManager > &objectmanager, const unsigned int &max_build=3, const unsigned int &max_work=3, const ResourceMap &production=ConstResourceMaps::GRASSLAND_BP\)](#)
Constructor for the class.
- virtual [~Grassland \(\)=default](#)
Default destructor.
- virtual std::string [getType \(\) const override](#)
getType Returns a string describing objects type. This should be overriden in each inherited class. Makes checking object's type easier for students.

Static Public Attributes

- static const unsigned int **MAX_BUILDINGS**
- static const unsigned int **MAX_WORKERS**
- static const ResourceMap **BASE_PRODUCTION**

Additional Inherited Members

6.21.1 Detailed Description

The [Grassland](#) class represents [Grassland](#) in the gameworld.

[Grassland](#) has [BasicProduction](#):

- Money = 2
- Food = 5
- Wood = 1
- Stone = 1
- Ore = 0

Functionality follows mainly the parent class' functionality.

Tile supports 3 buildings.

6.21.2 Constructor & Destructor Documentation

6.21.2.1 Grassland()

```
Course::Grassland::Grassland (
    const Coordinate & location,
    const std::shared_ptr< iGameEventHandler > & eventhandler,
    const std::shared_ptr< iObjectManager > & objectmanager,
    const unsigned int & max_build = 3,
    const unsigned int & max_work = 3,
    const ResourceMap & production = ConstResourceMaps::GRASSLAND_BP )
```

Constructor for the class.

Parameters

<i>location</i>	is the Coordinate where the Tile is located in the game.
<i>eventhandler</i>	points to the student's GameEventHandler.

6.21.3 Member Function Documentation

6.21.3.1 getType()

```
std::string Course::Grassland::getType ( ) const [override], [virtual]
```

`getType` Returns a string describing objects type. This should be overriden in each inherited class. Makes checking object's type easier for students.

Returns

`std::string` that represents Object's type.

Postcondition

Exception guarantee: No-throw

Note

You can use this in e.g. debugging and similar printing.

Reimplemented from [Course::GameObject](#).

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

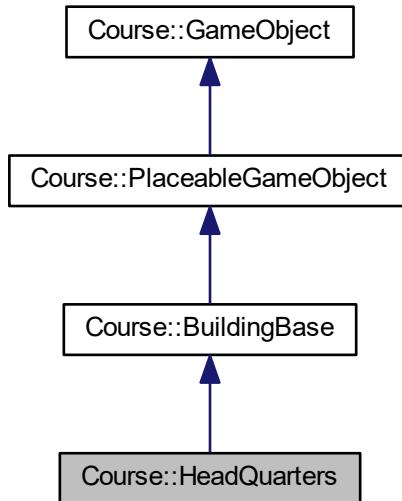
- E:/Gitin repo/ohjelmointi 3/whiskas/Course/CourseLib/tiles/grassland.h
- E:/Gitin repo/ohjelmointi 3/whiskas/Course/CourseLib/tiles/grassland.cpp

6.22 Course::HeadQuarters Class Reference

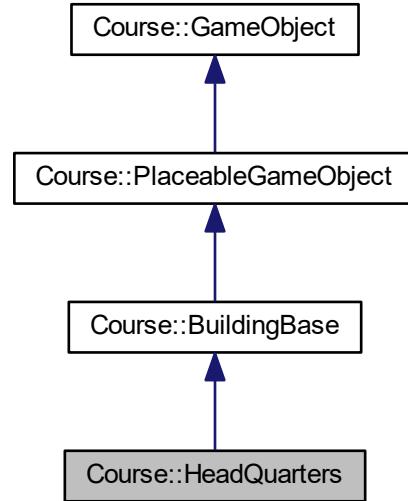
The [HeadQuarters](#) class represents a player's HeadQuarters-building.

```
#include <headquarters.h>
```

Inheritance diagram for Course::HeadQuarters:



Collaboration diagram for Course::HeadQuarters:



Public Member Functions

- [HeadQuarters \(\)=delete](#)
Disabled parameterless constructor.
- [HeadQuarters \(const std::shared_ptr< iGameEventHandler > &eventhandler, const std::shared_ptr< iObjectManager > &objectmanager, const std::shared_ptr< PlayerBase > &owner, const int &tilespaces=1, const ResourceMap &buildcost=ConstResourceMaps::HQ_BUILD_COST, const ResourceMap &production=ConstResourceMaps::HQ_PRODUCTION\)](#)
Constructor for the class.
- [~HeadQuarters \(\) override=default](#)
Default destructor.
- [std::string getType \(\) const override](#)
getType Returns a string describing objects type. This should be overriden in each inherited class. Makes checking object's type easier for students.
- [virtual void onBuildAction \(\) override](#)
Sets neighbouring Tiles' ownership to this building's ownership in 3 tile-radius, if the Tiles don't already have an owner.

Additional Inherited Members

6.22.1 Detailed Description

The [HeadQuarters](#) class represents a player's HeadQuarters-building.

It can be constructed on any tile that has not been claimed by any other player.

Effects: Claims surrounding unclaimed tiles.

Radius: 3 tiles.

6.22.2 Constructor & Destructor Documentation

6.22.2.1 HeadQuarters()

```
Course::HeadQuarters::HeadQuarters (
    const std::shared_ptr< iGameEventHandler > & eventhandler,
    const std::shared_ptr< iObjectManager > & objectmanager,
    const std::shared_ptr< PlayerBase > & owner,
    const int & tilespaces = 1,
    const ResourceMap & buildcost = ConstResourceMaps::HQ_BUILD_COST,
    const ResourceMap & production = ConstResourceMaps::HQ_PRODUCTION ) [explicit]
```

Constructor for the class.

Parameters

<i>eventhandler</i>	points to the student's GameEventHandler.
<i>owner</i>	points to the owning player.
<i>tile</i>	points to the tile upon which the building is constructed.

Postcondition

Exception Guarantee: No guarantee.

Exceptions

<i>OwnerConflict</i>	- if the building conflicts with tile's ownership.
----------------------	--

6.22.3 Member Function Documentation

6.22.3.1 getType()

```
std::string Course::HeadQuarters::getType ( ) const [override], [virtual]
```

getType Returns a string describing objects type. This should be overriden in each inherited class. Makes checking object's type easier for students.

Returns

std::string that represents Object's type.

Postcondition

Exception guarantee: No-throw

Note

You can use this in e.g. debugging and similar printing.

Reimplemented from [Course::BuildingBase](#).

6.22.3.2 onBuildAction()

```
void Course::HeadQuarters::onBuildAction ( ) [override], [virtual]
```

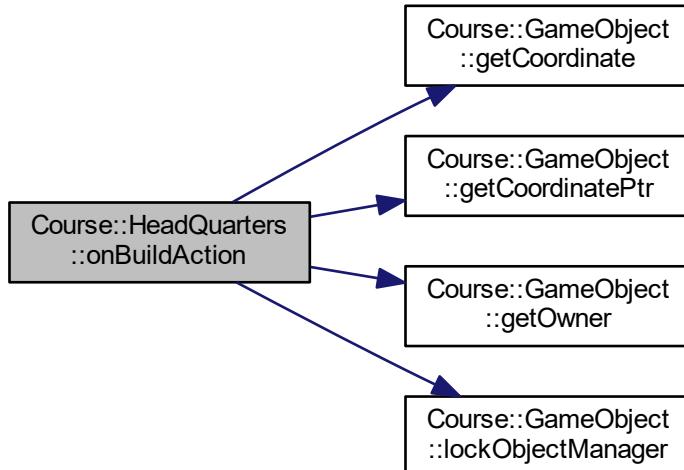
Sets neighbouring Tiles' ownership to this building's ownership in 3 tile-radius, if the Tiles don't already have an owner.

Postcondition

Exception guarantee: Basic

Reimplemented from [Course::BuildingBase](#).

Here is the call graph for this function:



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

- E:/Gitin repo/ohjelmointi 3/whiskas/Course/CourseLib/buildings/headquarters.h
- E:/Gitin repo/ohjelmointi 3/whiskas/Course/CourseLib/buildings/headquarters.cpp

6.23 Course::iGameEventHandler Class Reference

The [iGameEventHandler](#) class is an interface which the Course-side code uses to interact with the GameEvent Handler implemented by the students.

```
#include <igameeventhandler.h>
```

Inheritance diagram for Course::iGameEventHandler:



Public Member Functions

- virtual [~iGameEventHandler \(\)=default](#)
Default destructor.
- virtual bool [modifyResource \(std::shared_ptr< PlayerBase > player, BasicResource resource, int amount\)=0](#)
Modify Player's resource. Can be used to both sum or subtract.
- virtual bool [modifyResources \(std::shared_ptr< PlayerBase > player, ResourceMap resources\)=0](#)
Modify Player's resources. Can be used to both sum or subtract.

6.23.1 Detailed Description

The [iGameEventHandler](#) class is an interface which the Course-side code uses to interact with the GameEvent Handler implemented by the students.

Note

The interface declares only functions required by the Course-side code. The actual implementation can (and should!) contain more stuff.

In a "real" project, the GameEventHandler should be a singleton and not use an abstract base class to define the interface for it. **This design was chosen merely for pedagogical reasons and to give students more freedom in their project design.**

6.23.2 Member Function Documentation

6.23.2.1 **modifyResource()**

```
virtual bool Course::iGameEventHandler::modifyResource (
    std::shared_ptr< PlayerBase > player,
    BasicResource resource,
    int amount ) [pure virtual]
```

Modify Player's resource. Can be used to both sum or subtract.

Parameters

<i>player</i>	Pointer to the Player whose resource is being modified.
<i>resource</i>	Defines the modified resource.
<i>amount</i>	Defines the amount of change.

Postcondition

Exception guarantee: Basic

Returns

True - Modification was successful.
False - Modification failed.

Implemented in [Whiskas::gameEventHandler](#).

6.23.2.2 modifyResources()

```
virtual bool Course::iGameEventHandler::modifyResources (
    std::shared_ptr< PlayerBase > player,
    ResourceMap resources ) [pure virtual]
```

Modify Player's resources. Can be used to both sum or subtract.

Parameters

<i>player</i>	Pointer to the Player whose resources are being modified.
<i>resources</i>	ResourceMap containing change amounts.

Returns

True - Modification was successful.
False - Modification failed.

Implemented in [Whiskas::gameEventHandler](#).

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

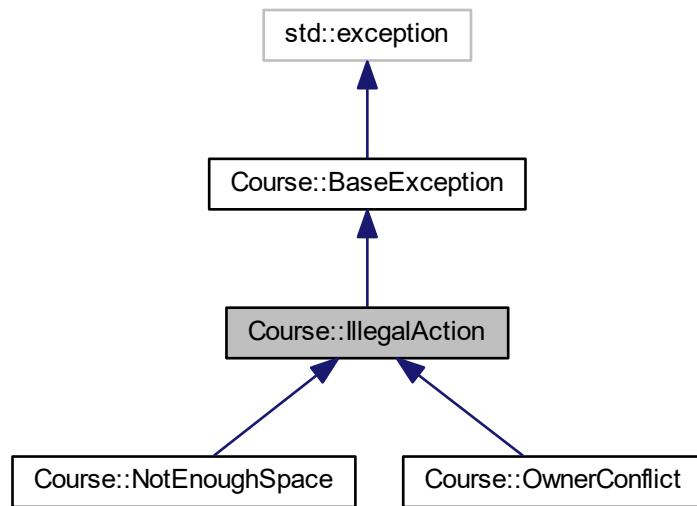
- E:/Gitin repo/ohjelmointi 3/whiskas/Course/CourseLib/interfaces/igameeventhandler.h

6.24 Course::IllegalAction Class Reference

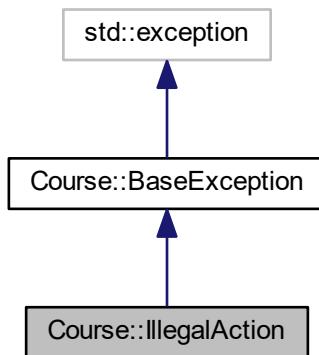
The [IllegalAction](#) exception is usually used in cases, where an illegal game action was attempted.

```
#include <illegalaction.h>
```

Inheritance diagram for Course::IllegalAction:



Collaboration diagram for Course::IllegalAction:



Public Member Functions

- `IllegalAction (const std::string &msg="")`
Exception Constructor.
- virtual `~IllegalAction ()=default`
~Exception Default destructor

6.24.1 Detailed Description

The `IllegalAction` exception is usually used in cases, where an illegal game action was attempted.

6.24.2 Constructor & Destructor Documentation

6.24.2.1 `IllegalAction()`

```
Course::IllegalAction::IllegalAction (
    const std::string & msg = "" ) [inline], [explicit]
```

Exception Constructor.

Parameters

<code>msg</code>	<code>std::string</code> describing the reason for exception.
------------------	---

6.24.2.2 `~IllegalAction()`

```
virtual Course::IllegalAction::~IllegalAction ( ) [virtual], [default]
```

~Exception Default destructor

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

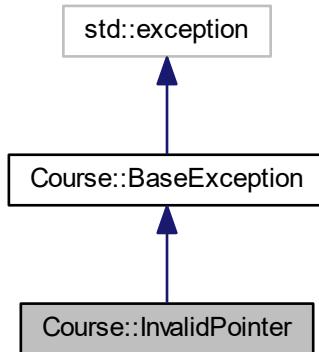
- E:/Gitin repo/ohjelmointi 3/whiskas/Course/CourseLib/exceptions/illegalaction.h

6.25 Course::InvalidPointer Class Reference

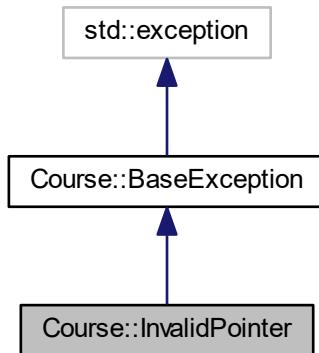
The `InvalidPointer` exception is usually used in cases, where data can't be accessed through a pointer.

```
#include <invalidpointer.h>
```

Inheritance diagram for Course::InvalidPointer:



Collaboration diagram for Course::InvalidPointer:



Public Member Functions

- [InvalidPointer](#) (const std::string &msg="")
Exception Constructor.
- virtual [~InvalidPointer](#) ()=default
~Exception Default destructor

6.25.1 Detailed Description

The [InvalidPointer](#) exception is usually used in cases, where data can't be accessed through a pointer.

6.25.2 Constructor & Destructor Documentation

6.25.2.1 InvalidPointer()

```
Course::InvalidPointer::InvalidPointer (
    const std::string & msg = "" ) [inline], [explicit]
```

Exception Constructor.

Parameters

<i>msg</i>	std::string describing the reason for exception.
------------	--

6.25.2.2 ~InvalidPointer()

```
virtual Course::InvalidPointer::~InvalidPointer () [virtual], [default]
```

~Exception Default destructor

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

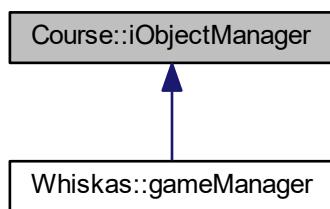
- E:/Gitin repo/ohjelmointi 3/whiskas/Course/CourseLib/exceptions/invalidpointer.h

6.26 Course::iObjectManager Class Reference

The [iObjectManager](#) class is an interface which the Course-side code uses to interact with the ObjectManager implemented by the students.

```
#include <iobjectmanager.h>
```

Inheritance diagram for Course::iObjectManager:



Public Member Functions

- virtual ~[iObjectManager](#) ()=default
Default destructor.
- virtual void [addTiles](#) (const std::vector< std::shared_ptr< [TileBase](#) >> &tiles)=0
Adds new tiles to the ObjectManager.
- virtual std::shared_ptr< [TileBase](#) > [getTile](#) (const [Coordinate](#) &coordinate)=0
Returns a shared pointer to a Tile that has specified coordinate.
- virtual std::shared_ptr< [TileBase](#) > [getTile](#) (const [ObjectID](#) &id)=0
Returns a shared pointer to a Tile that has specified ID.
- virtual std::vector< std::shared_ptr< [TileBase](#) > > [getTiles](#) (const std::vector< [Coordinate](#) > &coordinates)=0
Returns a vector of shared pointers to Tiles specified by a vector of Coordinates.

6.26.1 Detailed Description

The [iObjectManager](#) class is an interface which the Course-side code uses to interact with the ObjectManager implemented by the students.

Note

The interface declares only functions required by the Course-side code. The actual implementation can (and should!) contain more stuff.

6.26.2 Member Function Documentation

6.26.2.1 [addTiles\(\)](#)

```
virtual void Course::iObjectManager::addTiles (
    const std::vector< std::shared_ptr< TileBase >> & tiles ) [pure virtual]
```

Adds new tiles to the ObjectManager.

Parameters

<i>tiles</i>	a vector that contains the Tiles to be added.
--------------	---

Postcondition

The tile-pointers in the vector are stored in the ObjectManager. Exception Guarantee: Basic

6.26.2.2 [getTile\(\)](#) [1/2]

```
virtual std::shared_ptr<TileBase> Course::iObjectManager::getTile (
    const Coordinate & coordinate ) [pure virtual]
```

Returns a shared pointer to a Tile that has specified coordinate.

Parameters

<i>coordinate</i>	Requested Tile's Coordinate
-------------------	-----------------------------

Returns

a pointer to a Tile that has the given coordinate. If no for the coordinate exists, return empty pointer.

Postcondition

Exception Guarantee: Basic

Implemented in [Whiskas::gameManager](#).

6.26.2.3 getTile() [2/2]

```
virtual std::shared_ptr<TileBase> Course::iObjectManager::getTile (
    const ObjectId & id ) [pure virtual]
```

Returns a shared pointer to a Tile that has specified ID.

Parameters

<i>id</i>	Tile's ObjectId.
-----------	------------------

Returns

a pointer to the Tile that has the given ID If no for the id exists, return empty pointer.

Postcondition

Exception Guarantee: Basic

Implemented in [Whiskas::gameManager](#).

6.26.2.4 getTiles()

```
virtual std::vector<std::shared_ptr<TileBase>> Course::iObjectManager::getTiles (
    const std::vector< Coordinate > & coordinates ) [pure virtual]
```

Returns a vector of shared pointers to Tiles specified by a vector of Coordinates.

Parameters

<i>coordinates</i>	a vector of Coordinates for the requested Tiles
--------------------	---

Returns

Vector of that contains pointers to Tile's that match the coordinates. The vector is empty if no matches were made.

Postcondition

Exception Guarantee: Basic

Implemented in [Whiskas::gameManager](#).

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

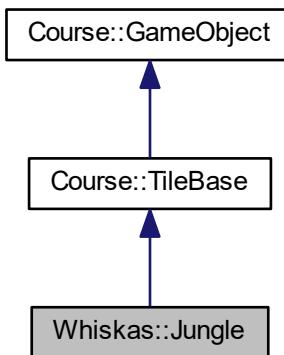
- E:/Gitin repo/ohjelmointi 3/whiskas/Course/CourseLib/interfaces/iobjectmanager.h

6.27 Whiskas::Jungle Class Reference

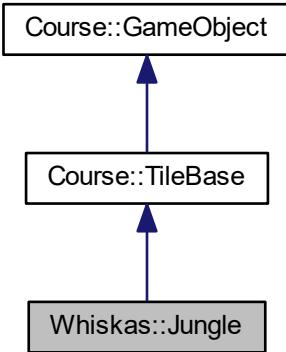
The [Jungle](#) class represents the jungle in the gameworld. The jungle supports ranged altars and sawmills.

```
#include <jungle.h>
```

Inheritance diagram for Whiskas::Jungle:



Collaboration diagram for Whiskas::Jungle:



Public Member Functions

- **Jungle** (const [Course::Coordinate](#) &location, const std::shared_ptr<[Course::iGameEventHandler](#)> &eventhandler, const std::shared_ptr<[Course::iObjectManager](#)> &objectmanager, const unsigned int &max_build=1, const unsigned int &max_work=1, const [Course::ResourceMap](#) &production={})
- std::string [getType \(\)](#) const override

Additional Inherited Members

6.27.1 Detailed Description

The [Jungle](#) class represents the jungle in the gameworld. The jungle supports ranged altars and sawmills.

6.27.2 Member Function Documentation

6.27.2.1 [getType\(\)](#)

```
std::string Whiskas::Jungle::getType ( ) const [override], [virtual]
```

Reimplemented from [Course::TileBase](#).

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

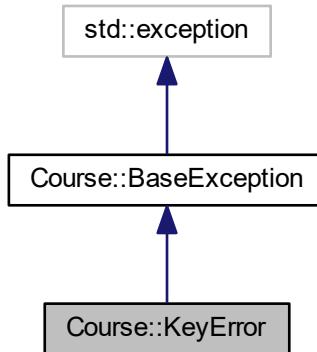
- E:/Gitin repo/ohjelmointi 3/whiskas/Game/tiles/jungle.h
- E:/Gitin repo/ohjelmointi 3/whiskas/Game/tiles/jungle.cpp

6.28 Course::KeyError Class Reference

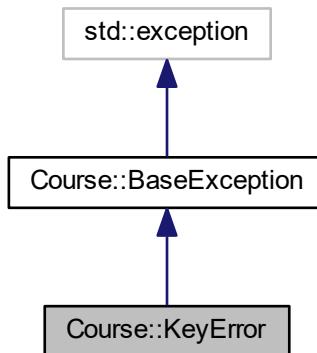
The [KeyError](#) class is an Exception-class for cases where the used key is invalid.

```
#include <keyerror.h>
```

Inheritance diagram for Course::KeyError:



Collaboration diagram for Course::KeyError:



Public Member Functions

- [KeyError](#) (const std::string &*msg*="")
Exception Constructor.
- virtual [~KeyError](#) ()=default
~Exception Default destructor

6.28.1 Detailed Description

The [KeyError](#) class is an Exception-class for cases where the used key is invalid.

6.28.2 Constructor & Destructor Documentation

6.28.2.1 KeyError()

```
Course::KeyError::KeyError (
    const std::string & msg = "" ) [inline], [explicit]
```

Exception Constructor.

Parameters

<i>msg</i>	std::string describing the reason for exception.
------------	--

6.28.2.2 ~KeyError()

```
virtual Course::KeyError::~KeyError ( ) [virtual], [default]
```

~Exception Default destructor

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

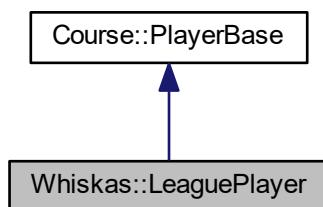
- E:/Gitin repo/ohjelmointi 3/whiskas/Course/CourseLib/exceptions/keyerror.h

6.29 Whiskas::LeaguePlayer Class Reference

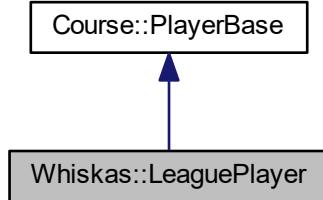
The [LeaguePlayer](#) class The playerClass for this game.

```
#include <leagueplayer.h>
```

Inheritance diagram for Whiskas::LeaguePlayer:



Collaboration diagram for Whiskas::LeaguePlayer:



Public Member Functions

- [LeaguePlayer](#) (const std::string &name, AdvancedResourceMap playerItems=PLAYER_ITEMS)
LeaguePlayer Constructor.
- AdvancedResourceMap [getItems](#) ()
- void [setPlayerItems](#) (AdvancedResourceMap toSet)
setPlayerItems sets the player items to be equal to the parameter map

6.29.1 Detailed Description

The [LeaguePlayer](#) class The playerClass for this game.

6.29.2 Constructor & Destructor Documentation

6.29.2.1 LeaguePlayer()

```
Whiskas::LeaguePlayer::LeaguePlayer (
    const std::string & name,
    AdvancedResourceMap playerItems = PLAYER_ITEMS )
```

[LeaguePlayer](#) Constructor.

Parameters

<i>name</i>	
<i>playerItems</i>	Resource the player has at the start of the game

6.29.3 Member Function Documentation

6.29.3.1 setPlayerItems()

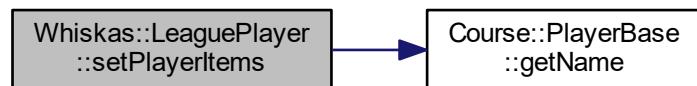
```
void Whiskas::LeaguePlayer::setPlayerItems (AdvancedResourceMap toSet )
```

setPlayerItems sets the player items to be equal to the parameter map

Parameters

<i>toSet</i>	New items
--------------	-----------

Here is the call graph for this function:



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

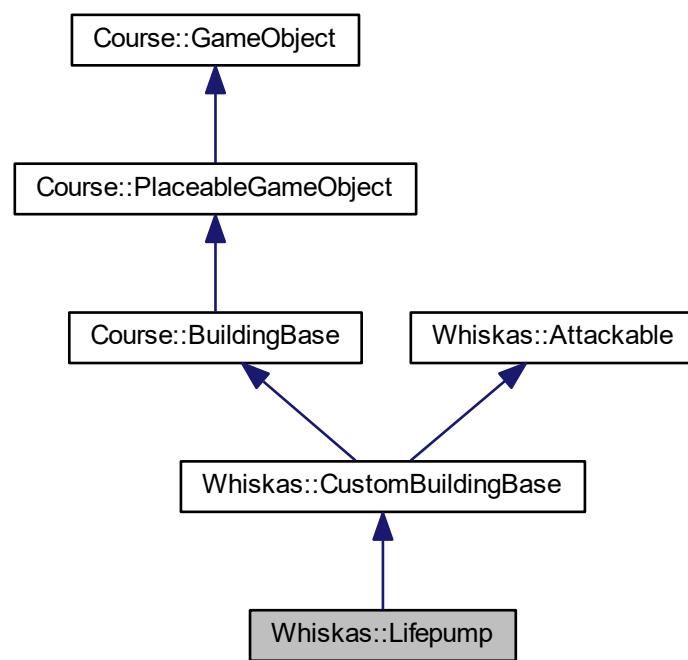
- E:/Gitin repo/ohjelmointi 3/whiskas/Game/leagueplayer.h
- E:/Gitin repo/ohjelmointi 3/whiskas/Game/leagueplayer.cpp

6.30 Whiskas::Lifepump Class Reference

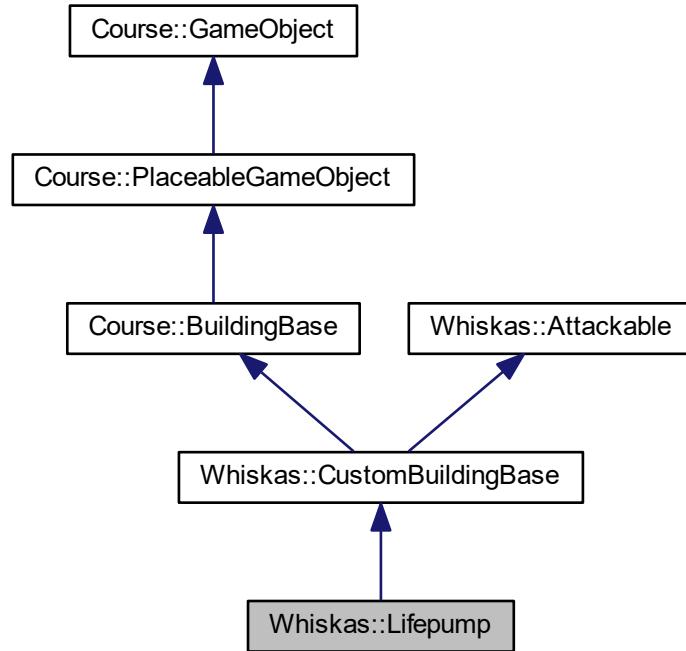
The [Lifepump](#) class represents a lifepump-building in the game.

```
#include <lifepump.h>
```

Inheritance diagram for Whiskas::Lifepump:



Collaboration diagram for Whiskas::Lifepump:



Public Member Functions

- `Lifepump (const std::shared_ptr< gameEventHandler > &eventhandler, const std::shared_ptr< gameManager > &objectmanager, const std::shared_ptr< Course::PlayerBase > &owner, const int &tilespaces=1, const AdvancedResourceMap &buildcost=LIFEPUmp_COST, const AdvancedResourceMap &production=LIFEPUmp_PRODUCE, int health=5, int attack=0)`
- `~Lifepump () override=default`
Default destructor.
- `std::string getType () const override`

Additional Inherited Members

6.30.1 Detailed Description

The `Lifepump` class represents a lifepump-building in the game.

The lifepump produces 1 lifewater per turn Can only be built on a spring

6.30.2 Member Function Documentation

6.30.2.1 `getType()`

```
std::string Whiskas::LifePump::getType () const [override], [virtual]
```

Reimplemented from [Course::BuildingBase](#).

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

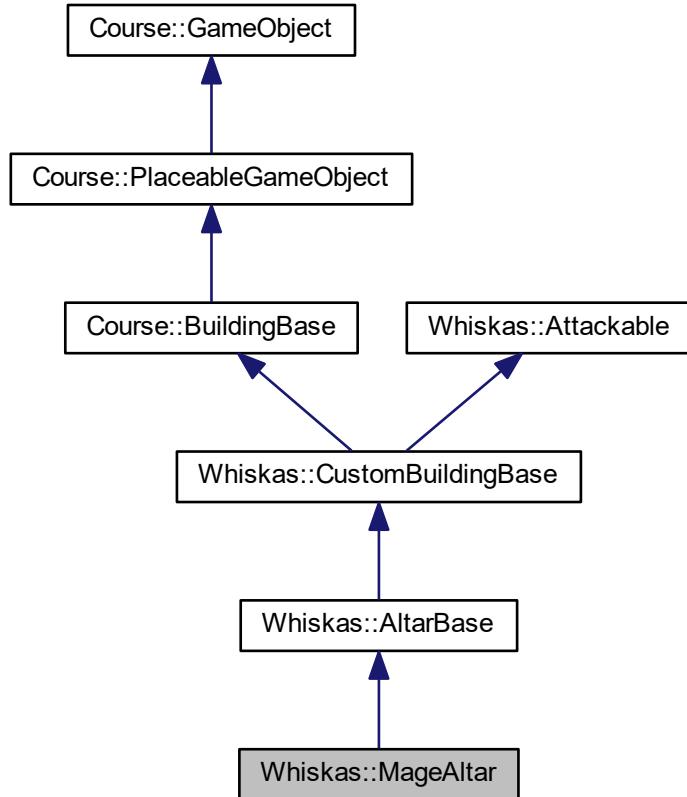
- E:/Gitin repo/ohjelmointi 3/whiskas/Game/buildings/lifePump.h
- E:/Gitin repo/ohjelmointi 3/whiskas/Game/buildings/lifePump.cpp

6.31 Whiskas::MageAltar Class Reference

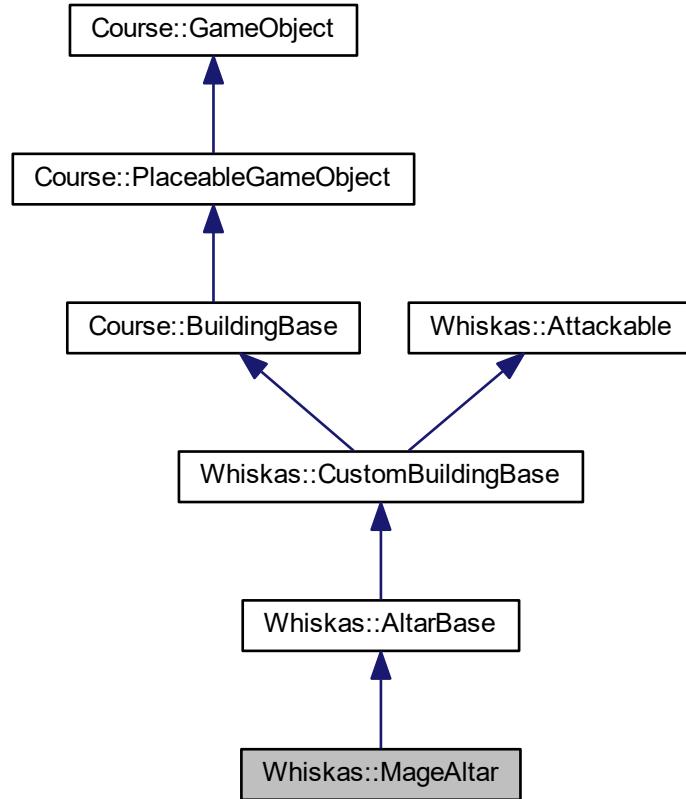
The [MageAltar](#) class is an altar, that upgrades minions to mages.

```
#include <magealtar.h>
```

Inheritance diagram for Whiskas::MageAltar:



Collaboration diagram for Whiskas::MageAltar:



Public Member Functions

- `MageAltar (const std::shared_ptr< gameEventHandler > &eventhandler, const std::shared_ptr< gameManager > &objectmanager, const std::shared_ptr< Course::PlayerBase > &owner)`
- void `upgradeMinion ()` override
upgradeMinion upgrades a minion on the tile when called.
- `std::string getType () const` override

Additional Inherited Members

6.31.1 Detailed Description

The `MageAltar` class is an altar, that upgrades minions to mages.

6.31.2 Member Function Documentation

6.31.2.1 `getType()`

```
std::string Whiskas::MageAltar::getType ( ) const [override], [virtual]
```

Reimplemented from [Whiskas::AltarBase](#).

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

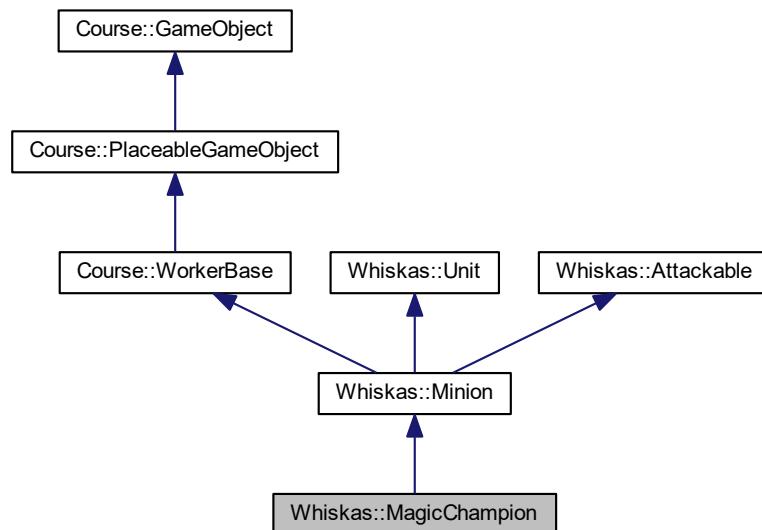
- E:/Gitin repo/ohjelmointi 3/whiskas/Game/buildings/magealtar.h
- E:/Gitin repo/ohjelmointi 3/whiskas/Game/buildings/magealtar.cpp

6.32 Whiskas::MagicChampion Class Reference

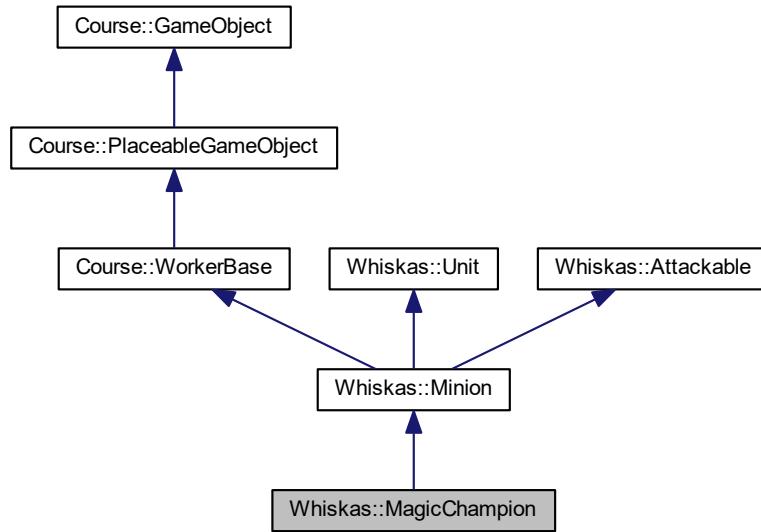
The [MagicChampion](#) class Subminion, has a higher dmg and an area attack.

```
#include <magicchampion.h>
```

Inheritance diagram for Whiskas::MagicChampion:



Collaboration diagram for Whiskas::MagicChampion:



Public Member Functions

- `MagicChampion (const std::shared_ptr< Course::iGameEventHandler > &handler, const std::shared_ptr< Course::IObjectManager > &manager, const std::shared_ptr< Course::PlayerBase > &owner, int movement=2, int health=2, int attack=5, int number_of_attacks=1)`
- `std::string getType () const override`

Additional Inherited Members

6.32.1 Detailed Description

The `MagicChampion` class Subminion, has a higher dmg and an area attack.

6.32.2 Member Function Documentation

6.32.2.1 `getType()`

```
std::string Whiskas::MagicChampion::getType ( ) const [override], [virtual]
```

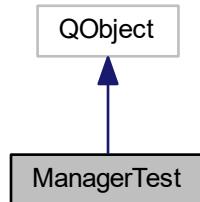
Reimplemented from `Course::WorkerBase`.

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

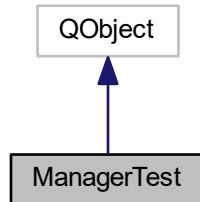
- E:/Gitin repo/ohjelmointi 3/whiskas/Game/minion/magicchampion.h
- E:/Gitin repo/ohjelmointi 3/whiskas/Game/minion/magicchampion.cpp

6.33 ManagerTest Class Reference

Inheritance diagram for ManagerTest:



Collaboration diagram for ManagerTest:



Public Member Functions

- **ManagerTest** (QObject *parent=nullptr)

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

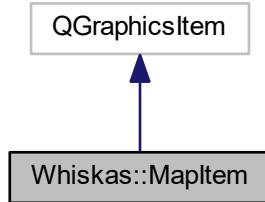
- E:/Gitin repo/ohjelmointi 3/whiskas/UnitTests/unittestfolder/managertest.h
- E:/Gitin repo/ohjelmointi 3/whiskas/UnitTests/unittestfolder/managertest.cpp

6.34 Whiskas::MapItem Class Reference

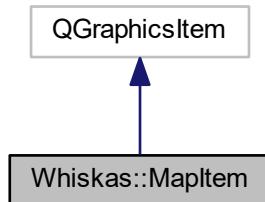
The [MapItem](#) class is a custom QGraphicsItem that acts as a single GameObject's graphical element.

```
#include <mapitem.h>
```

Inheritance diagram for Whiskas::MapItem:



Collaboration diagram for Whiskas::MapItem:



Public Member Functions

- **MapItem** (std::shared_ptr<Course::GameObject> obj, int size)
Constructor.
- QRectF **boundingRect** () const override
boundingRect
- void **paint** (QPainter *painter, const QStyleOptionGraphicsItem *option, QWidget *widget) override
paints the item
- const std::shared_ptr<Course::GameObject> & **getBoundObject** ()
getBoundObject
- void **updateLoc** ()
updateLoc moves the item if the position has changed.
- bool **isSameObj** (const std::shared_ptr<Course::GameObject> &obj)
checks if this instance has obj as bound obj.
- int **getSize** () const
getSize
- void **setSize** (int size)
setSize

6.34.1 Detailed Description

The [MapItem](#) class is a custom QGraphicsItem that acts as a single GameObject's graphical element.

6.34.2 Constructor & Destructor Documentation

6.34.2.1 MapItem()

```
Whiskas::MapItem::MapItem (
    std::shared_ptr< Course::GameObject > obj,
    int size )
```

Constructor.

Parameters

<i>obj</i>	shared_ptr to the obj.
<i>size</i>	of the created item in pixels.

Precondition

obj must have a valid Coordinate.

6.34.3 Member Function Documentation

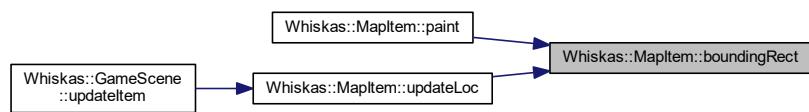
6.34.3.1 boundingRect()

```
QRectF Whiskas::MapItem::boundingRect ( ) const [override]  
boundingRect
```

Returns

the bounding rectangle of this item.

Here is the caller graph for this function:



6.34.3.2 getBoundObject()

```
const std::shared_ptr< Course::GameObject > & Whiskas::MapItem::getBoundObject ( )  
  
getBoundObject
```

Returns

the object this item is bound to.

6.34.3.3 isSameObj()

```
bool Whiskas::MapItem::isSameObj (   
    const std::shared_ptr< Course::GameObject > & obj )
```

checks if this instance has obj as bound obj.

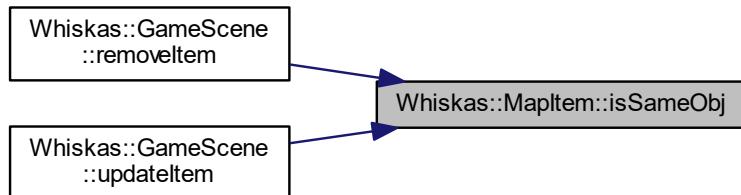
Parameters

<i>obj</i>	to compare to.
------------	----------------

Returns

True: if obj is pointing to the same object as this item. False otherwise.

Here is the caller graph for this function:



6.34.3.4 paint()

```
void Whiskas::MapItem::paint (   
     QPainter * painter,  
     const QStyleOptionGraphicsItem * option,  
     QWidget * widget ) [override]
```

paints the item

Parameters

<i>painter</i>	
<i>option</i>	
<i>widget</i>	

Note

The GraphicsView containing the scene this belongs to usually calls this function

Here is the call graph for this function:

**6.34.3.5 setSize()**

```
void Whiskas::MapItem::setSize (
    int size )
```

setSize**Parameters**

<i>size</i>	of the object in pixels.
-------------	--------------------------

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

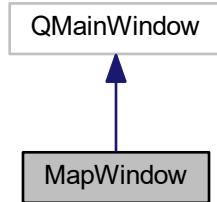
- E:/Gitin repo/ohjelmointi 3/whiskas/Game/graphics/mapitem.h
- E:/Gitin repo/ohjelmointi 3/whiskas/Game/graphics/mapitem.cpp

6.35 MapWindow Class Reference

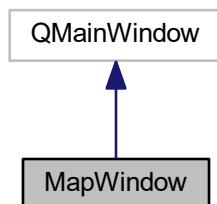
The [MapWindow](#) class Manages the UI interface.

```
#include <mapwindow.hh>
```

Inheritance diagram for MapWindow:



Collaboration diagram for MapWindow:



Public Slots

- void `initMap` (int x, int y)
initMap Generates the world, accoring to the size of x,y
- void `selectBuilding` (const std::string &type)
selectBuilding updates the type of building to be built
- void `closeWindow` ()
Closes the mainwidow.

Public Member Functions

- `MapWindow` (QWidget *parent=nullptr, std::shared_ptr<[Whiskas::gameEventHandler](#)> GEHandler={})
- void `setGEHandler` (std::shared_ptr<[Whiskas::gameEventHandler](#)> nHandler)
setGEHandler sets the current Game Handler
- void `setGManager` (std::shared_ptr<[Whiskas::gameManager](#)> manager)
setGManager sets the curretn game manager
- void `setSize` (int width, int height)
setSize sets the game size
- void `drawItem` (std::shared_ptr<[Course::GameObject](#)> obj)

- drawItem draws a new item to the map*
- void **mousePressEvent** (QMouseEvent *event) override
mousePressEvent calls event handler to manage the click and updates The description labels
- void **openBuildingDialog** (std::shared_ptr< Whiskas::gameManager > manager, std::shared_ptr< Whiskas::gameEventHandler > handler, std::shared_ptr< Course::PlayerBase > owner)
openBuildingDialog opens a dialog to select the building to be built
- void **updateDisplays** ()
updateDisplays updates resource icons
- void **generateLCDList** ()
generateLCDList makes a vector of the lcd widgets
- void **updateDescriptions** ()

6.35.1 Detailed Description

The [MapWindow](#) class Manages the UI interface.

6.35.2 Member Function Documentation

6.35.2.1 drawItem()

```
void MapWindow::drawItem (
    std::shared_ptr< Course::GameObject > obj )
```

drawItem draws a new item to the map

Parameters

<i>obj</i>	to be drawn
------------	-------------

6.35.2.2 initMap

```
void MapWindow::initMap (
    int x,
    int y ) [slot]
```

initMap Generates the world, accoring to the size of x,y

Parameters

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

6.35.2.3 mousePressEvent()

```
void MapWindow::mousePressEvent (
    QMouseEvent * event ) [override]
```

mousePressEvent calls event handler to manage the click and updates The description labels

Parameters

<i>event</i>	The click event
--------------	-----------------

6.35.2.4 openBuildingDialog()

```
void MapWindow::openBuildingDialog (
    std::shared_ptr< Whiskas::gameManager > manager,
    std::shared_ptr< Whiskas::gameEventHandler > handler,
    std::shared_ptr< Course::PlayerBase > owner )
```

openBuildingDialog opens a dialog to select the building to be built

Parameters

<i>manager</i>	ObjectManager that receives the building
<i>handler</i>	buildings handler
<i>owner</i>	player that owns the building

6.35.2.5 setGEHandler()

```
void MapWindow::setGEHandler (
    std::shared_ptr< Whiskas::gameEventHandler > nHandler )
```

setGEHandler sets the current Game Handler

Parameters

<i>nHandler</i>	the new game handler
-----------------	----------------------

6.35.2.6 setGManager()

```
void MapWindow::setGManager (
```

```
std::shared_ptr< Whiskas::gameManager > manager )
```

setGManager sets the current game manager

Parameters

<i>manager</i>	the new game manager
----------------	----------------------

6.35.2.7 setSize()

```
void MapWindow::setSize (
    int width,
    int height )
```

setSize sets the game size

Parameters

<i>width</i>	in tiles
<i>height</i>	int tiles

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

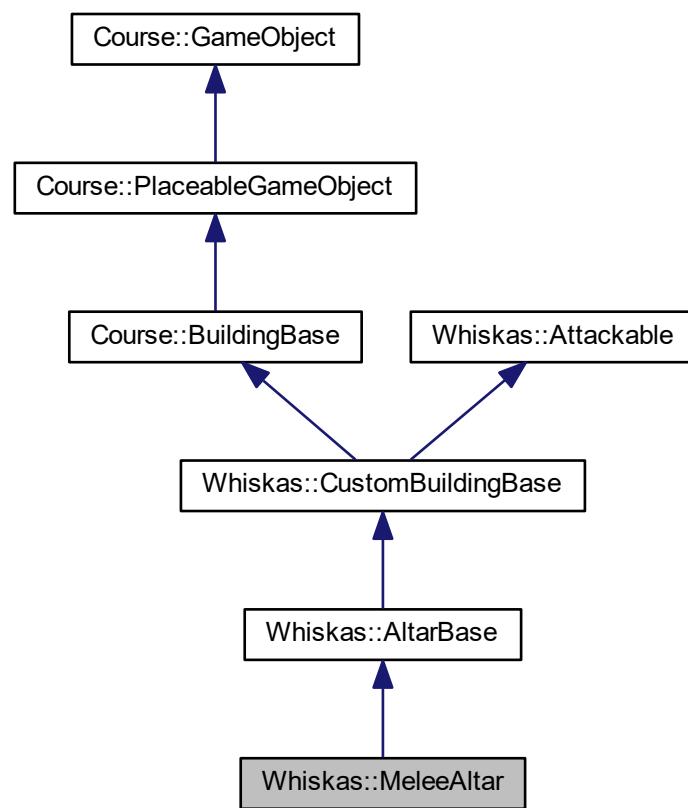
- E:/Gitin repo/ohjelmointi 3/whiskas/Game/mapwindow.hh
- E:/Gitin repo/ohjelmointi 3/whiskas/Game/mapwindow.cc

6.36 Whiskas::MeleeAltar Class Reference

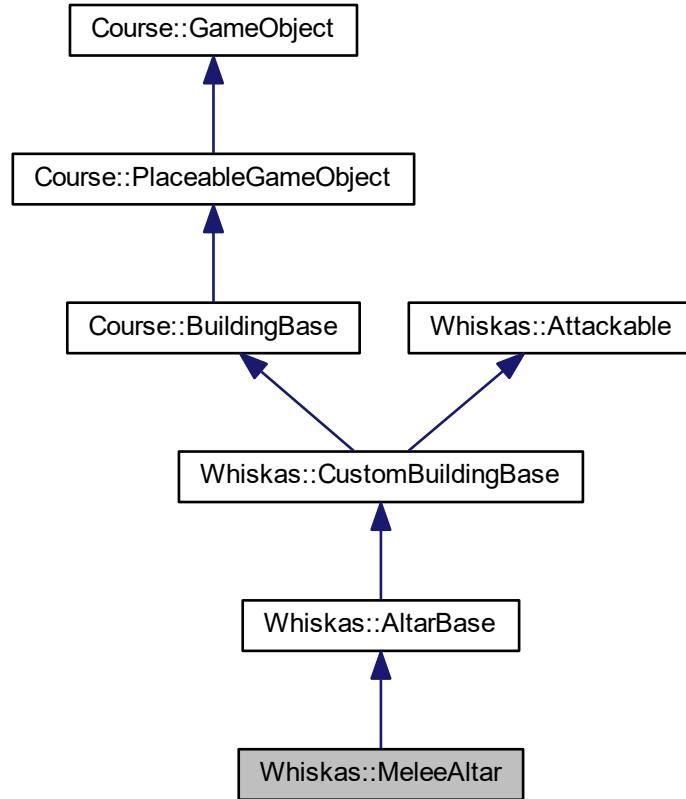
The [MeleeAltar](#) class is an altar that upgrades minions to melee champs.

```
#include <meleealtar.h>
```

Inheritance diagram for Whiskas::MeleeAltar:



Collaboration diagram for Whiskas::MeleeAltar:



Public Member Functions

- `MeleeAltar (const std::shared_ptr< gameEventHandler > &eventhandler, const std::shared_ptr< gameManager > &objectmanager, const std::shared_ptr< Course::PlayerBase > &owner)`
- void `upgradeMinion ()` override
upgradeMinion upgrades a minion on the tile when called.
- `std::string getType () const` override

Additional Inherited Members

6.36.1 Detailed Description

The [MeleeAltar](#) class is an altar that upgrades minions to melee champs.

6.36.2 Member Function Documentation

6.36.2.1 getType()

```
std::string Whiskas::MeleeAltar::getType ( ) const [override], [virtual]
```

Reimplemented from [Whiskas::AltarBase](#).

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

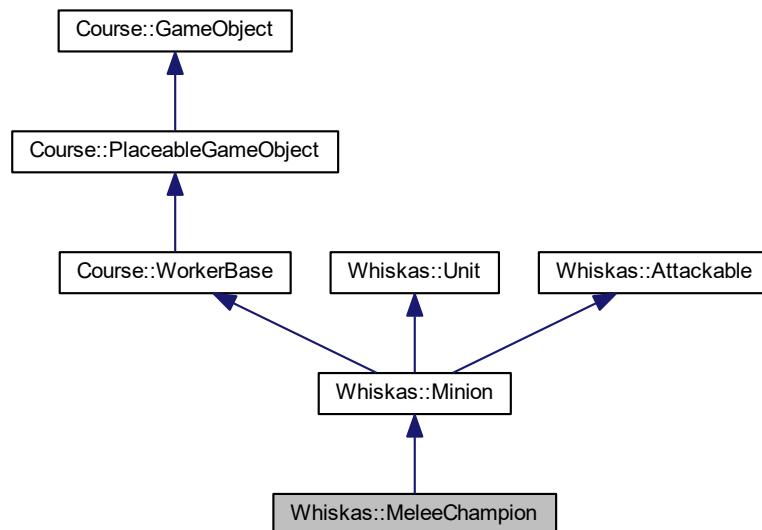
- E:/Gitin repo/ohjelmointi 3/whiskas/Game/buildings/meleealtar.h
- E:/Gitin repo/ohjelmointi 3/whiskas/Game/buildings/meleealtar.cpp

6.37 Whiskas::MeleeChampion Class Reference

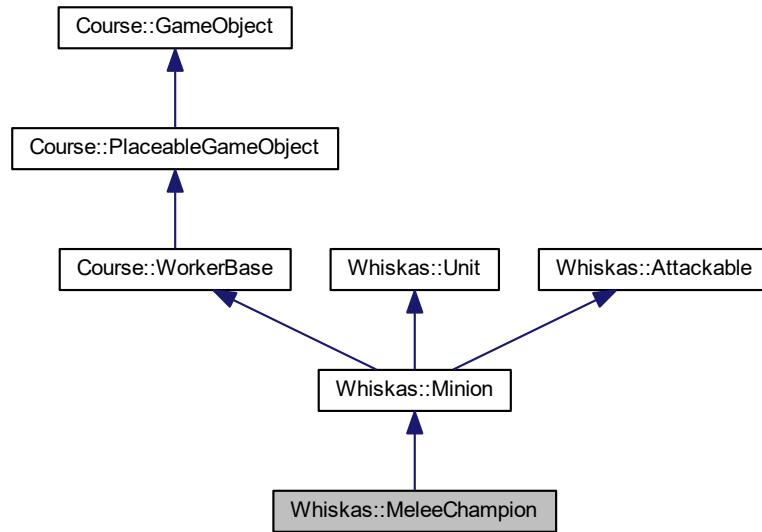
The [MeleeChampion](#) class is a minion subtype. Has a melee attack of higher dmg and better defenses.

```
#include <meleechampion.h>
```

Inheritance diagram for Whiskas::MeleeChampion:



Collaboration diagram for Whiskas::MeleeChampion:



Public Member Functions

- **MeleeChampion** (const std::shared_ptr<[Course::iGameEventHandler](#)> &handler, const std::shared_ptr<[Course::iObjectManager](#)> &manager, const std::shared_ptr<[Course::PlayerBase](#)> &owner, int movement=1, int health=5, int attack=2, int numberOfAttacks=2)
- std::string **getType** () const override
- bool **modifyHealth** (int hModifier) override

modifyHealth Melee Champion takes 1 less damage from attacks

Additional Inherited Members

6.37.1 Detailed Description

The [MeleeChampion](#) class is a minion subtype. Has a melee attack of higher dmg and better defenses.

6.37.2 Member Function Documentation

6.37.2.1 [getType\(\)](#)

```
std::string Whiskas::MeleeChampion::getType ( ) const [override], [virtual]
```

Reimplemented from [Course::WorkerBase](#).

6.37.2.2 modifyHealth()

```
bool Whiskas::MeleeChampion::modifyHealth (
    int hModifier ) [override], [virtual]
```

modifyHealth Melee Champion takes 1 less damage from attacks

Parameters

<i>hModifier</i>	
------------------	--

Returns

true if dead, false otherwise

Reimplemented from [Whiskas::Attackable](#).

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

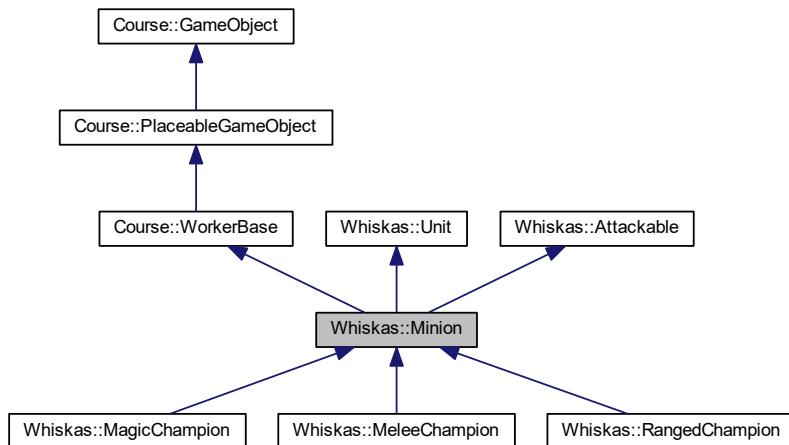
- E:/Gitin repo/ohjelmointi 3/whiskas/Game/minion/meleechampion.h
- E:/Gitin repo/ohjelmointi 3/whiskas/Game/minion/meleechampion.cpp

6.38 Whiskas::Minion Class Reference

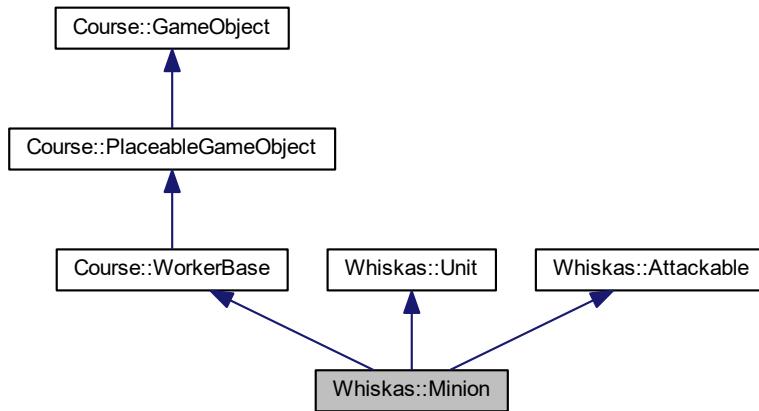
The [Minion](#) class is a subtype of worker. It can be upgraded to champions using altars. Is attackable and a unit (so it can attack and move).

```
#include <minion.h>
```

Inheritance diagram for Whiskas::Minion:



Collaboration diagram for Whiskas::Minion:



Public Member Functions

- **Minion** (const std::shared_ptr< Course::iGameEventHandler > &handler, const std::shared_ptr< Course::iObjectManager > &manager, const std::shared_ptr< Course::PlayerBase > &owner, int movement=1, int health=3, int attack=1, int numberOfattacks=1, const AdvancedResourceMap &cost={})
- void **doSpecialAction** () override
doSpecialAction does nothing, nada. Need to declare it for inheritance
- std::string **getType** () const override
- bool **modifyHealth** (int hModifier) override
modifyHealth if a minion takes more than 1 dmg from an attack, its instantly slain. Weaklings

Protected Attributes

- std::shared_ptr< gameManager > **manager_**
- AdvancedResourceMap **testCost_**

Additional Inherited Members

6.38.1 Detailed Description

The **Minion** class is a subtype of worker. It can be upgraded to champions using altars. Is attackable and a unit (so it can attack and move).

6.38.2 Member Function Documentation

6.38.2.1 getType()

```
std::string Whiskas::Minion::getType ( ) const [override], [virtual]
```

Reimplemented from [Course::WorkerBase](#).

Reimplemented in [Whiskas::RangedChampion](#).

6.38.2.2 modifyHealth()

```
bool Whiskas::Minion::modifyHealth ( int hModifier ) [override], [virtual]
```

modifyHealth if a minion takes more than 1 dmg from an attack, its instantly slain. Weaklings

Parameters

<i>hModifier</i>	
------------------	--

Returns

true if dead, false otherwise

Reimplemented from [Whiskas::Attackable](#).

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

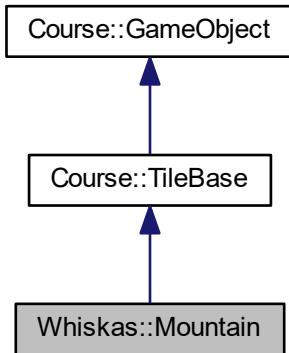
- E:/Gitin repo/ohjelmointi 3/whiskas/Game/minion/minion.h
- E:/Gitin repo/ohjelmointi 3/whiskas/Game/minion/minion.cpp

6.39 Whiskas::Mountain Class Reference

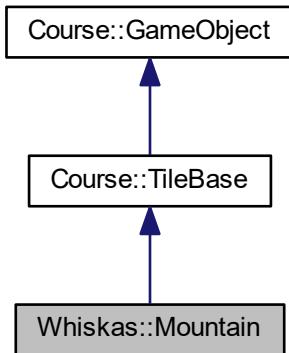
The [Mountain](#) class represents the mountain in the gameworld. The mountain supports melee altars and quarries.

```
#include <mountain.h>
```

Inheritance diagram for Whiskas::Mountain:



Collaboration diagram for Whiskas::Mountain:



Public Member Functions

- **Mountain** (const `Course::Coordinate &location`, const `std::shared_ptr< Course::iGameEventHandler > &eventhandler`, const `std::shared_ptr< Course::iObjectManager > &objectmanager`, const `unsigned int &max_build=1`, const `unsigned int &max_work=1`, const `Course::ResourceMap &production={}`)
- `virtual std::string getType () const override`
- `void addBuilding (const std::shared_ptr< Course::BuildingBase > &building) override`

Adds a new Building-object to the tile.

Additional Inherited Members

6.39.1 Detailed Description

The `Mountain` class represents the mountain in the gameworld. The mountain supports melee altars and quarries.

6.39.2 Member Function Documentation

6.39.2.1 addBuilding()

```
void Whiskas::Mountain::addBuilding (
    const std::shared_ptr< Course::BuildingBase > & building ) [override], [virtual]
```

Adds a new Building-object to the tile.

Phases:

1. Tile checks if it has space for the building.
2. Building checks whether it can be placed on this tile.
3. Building is added to this Tile.
4. Tile update's Building's location.

Parameters

<i>building</i>	A pointer to the Building that is being added.
-----------------	--

Postcondition

Exception guarantee: Basic

Exceptions

<i>InvalidPointer</i>	- If the building's pointer doesn't point to anything or ObjectManager doesn't return valid shared_ptr to this tile.
<i>IllegalMove</i>	- Any IllegalException can be thrown by a Tile or Building if it breaks a placement rule.
<i>NotEnoughSpace</i>	(IllegalMove) - If the tile doesn't have enough space for the Building.

Reimplemented from [Course::TileBase](#).

6.39.2.2 getType()

```
std::string Whiskas::Mountain::getType ( ) const [override], [virtual]
```

Reimplemented from [Course::TileBase](#).

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

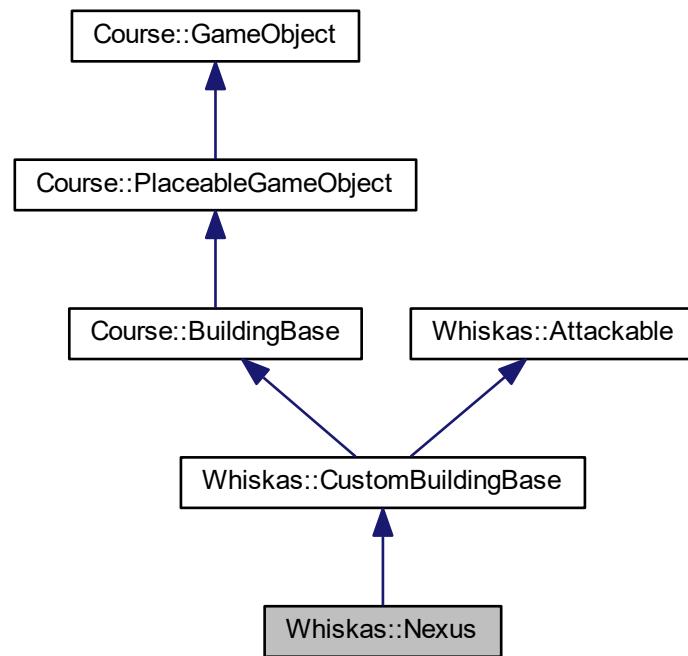
- E:/Gitin repo/ohjelmointi 3/whiskas/Game/tiles/mountain.h
- E:/Gitin repo/ohjelmointi 3/whiskas/Game/tiles/mountain.cpp

6.40 Whiskas::Nexus Class Reference

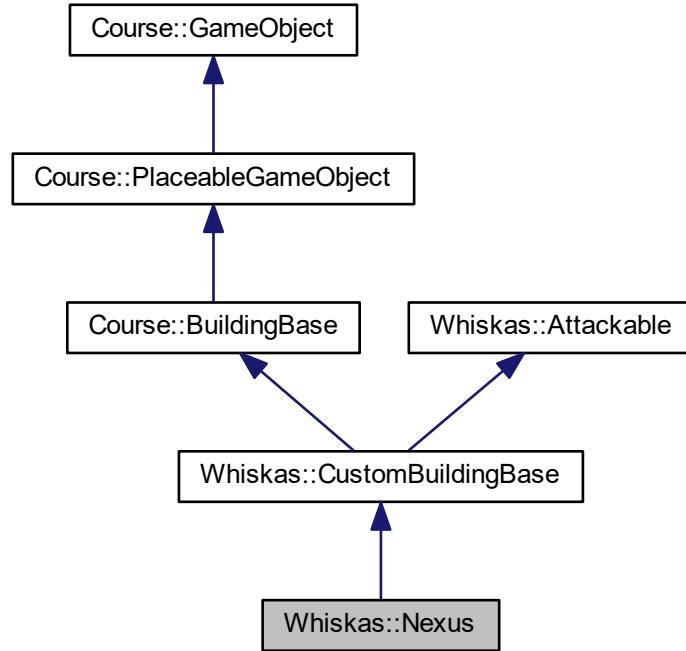
The [Nexus](#) class represents a Nexus-building in the game.

```
#include <nexus.h>
```

Inheritance diagram for Whiskas::Nexus:



Collaboration diagram for Whiskas::Nexus:



Public Member Functions

- `Nexus` (const std::shared_ptr<[gameEventHandler](#)> &eventhandler, const std::shared_ptr<[gameManager](#)> &objectmanager, const std::shared_ptr<[Course::PlayerBase](#)> &owner, const int &tilespaces=1, const AdvancedResourceMap buildcost={}, const AdvancedResourceMap production=NEXUS_PRODUCE, int health=5, int attack=0)
- `~Nexus` () override=default
Default destructor.
- std::string `getType` () const override

Additional Inherited Members

6.40.1 Detailed Description

The [Nexus](#) class represents a Nexus-building in the game.

The [Nexus](#) produces 1 of each resource per turn. The [Nexus](#) also spawns a minion at the start of the game and every 5 turns.

6.40.2 Member Function Documentation

6.40.2.1 `getType()`

```
std::string Whiskas::Nexus::getType ( ) const [override], [virtual]
```

Reimplemented from [Course::BuildingBase](#).

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

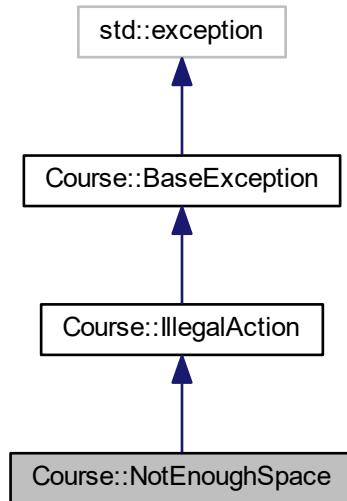
- E:/Gitin repo/ohjelmointi 3/whiskas/Game/buildings/nexus.h
- E:/Gitin repo/ohjelmointi 3/whiskas/Game/buildings/nexus.cpp

6.41 Course::NotEnoughSpace Class Reference

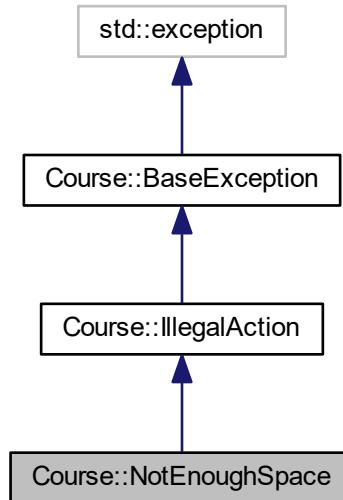
The [NotEnoughSpace](#) class is an Exception-class for errors where GameObjects are being placed onto Tiles with no space available.

```
#include <notenoughspace.h>
```

Inheritance diagram for Course::NotEnoughSpace:



Collaboration diagram for Course::NotEnoughSpace:



Public Member Functions

- [NotEnoughSpace \(const std::string &msg=""\)](#)
Exception Constructor.
- virtual [~NotEnoughSpace \(\)=default](#)
~Exception Default destructor

6.41.1 Detailed Description

The [NotEnoughSpace](#) class is an Exception-class for errors where GameObjects are being placed onto Tiles with no space available.

6.41.2 Constructor & Destructor Documentation

6.41.2.1 NotEnoughSpace()

```
Course::NotEnoughSpace::NotEnoughSpace (const std::string & msg = "") [inline], [explicit]
```

Exception Constructor.

Parameters

<i>msg</i>	std::string describing the reason for exception.
------------	--

6.41.2.2 ~NotEnoughSpace()

```
virtual Course::NotEnoughSpace::~NotEnoughSpace ( ) [virtual], [default]
```

~Exception Default destructor

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

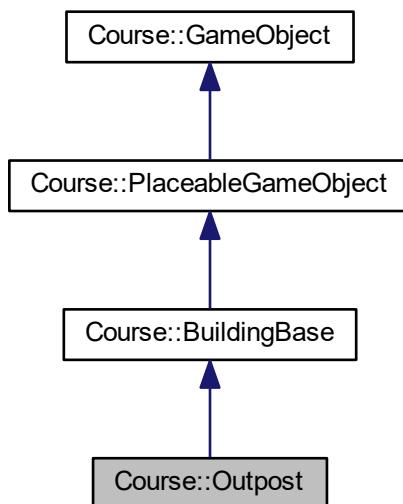
- E:/Gitin repo/ohjelmointi 3/whiskas/Course/CourseLib/exceptions/notenoughspace.h

6.42 Course::Outpost Class Reference

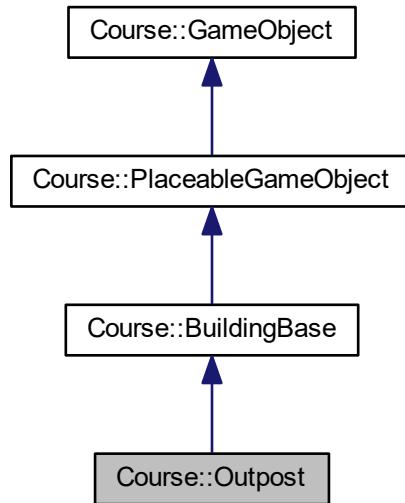
The [Outpost](#) class represents a player's Outpost-building.

```
#include <outpost.h>
```

Inheritance diagram for Course::Outpost:



Collaboration diagram for Course::Outpost:



Public Member Functions

- [Outpost \(\)=delete](#)
Disabled parameterless constructor.
- [Outpost \(const std::shared_ptr< iGameEventHandler > &eventhandler, const std::shared_ptr< iObjectManager > &objectmanager, const std::shared_ptr< PlayerBase > &owner, const int &tilespaces=1, const ResourceMap &buildcost=ConstResourceMaps::OUTPOST_BUILD_COST, const ResourceMap &production=ConstResourceMaps::OUTPOST_PRODUCTION\)](#)
Constructor for the class.
- virtual [~Outpost \(\)=default](#)
Default destructor.
- virtual std::string [getType \(\) const override](#)
GetType Returns a string describing objects type. This should be overriden in each inherited class. Makes checking object's type easier for students.
- virtual void [onBuildAction \(\) override](#)
Sets neighbouring Tiles' ownership to this building's ownership in 1 tile-radius, if the Tiles don't already have an owner.
- virtual ResourceMap [getProduction \(\) override](#)
getProduction

Additional Inherited Members

6.42.1 Detailed Description

The [Outpost](#) class represents a player's Outpost-building.

It can be constructed on any tile that has not been claimed by any other player.

Effects: Claims surrounding unclaimed tiles.

Radius: 1 tiles

Production: -10 money (upkeep)

6.42.2 Constructor & Destructor Documentation

6.42.2.1 Outpost()

```
Course::Outpost::Outpost (
    const std::shared_ptr< iGameEventHandler > & eventhandler,
    const std::shared_ptr< iObjectManager > & objectmanager,
    const std::shared_ptr< PlayerBase > & owner,
    const int & tilespaces = 1,
    const ResourceMap & buildcost = ConstResourceMaps::OUTPOST_BUILD_COST,
    const ResourceMap & production = ConstResourceMaps::OUTPOST_PRODUCTION ) [explicit]
```

Constructor for the class.

Parameters

<i>eventhandler</i>	points to the student's GameEventHandler.
<i>owner</i>	points to the owning player.
<i>tile</i>	points to the tile upon which the building is constructed.

Postcondition

Exception Guarantee: No guarantee.

Exceptions

<i>OwnerConflict</i>	- if the building conflicts with tile's ownership.
----------------------	--

6.42.3 Member Function Documentation

6.42.3.1 getProduction()

```
ResourceMap Course::Outpost::getProduction ( ) [override], [virtual]
```

getProduction

Returns

Postcondition

Exception guarantee: Basic

Reimplemented from [Course::BuildingBase](#).

6.42.3.2 getType()

```
std::string Course::Outpost::getType ( ) const [override], [virtual]
```

getType Returns a string describing objects type. This should be overriden in each inherited class. Makes checking object's type easier for students.

Returns

std::string that represents Object's type.

Postcondition

Exception guarantee: No-throw

Note

You can use this in e.g. debugging and similar printing.

Reimplemented from [Course::BuildingBase](#).

6.42.3.3 onBuildAction()

```
void Course::Outpost::onBuildAction ( ) [override], [virtual]
```

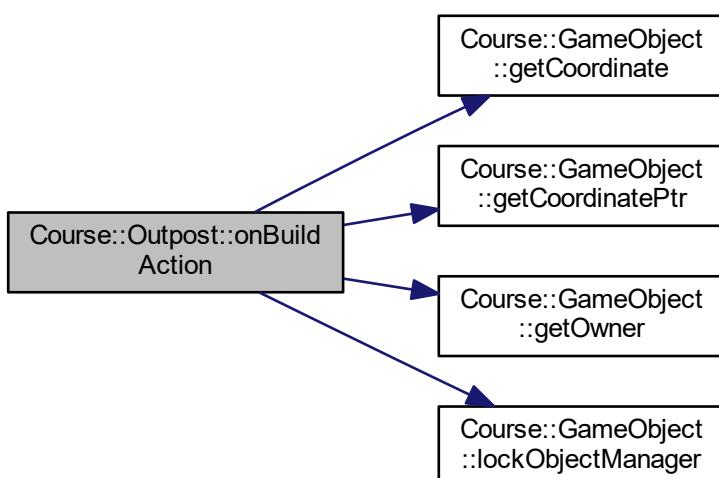
Sets neighbouring Tiles' ownership to this building's ownership in 1 tile-radius, if the Tiles don't already have an owner.

Postcondition

Exception guarantee: Basic

Reimplemented from [Course::BuildingBase](#).

Here is the call graph for this function:



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

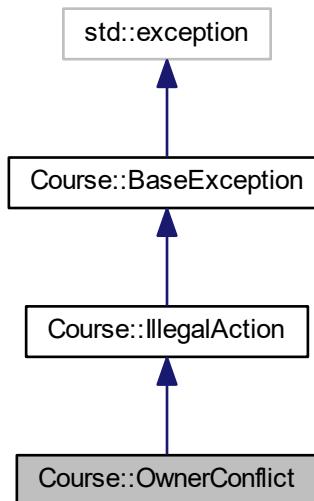
- E:/Gitin repo/ohjelmointi 3/whiskas/Course/CourseLib/buildings/outpost.h
- E:/Gitin repo/ohjelmointi 3/whiskas/Course/CourseLib/buildings/outpost.cpp

6.43 Course::OwnerConflict Class Reference

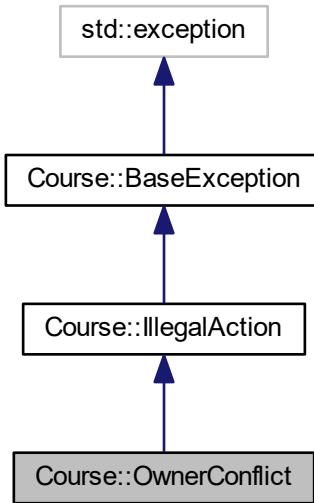
The [OwnerConflict](#) class is an Exception-class for errors where an operation is conflicting with a [GameObject](#)'s ownership.

```
#include <ownerconflict.h>
```

Inheritance diagram for Course::OwnerConflict:



Collaboration diagram for Course::OwnerConflict:



Public Member Functions

- [OwnerConflict \(const std::string &msg=""\)](#)
Exception Constructor.
- virtual [~OwnerConflict \(\)=default](#)
~Exception Default destructor

6.43.1 Detailed Description

The [OwnerConflict](#) class is an Exception-class for errors where an operation is conflicting with a [GameObject](#)'s ownership.

6.43.2 Constructor & Destructor Documentation

6.43.2.1 OwnerConflict()

```
Course::OwnerConflict::OwnerConflict (
    const std::string & msg = "" ) [inline], [explicit]
```

Exception Constructor.

Parameters

<i>msg</i>	std::string describing the reason for exception.
------------	--

6.43.2.2 ~OwnerConflict()

```
virtual Course::OwnerConflict::~OwnerConflict () [virtual], [default]
```

~Exception Default destructor

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

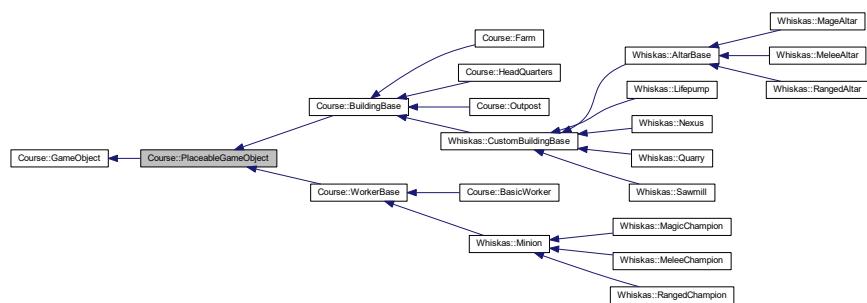
- E:/Gitin repo/ohjelmointi 3/whiskas/Course/CourseLib/exceptions/ownerconflict.h

6.44 Course::PlaceableGameObject Class Reference

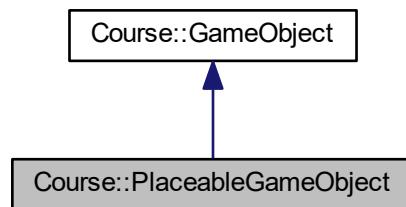
The [PlaceableGameObject](#) class represents GameObjects that can be placed on Tile Objects.

```
#include <placeablegameobject.h>
```

Inheritance diagram for Course::PlaceableGameObject:



Collaboration diagram for Course::PlaceableGameObject:



Public Member Functions

- `PlaceableGameObject ()=delete`
Disabled parameterless constructor.
- `PlaceableGameObject (const std::shared_ptr< iGameEventHandler > &eventhandler, const std::shared_ptr< iObjectManager > &objectmanager, const std::shared_ptr< PlayerBase > &owner, const int &tilespace=1)`
Constructor for the class.
- `~PlaceableGameObject () override=default`
Default destructor.
- `std::string getType () const override`
getType Returns a string describing objects type. This should be overriden in each inherited class. Makes checking object's type easier for students.
- `virtual int spacesInTileCapacity () const`
How many spaces does the `GameObject` take from a Tile's capacity.
- `virtual bool canBePlacedOnTile (const std::shared_ptr< TileBase > &target) const`
Check if the object's own placement rules allow placement on the specified tile.
- `virtual void setLocationTile (const std::shared_ptr< TileBase > &tile) final`
Set the `PlaceableGameObject`'s location.
- `virtual std::shared_ptr< TileBase > currentLocationTile () const final`
Returns a shared_ptr to current location-tile.

Public Attributes

- `const int TILESPACES`

Additional Inherited Members

6.44.1 Detailed Description

The `PlaceableGameObject` class represents GameObjects that can be placed on Tile Objects.

Handles placement on a Tile and throws exception if the placement would violate PlaceableGameObjects own placement rules.

Note

One new overridable method: `canBePlacedOnTile` - Can be used to re-specify placement-rules.

6.44.2 Constructor & Destructor Documentation

6.44.2.1 PlaceableGameObject()

```
Course::PlaceableGameObject::PlaceableGameObject (
    const std::shared_ptr< iGameEventHandler > & eventhandler,
    const std::shared_ptr< iObjectManager > & objectmanager,
    const std::shared_ptr< PlayerBase > & owner,
    const int & tilespace = 1 ) [explicit]
```

Constructor for the class.

Parameters

<i>eventhandler</i>	points to the student's GameEventHandler.
<i>objectmanager</i>	points to the student's ObjectManager
<i>owner</i>	points to the owning player.
<i>tilespace</i>	indicates the amount of tilespace the object would take when placed on a tile.

6.44.3 Member Function Documentation**6.44.3.1 canBePlacedOnTile()**

```
bool Course::PlaceableGameObject::canBePlacedOnTile (
    const std::shared_ptr< TileBase > & target ) const [virtual]
```

Check if the object's own placement rules allow placement on the specified tile.

Parameters

<i>target</i>	is a pointer to the tile that is being targeted.
---------------	--

Returns

True - Object has same owner as the tile.
 False - If condition doesn't match.

Postcondition

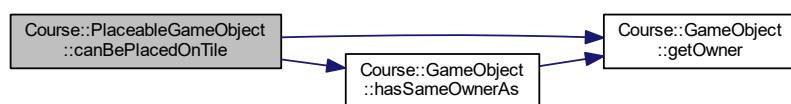
Exception guarantee: Basic

Note

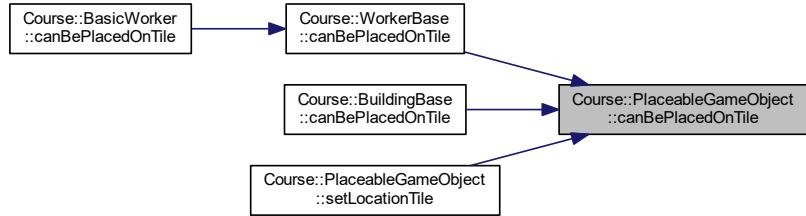
Override to change default behaviour

Reimplemented in [Course::BuildingBase](#), [Course::WorkerBase](#), and [Course::BasicWorker](#).

Here is the call graph for this function:



Here is the caller graph for this function:



6.44.3.2 currentLocationTile()

```
std::shared_ptr< TileBase > Course::PlaceableGameObject::currentLocationTile ( ) const [final],  
[virtual]
```

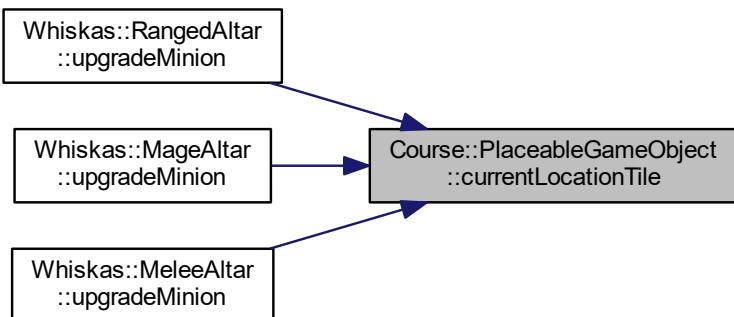
Returns a shared_ptr to current location-tile.

Returns

Postcondition

Exception guarantee: No-throw

Here is the caller graph for this function:



6.44.3.3 getType()

```
std::string Course::PlaceableGameObject::getType ( ) const [override], [virtual]
```

`getType` Returns a string describing objects type. This should be overriden in each inherited class. Makes checking object's type easier for students.

Returns

`std::string` that represents Object's type.

Postcondition

Exception guarantee: No-throw

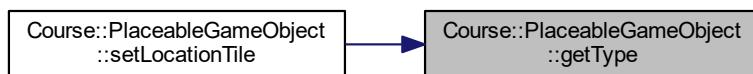
Note

You can use this in e.g. debugging and similar printing.

Reimplemented from [Course::GameObject](#).

Reimplemented in [Course::WorkerBase](#), [Course::BasicWorker](#), [Whiskas::Minion](#), [Whiskas::Nexus](#), [Whiskas::Quarry](#), [Whiskas::Sawmill](#), [Whiskas::Lifepump](#), [Whiskas::MeleeChampion](#), [Whiskas::MagicChampion](#), [Whiskas::RangedChampion](#), [Whiskas::AltarBase](#), [Whiskas::MeleeAltar](#), [Whiskas::MageAltar](#), and [Whiskas::RangedAltar](#).

Here is the caller graph for this function:



6.44.3.4 setLocationTile()

```
void Course::PlaceableGameObject::setLocationTile (
    const std::shared_ptr< TileBase > & tile ) [final], [virtual]
```

Set the [PlaceableGameObject](#)'s location.

Parameters

<code>tile</code>	points to the Tile where the object is placed.
-------------------	--

Postcondition

Exception guarantee: Strong

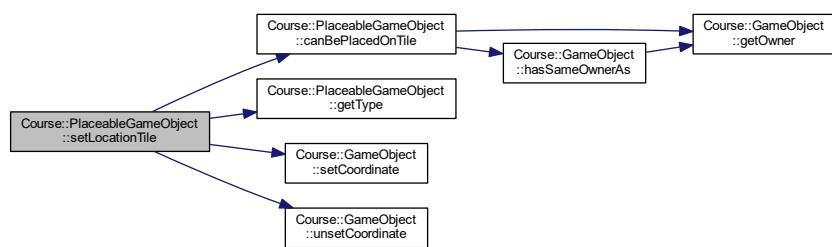
Note

`nullptr` can be used to clear the location.

Exceptions

<i>IllegalAction</i>	- If <code>canBePlacedOnTile</code> returns False.
----------------------	--

Here is the call graph for this function:

**6.44.3.5 spacesInTileCapacity()**

```
int Course::PlaceableGameObject::spacesInTileCapacity ( ) const [virtual]
```

How many spaces does the [GameObject](#) take from a Tile's capacity.

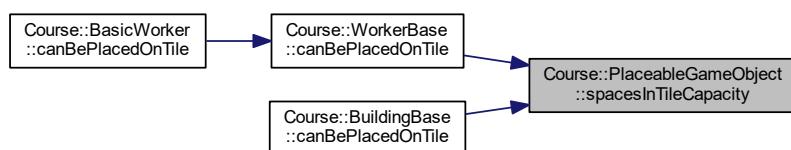
Returns

Amount of spaces that is being taken.

Postcondition

Exception guarantee: No-throw

Here is the caller graph for this function:



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

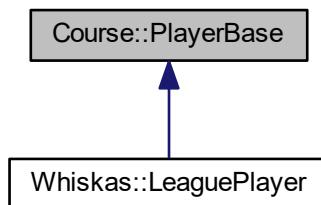
- E:/Gitin repo/ohjelmointi 3/whiskas/Course/CourseLib/core/placeablegameobject.h
- E:/Gitin repo/ohjelmointi 3/whiskas/Course/CourseLib/core/placeablegameobject.cpp

6.45 Course::PlayerBase Class Reference

The [PlayerBase](#) class is a base class for classes used to describe a player in game.

```
#include <playerbase.h>
```

Inheritance diagram for Course::PlayerBase:



Public Member Functions

- [PlayerBase](#) (const std::string &name, std::vector< std::shared_ptr< [GameObject](#) > > objects={})
Constructor for the class.
- virtual ~[PlayerBase](#) ()=default
Default destructor.
- virtual void [setName](#) (const std::string &new_name) final
Sets a new m_name value for the class.
- virtual void [addObject](#) (std::shared_ptr< [GameObject](#) > object) final
Stores a weak GameObject-pointer.
- virtual void [addObjects](#) (const std::vector< std::shared_ptr< [GameObject](#) > > &objects) final
Stores a vector of weak GameObject-pointers.
- virtual void [removeObject](#) (const std::shared_ptr< [GameObject](#) > &object) final
Removes a weak GameObject-pointer and expired weak pointers.
- virtual void [removeObjects](#) (const std::vector< std::shared_ptr< [GameObject](#) > > &objects) final
Removes a list of weak GameObject-pointers and expired weak pointers.
- virtual void [removeObject](#) (const ObjectId &id) final
Removes a weak GameObject-pointers based on a ObjectId and removes expired weak pointers.
- virtual void [removeObjects](#) (const std::vector< ObjectId > &objects) final
Removes a list of weak GameObject-pointers based on a ObjectId and removes expired weak pointers.
- virtual std::vector< std::shared_ptr< [GameObject](#) > > [getObjects](#) () const final
Returns the vector of weak GameObject-pointers that are currently stored in the Player-object.
- virtual std::string [getName](#) () const final
Returns the Player-object's name.

6.45.1 Detailed Description

The [PlayerBase](#) class is a base class for classes used to describe a player in game.

The class can be used to store and access GameObjects. Expired weak pointers are automatically removed when requesting or removing objects.

Note

Objects are stored as weak pointers.

6.45.2 Constructor & Destructor Documentation

6.45.2.1 PlayerBase()

```
Course::PlayerBase::PlayerBase (
    const std::string & name,
    std::vector< std::shared_ptr< GameObject > > objects = {} )
```

Constructor for the class.

Parameters

<i>name</i>	A std::string for player's name
<i>objects</i>	(optional) A std::vector of shared-pointers to GameObjects.

6.45.3 Member Function Documentation

6.45.3.1 addObject()

```
void Course::PlayerBase::addObject (
    std::shared_ptr< GameObject > object ) [final], [virtual]
```

Stores a weak GameObject-pointer.

Parameters

<i>object</i>	Is a weak pointer to the stored GameObject
---------------	--

Postcondition

Exception guarantee: Strong

Exceptions

<i>See</i>	<code>std::vector::push_back()</code>
------------	---------------------------------------

6.45.3.2 addObjects()

```
void Course::PlayerBase::addObjects (
    const std::vector< std::shared_ptr< GameObject > > & objects ) [final], [virtual]
```

Stores a vector of weak GameObject-pointers.

Parameters

<i>objects</i>	Is an <code>std::vector</code> of weak <code>GameObject</code> -pointers.
----------------	---

Postcondition

Exception guarantee: Strong

Exceptions

<i>See</i>	<code>std::vector::insert()</code>
------------	------------------------------------

6.45.3.3 getName()

```
std::string Course::PlayerBase::getName ( ) const [final], [virtual]
```

Returns the Player-object's name.

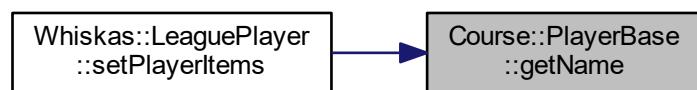
Returns

Copy of current string in `m_name`

Postcondition

Exception guarantee: No-throw

Here is the caller graph for this function:



6.45.3.4 getObjects()

```
std::vector< std::shared_ptr< GameObject > > Course::PlayerBase::getObjects ( ) const [final],  
[virtual]
```

Returns the vector of weak GameObject-pointers that are currently stored in the Player-object.

Returns

Copy of m_objects -vector

Postcondition

Exception guarantee: Strong

6.45.3.5 removeObject() [1/2]

```
void Course::PlayerBase::removeObject (   
    const ObjectId & id ) [final], [virtual]
```

Removes a weak GameObject-pointers based on a ObjectId and removes expired weak pointers.

Parameters

<i>id</i>	An ObjectId (unsigned int) for GameObject which is removed
-----------	--

Postcondition

Exception guarantee: Basic

Exceptions

KeyError	- No GameObjects with given ID were found.
See	std::remove_if

6.45.3.6 removeObject() [2/2]

```
void Course::PlayerBase::removeObject (   
    const std::shared_ptr< GameObject > & object ) [final], [virtual]
```

Removes a weak GameObject-pointer and expired weak pointers.

Parameters

<i>object</i>	a weak pointer to GameObject
---------------	--

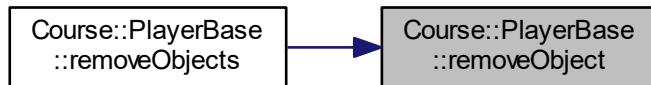
Postcondition

Exception guarantee: Basic

Exceptions

<i>ExpiredPointer</i>	- object is expired
<i>KeyError</i>	- No objects match the searched object

Here is the caller graph for this function:

**6.45.3.7 removeObjects() [1/2]**

```
void Course::PlayerBase::removeObjects (
    const std::vector< ObjectId > & objects ) [final], [virtual]
```

Removes a list of weak GameObject-pointers based on a ObjectId and removes expired weak pointers.

Parameters

<i>objects</i>	A vector of ObjectId (unsigned int) for GameObjects that are removed
----------------	--

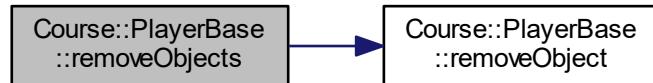
Postcondition

Exception guarantee: No-throw

Note

Even if some of the provided ID's are not found, no exceptions are thrown.

Here is the call graph for this function:



6.45.3.8 removeObjects() [2/2]

```
void Course::PlayerBase::removeObjects (
    const std::vector< std::shared_ptr< GameObject > > & objects ) [final], [virtual]
```

Removes a list of weak GameObject-pointers and expired weak pointers.

Parameters

<i>objects</i>	A vector of weak GameObject-pointers
----------------	--------------------------------------

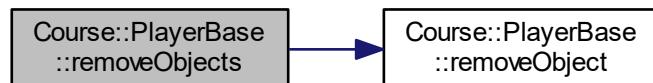
Postcondition

Exception guarantee: No-throw

Note

If some of the provided weak pointers are expired, no exceptions are thrown.

Here is the call graph for this function:



6.45.3.9 setName()

```
void Course::PlayerBase::setName (
    const std::string & new_name ) [final], [virtual]
```

Sets a new *m_name* value for the class.

Parameters

<i>new_name</i>	The new name.
-----------------	---------------

Postcondition

Exception guarantee: No-throw

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

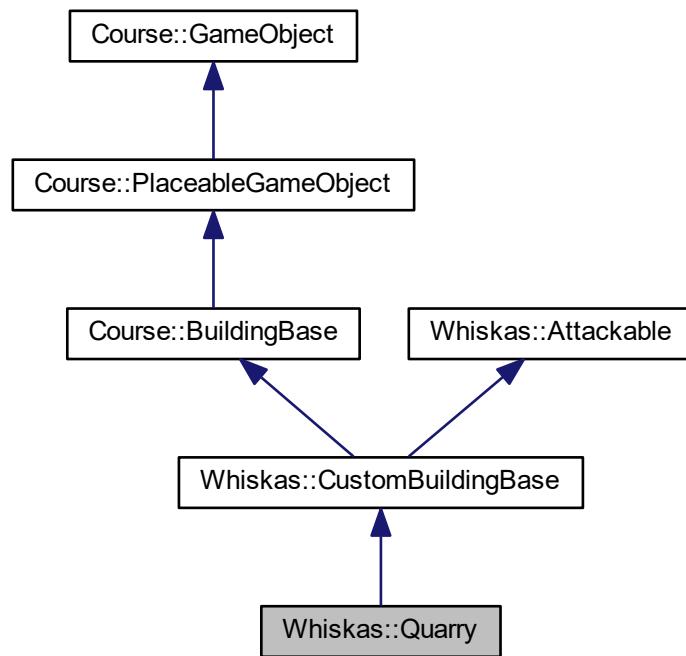
- E:/Gitin repo/ohjelmointi 3/whiskas/Course/CourseLib/core/playerbase.h
- E:/Gitin repo/ohjelmointi 3/whiskas/Course/CourseLib/core/playerbase.cpp

6.46 Whiskas::Quarry Class Reference

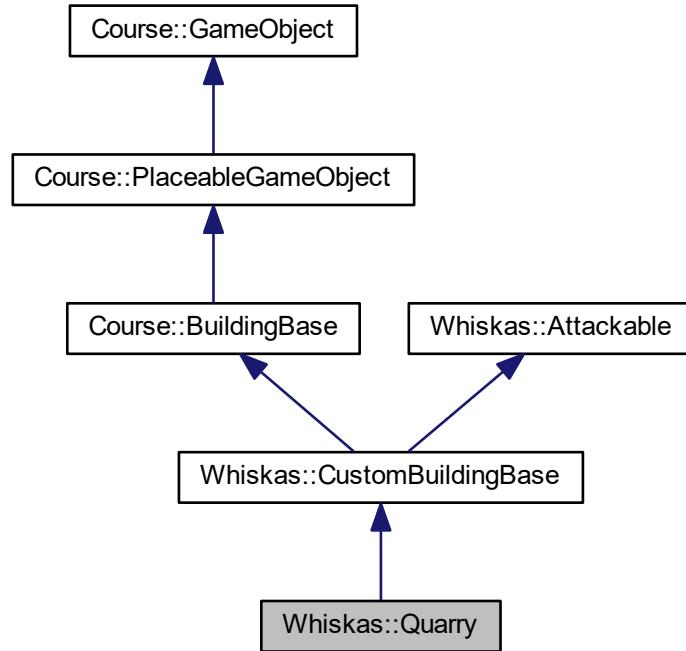
The [Quarry](#) class represents a quarry-building in the game.

```
#include <quarry.h>
```

Inheritance diagram for Whiskas::Quarry:



Collaboration diagram for Whiskas::Quarry:



Public Member Functions

- `Quarry (const std::shared_ptr< gameEventHandler > &eventhandler, const std::shared_ptr< gameManager > &objectmanager, const std::shared_ptr< Course::PlayerBase > &owner, const int &tilespaces=1, const AdvancedResourceMap &buildcost=QUARRY_COST, const AdvancedResourceMap &production=QUARRY_PRODUCE, int health=5, int attack=0)`
- `~Quarry () override=default`
Default destructor.
- `std::string getType () const override`

Additional Inherited Members

6.46.1 Detailed Description

The `Quarry` class represents a quarry-building in the game.

The quarry produces 1 stone per turn Can only be built on a mountain

6.46.2 Member Function Documentation

6.46.2.1 `getType()`

```
std::string Whiskas::Quarry::getType ( ) const [override], [virtual]
```

Reimplemented from [Course::BuildingBase](#).

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

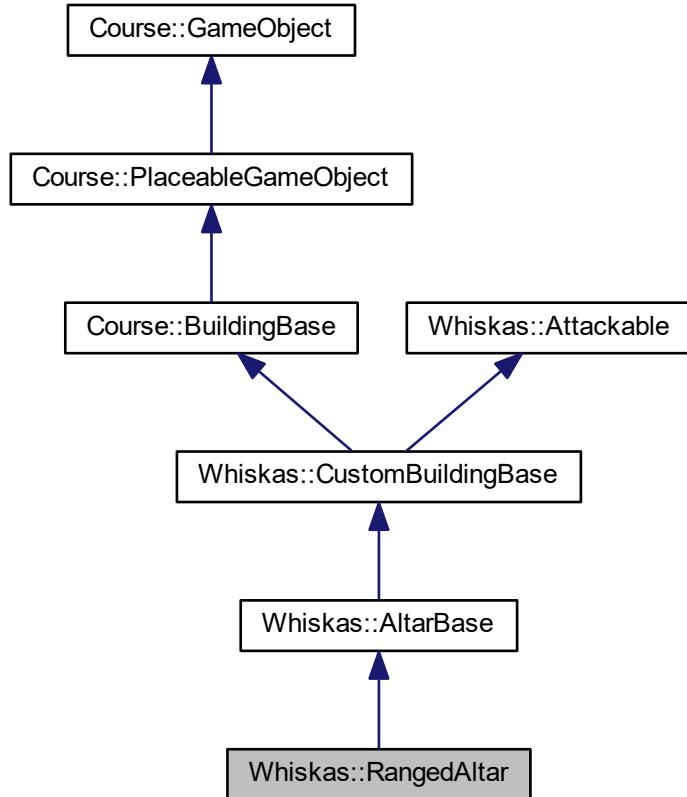
- E:/Gitin repo/ohjelmointi 3/whiskas/Game/buildings/quarry.h
- E:/Gitin repo/ohjelmointi 3/whiskas/Game/buildings/quarry.cpp

6.47 Whiskas::RangedAltar Class Reference

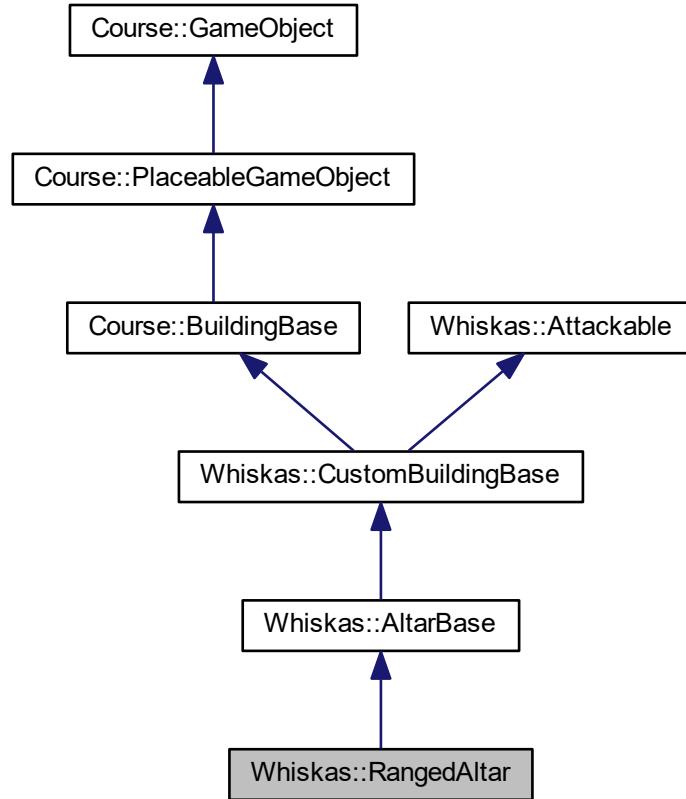
The [RangedAltar](#) class is an altar that upgrades minions to ranged units.

```
#include <rangedaltar.h>
```

Inheritance diagram for Whiskas::RangedAltar:



Collaboration diagram for Whiskas::RangedAltar:



Public Member Functions

- `RangedAltar (const std::shared_ptr< gameEventHandler > &eventhandler, const std::shared_ptr< gameManager > &objectmanager, const std::shared_ptr< Course::PlayerBase > &owner)`
- void `upgradeMinion ()` override
upgradeMinion upgrades a minion on the tile when called.
- `std::string getType () const` override

Additional Inherited Members

6.47.1 Detailed Description

The `RangedAltar` class is an altar that upgrades minions to ranged units.

6.47.2 Member Function Documentation

6.47.2.1 `getType()`

```
std::string Whiskas::RangedAltar::getType () const [override], [virtual]
```

Reimplemented from [Whiskas::AltarBase](#).

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

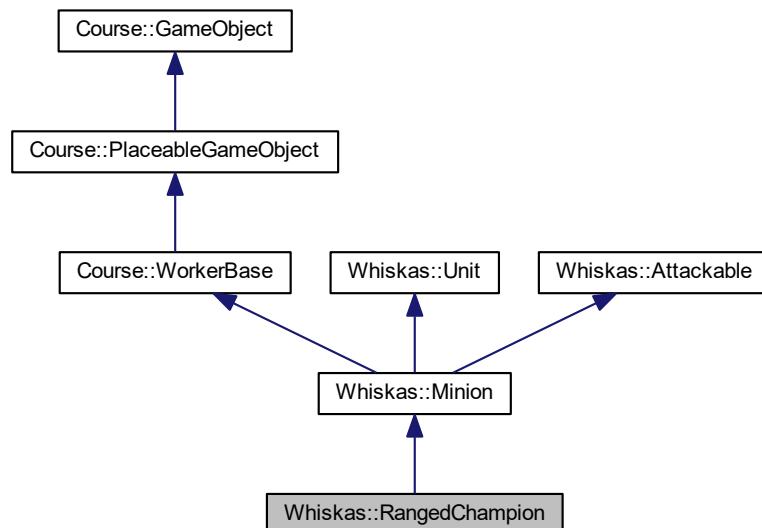
- E:/Gitin repo/ohjelmointi 3/whiskas/Game/buildings/rangedaltar.h
- E:/Gitin repo/ohjelmointi 3/whiskas/Game/buildings/rangedaltar.cpp

6.48 Whiskas::RangedChampion Class Reference

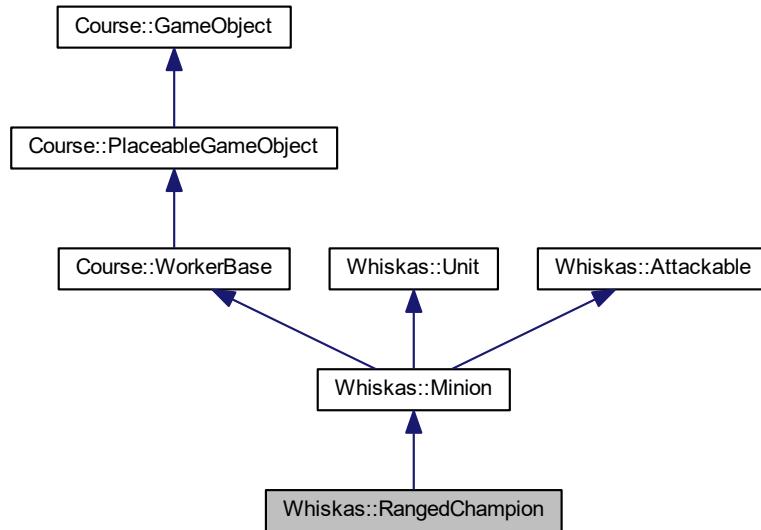
The [RangedChampion](#) class Has a ranged attack and higher movement.

```
#include <rangedchampion.h>
```

Inheritance diagram for Whiskas::RangedChampion:



Collaboration diagram for Whiskas::RangedChampion:



Public Member Functions

- `RangedChampion (const std::shared_ptr< Course::iGameEventHandler > &handler, const std::shared_ptr< Course::IObjectManager > &manager, const std::shared_ptr< Course::PlayerBase > &owner, int movement=3, int health=3, int attack=1, int number_of_attacks=3)`
- `std::string getType () const override`

Additional Inherited Members

6.48.1 Detailed Description

The `RangedChampion` class has a ranged attack and higher movement.

6.48.2 Member Function Documentation

6.48.2.1 `getType()`

```
std::string Whiskas::RangedChampion::getType ( ) const [override], [virtual]
```

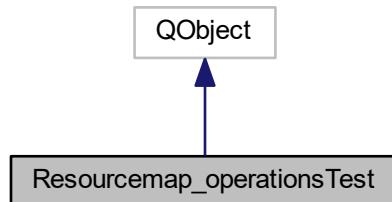
Reimplemented from [Whiskas::Minion](#).

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

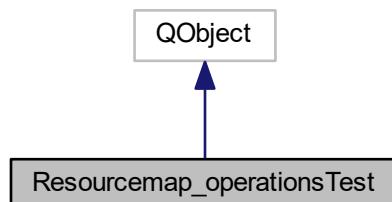
- E:/Gitin repo/ohjelmointi 3/whiskas/Game/minion/rangedchampion.h
- E:/Gitin repo/ohjelmointi 3/whiskas/Game/minion/rangedchampion.cpp

6.49 Resourcemap_operationsTest Class Reference

Inheritance diagram for Resourcemap_operationsTest:



Collaboration diagram for Resourcemap_operationsTest:



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

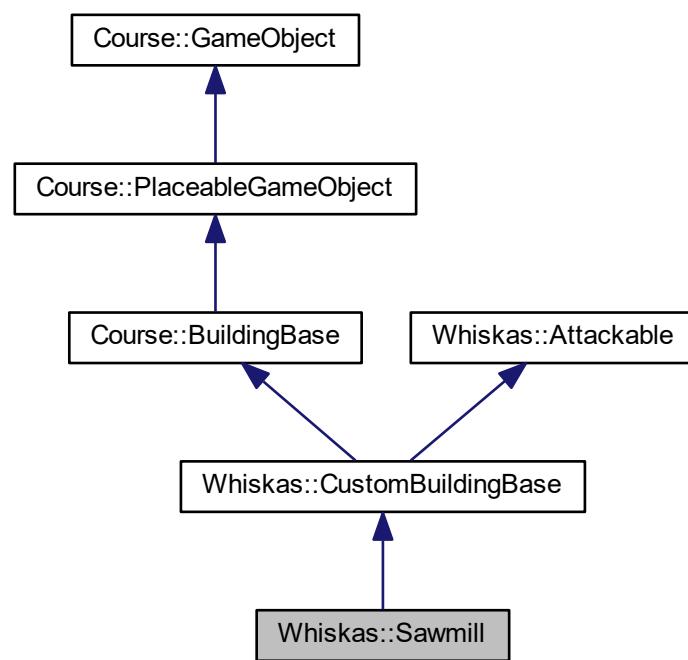
- E:/Gitin repo/ohjelmointi 3/whiskas/Course/UnitTests/resourcemap_operations/tst_resourcemap_operationstest.cpp

6.50 Whiskas::Sawmill Class Reference

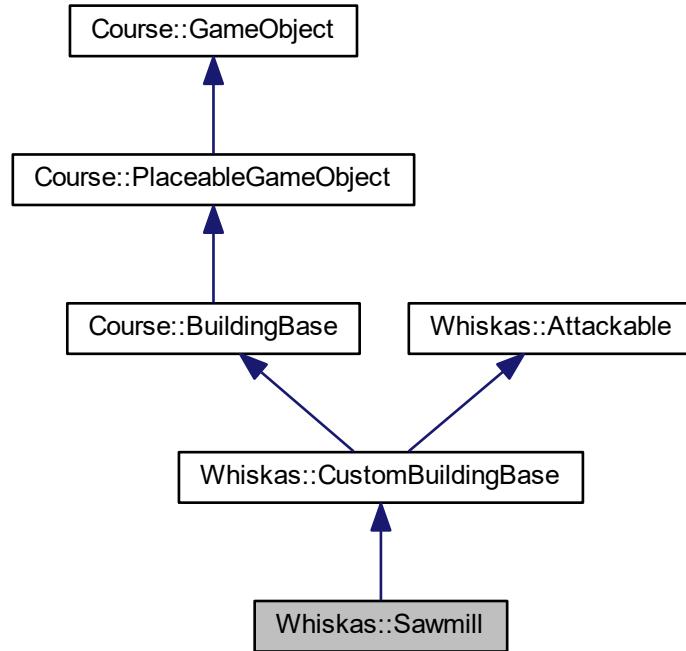
The [Sawmill](#) class represents a Sawmill-building in the game.

```
#include <sawmill.h>
```

Inheritance diagram for Whiskas::Sawmill:



Collaboration diagram for Whiskas::Sawmill:



Public Member Functions

- `Sawmill` (const std::shared_ptr< `gameEventHandler` > &eventhandler, const std::shared_ptr< `gameManager` > &objectmanager, const std::shared_ptr< `Course::PlayerBase` > &owner, const int &tilespaces=1, const AdvancedResourceMap &buildcost=SAWMILL_COST, const AdvancedResourceMap &production=SAWMILL_PRODUCE, int health=5, int attack=0)
- `~Sawmill` () override=default
Default destructor.
- std::string `getType` () const override

Additional Inherited Members

6.50.1 Detailed Description

The `Sawmill` class represents a Sawmill-building in the game.

The sawmill produces 1 wood per turn Can only be built on a jungle

6.50.2 Member Function Documentation

6.50.2.1 getType()

```
std::string Whiskas::Sawmill::getType ( ) const [override], [virtual]
```

Reimplemented from [Course::BuildingBase](#).

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

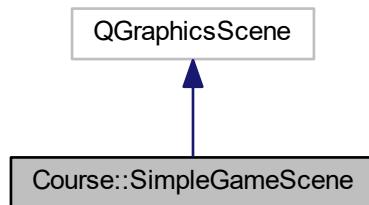
- E:/Gitin repo/ohjelmointi 3/whiskas/Game/buildings/sawmill.h
- E:/Gitin repo/ohjelmointi 3/whiskas/Game/buildings/sawmill.cpp

6.51 Course::SimpleGameScene Class Reference

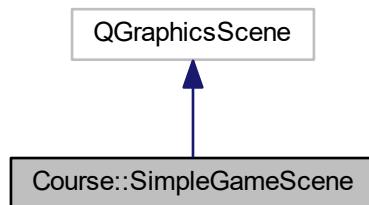
The [SimpleGameScene](#) is a custom QGraphicsScene that shows a simple rendering of the game map.

```
#include <simplegamescene.h>
```

Inheritance diagram for Course::SimpleGameScene:



Collaboration diagram for Course::SimpleGameScene:



Signals

- void **objectClicked** (unsigned int object_id)

Public Member Functions

- **SimpleGameScene** (QWidget *qt_parent=nullptr, int width=10, int height=10, int scale=50)
Constructor for the class.
- **~SimpleGameScene** ()=default
Default destructor.
- void **setSize** (int width, int height)
Sets the map size and calls `resize()`.
- void **setScale** (int scale)
Set the tile size, aka scale of the map and calls `resize()`. Function behaviour after objects has been drawn is not specified.
- void **resize** ()
resize recalculates the bounding rectangle
- int **getScale** () const
get the size of a single tile
- std::pair< int, int > **getSize** () const
get the size of the map.
- void **drawItem** (std::shared_ptr< Course::GameObject > obj)
draw a new item to the map.
- void **removeItem** (std::shared_ptr< Course::GameObject > obj)
tries to remove drawn object at the location obj points to. If there's multiple objects, will remove the one that matches obj.
- void **updateItem** (std::shared_ptr< Course::GameObject > obj)
updates the position of obj.
- virtual bool **event** (QEvent *event) override
simple event handler that notifies when objects or the play area is clicked.

6.51.1 Detailed Description

The **SimpleGameScene** is a custom QGraphicsScene that shows a simple rendering of the game map.

6.51.2 Constructor & Destructor Documentation

6.51.2.1 SimpleGameScene()

```
Course::SimpleGameScene::SimpleGameScene (
    QWidget * qt_parent = nullptr,
    int width = 10,
    int height = 10,
    int scale = 50 )
```

Constructor for the class.

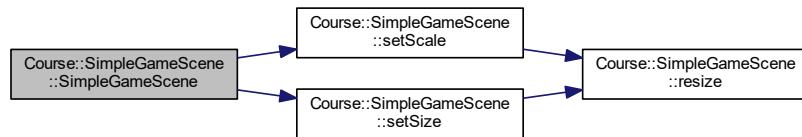
Parameters

<i>qt_parent</i>	points to the parent object per Qt's parent-child-system.
<i>width</i>	in tiles for the game map.
<i>height</i>	in tiles for the game map.
<i>scale</i>	is the size in pixels of a single square tile.

Precondition

$0 < \text{width} \leq 100 \ \&\& 0 < \text{height} \leq 100 \ \&\& 0 < \text{scale} \leq 500$. Otherwise default values are used for the created object.

Here is the call graph for this function:

**6.51.3 Member Function Documentation****6.51.3.1 drawItem()**

```
void Course::SimpleGameScene::drawItem (
    std::shared_ptr< Course::GameObject > obj )
```

draw a new item to the map.

Parameters

<i>obj</i>	shared ptr to the object
------------	--------------------------

Precondition

obj must have a valid coordinate property.

Postcondition

Exception guarantee: None

6.51.3.2 event()

```
bool Course::SimpleGameScene::event (
    QEvent * event ) [override], [virtual]
```

simple event handler that notifies when objects or the play area is clicked.

Parameters

<i>event</i>	that has happened.
--------------	--------------------

Returns

True: if event was handled in the handler. False: if the event handling was passed over.

6.51.3.3 getScale()

```
int Course::SimpleGameScene::getScale ( ) const
```

get the size of a single tile

Returns

the size of a tile in pixels.

Postcondition

Exception guarantee: No-throw

6.51.3.4 getSize()

```
std::pair< int, int > Course::SimpleGameScene::getSize ( ) const
```

get the size of the map.

Returns

pair<width, height> in tiles.

Postcondition

Exception guarantee: No-throw

6.51.3.5 removeItem()

```
void Course::SimpleGameScene::removeItem (   
    std::shared_ptr< Course::GameObject > obj )
```

tries to remove drawn object at the location obj points to. If there's multiple objects, will remove the one that matches obj.

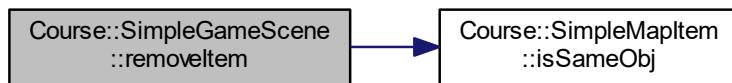
Parameters

<i>obj</i>	shared ptr to the object being deleted.
------------	---

Postcondition

Exception guarantee: None

Here is the call graph for this function:



6.51.3.6 setScale()

```
void Course::SimpleGameScene::setScale ( int scale )
```

set the tile size, aka scale of the map and calls [resize\(\)](#). Function behaviour after objects has been drawn is not specified.

Parameters

<i>scale</i>	in pixels.
--------------	------------

Precondition

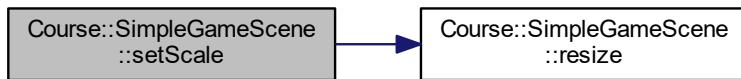
$0 < \text{scale} \leq 500$

Postcondition

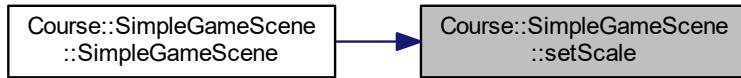
Scene scale is set to scale.

Exception guarantee: None

Here is the call graph for this function:



Here is the caller graph for this function:



6.51.3.7 `setSize()`

```
void Course::SimpleGameScene::setSize ( int width, int height )
```

Sets the map size and calls [resize\(\)](#).

Parameters

<i>width</i>	in tiles.
<i>height</i>	in tiles.

Precondition

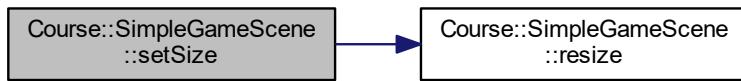
width and *height* must each be between 1 and 100.

Postcondition

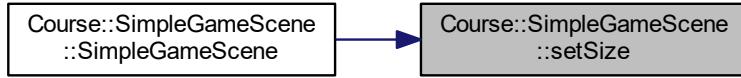
width and height are set to given sizes.

Exception guarantee: No-throw

Here is the call graph for this function:



Here is the caller graph for this function:



6.51.3.8 updateItem()

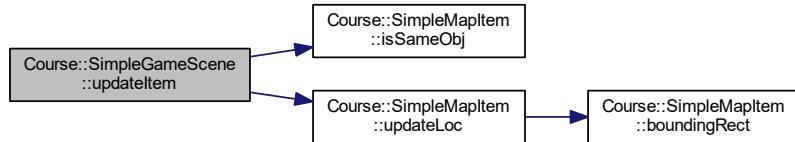
```
void Course::SimpleGameScene::updateItem ( std::shared_ptr< Course::GameObject > obj )
```

updates the position of obj.

Parameters

<i>obj</i>	shared ptr to the obj being updated.
------------	--------------------------------------

Here is the call graph for this function:



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

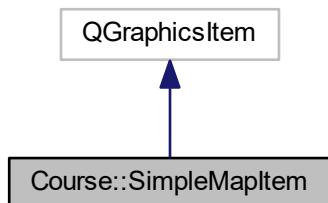
- E:/Gitin repo/ohjelmointi 3/whiskas/Course/CourseLib/graphics/simplegamescene.h
- E:/Gitin repo/ohjelmointi 3/whiskas/Course/CourseLib/graphics/simplegamescene.cpp

6.52 Course::SimpleMapItem Class Reference

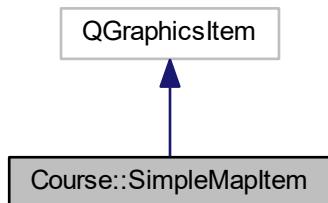
The [SimpleMapItem](#) class is a custom QGraphicsItem that acts as a single [GameObject](#)'s graphical element.

```
#include <simplemapitem.h>
```

Inheritance diagram for Course::SimpleMapItem:



Collaboration diagram for Course::SimpleMapItem:



Public Member Functions

- `SimpleMapItem (const std::shared_ptr< Course::GameObject > &obj, int size)`
Constructor.
- `QRectF boundingRect () const override`
boundingRect
- `void paint (QPainter *painter, const QStyleOptionGraphicsItem *option, QWidget *widget)`
paints the item
- `const std::shared_ptr< Course::GameObject > & getBoundObject ()`
getBoundObject
- `void updateLoc ()`
updateLoc moves the item if the position has changed.
- `bool isSameObj (std::shared_ptr< Course::GameObject > obj)`
checks if this instance has obj as bound obj.
- `int getSize () const`
getSize
- `void setSize (int size)`
setSize

6.52.1 Detailed Description

The `SimpleMapItem` class is a custom QGraphicsItem that acts as a single `GameObject`'s graphical element.

6.52.2 Constructor & Destructor Documentation

6.52.2.1 SimpleMapItem()

```
Course::SimpleMapItem::SimpleMapItem (
    const std::shared_ptr< Course::GameObject > & obj,
    int size )
```

Constructor.

Parameters

<code>obj</code>	shared_ptr to the obj.
<code>size</code>	of the created item in pixels.

Precondition

`obj` must have a valid `Coordinate`.

6.52.3 Member Function Documentation

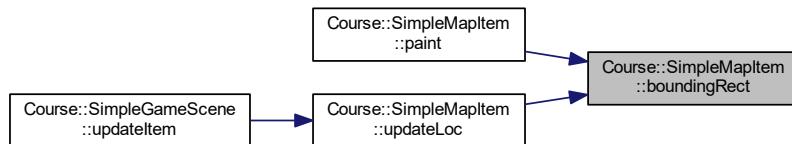
6.52.3.1 boundingRect()

```
QRectF Course::SimpleMapItem::boundingRect () const [override]
boundingRect
```

Returns

the bounding rectangle of this item.

Here is the caller graph for this function:



6.52.3.2 getBoundObject()

```
const std::shared_ptr< Course::GameObject > & Course::SimpleMapItem::getBoundObject ()
```

getBoundObject

Returns

the object this item is bound to.

6.52.3.3 getSize()

```
int Course::SimpleMapItem::getSize () const
getSize
```

Returns

size of the object in pixels.

Postcondition

Exception guarantee: No-throw

6.52.3.4 isSameObj()

```
bool Course::SimpleMapItem::isSameObj (
    std::shared_ptr< Course::GameObject > obj )
```

checks if this instance has obj as bound obj.

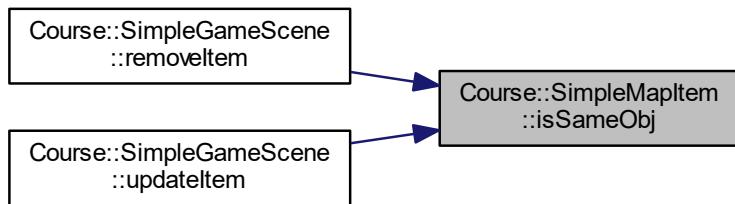
Parameters

<i>obj</i>	to compare to.
------------	----------------

Returns

True: if obj is pointing to the same object as this item. False otherwise.

Here is the caller graph for this function:



6.52.3.5 `paint()`

```
void Course::SimpleMapItem::paint (
    QPainter * painter,
    const QStyleOptionGraphicsItem * option,
    QWidget * widget )
```

paints the item

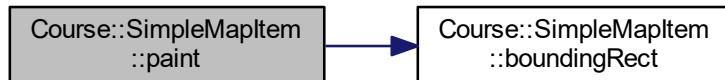
Parameters

<i>painter</i>	
<i>option</i>	
<i>widget</i>	

Note

The GraphicsView containing the scene this belongs to usually calls this function.

Here is the call graph for this function:



6.52.3.6 setSize()

```
void Course::SimpleMapItem::setSize (
    int size )
```

setSize

Parameters

<i>size</i>	of the object in pixels.
-------------	--------------------------

Precondition

$0 < \text{size} \leq 500$

Postcondition

Exception guarantee: No-throw

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

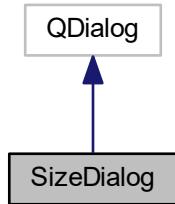
- E:/Gitin repo/ohjelmointi 3/whiskas/Course/CourseLib/graphics/simplemapitem.h
- E:/Gitin repo/ohjelmointi 3/whiskas/Course/CourseLib/graphics/simplemapitem.cpp

6.53 SizeDialog Class Reference

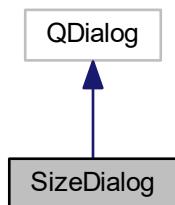
The [SizeDialog](#) class Sets the mapSize.

```
#include <sizedialog.h>
```

Inheritance diagram for SizeDialog:



Collaboration diagram for SizeDialog:



Signals

- void **Msize** (int)

Public Member Functions

- **SizeDialog** (QWidget *parent=nullptr)

6.53.1 Detailed Description

The [SizeDialog](#) class Sets the mapSize.

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

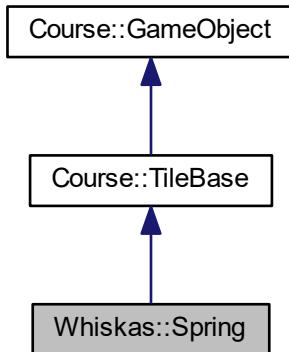
- E:/Gitin repo/ohjelmointi 3/whiskas/Game/dialogs/sizedialog.h
- E:/Gitin repo/ohjelmointi 3/whiskas/Game/dialogs/sizedialog.cpp

6.54 Whiskas::Spring Class Reference

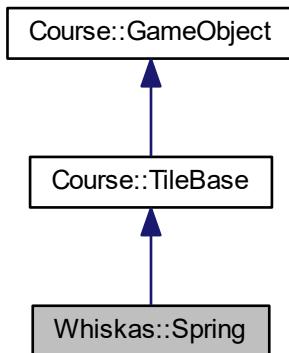
The [Spring](#) class represents the spring in the gameworld. The spring supports mage altars and lifepumps.

```
#include <spring.h>
```

Inheritance diagram for Whiskas::Spring:



Collaboration diagram for Whiskas::Spring:



Public Member Functions

- `Spring (const Course::Coordinate &location, const std::shared_ptr< Course::iEventHandler > &eventHandler, const std::shared_ptr< Course::iObjectManager > &objectManager, const unsigned int &maxBuild=1, const unsigned int &maxWork=1, const Course::ResourceMap &production={})`
- `std::string getType () const override`
- `void addBuilding (const std::shared_ptr< Course::BuildingBase > &building) override`

Adds a new Building-object to the tile.

Additional Inherited Members

6.54.1 Detailed Description

The [Spring](#) class represents the spring in the gameworld. The spring supports mage altars and lifepumps.

6.54.2 Member Function Documentation

6.54.2.1 addBuilding()

```
void Whiskas::Spring::addBuilding (
    const std::shared_ptr< Course::BuildingBase > & building ) [override], [virtual]
```

Adds a new Building-object to the tile.

Phases:

1. Tile checks if it has space for the building.
2. Building checks whether it can be placed on this tile.
3. Building is added to this Tile.
4. Tile update's Building's location.

Parameters

<i>building</i>	A pointer to the Building that is being added.
-----------------	--

Postcondition

Exception guarantee: Basic

Exceptions

<i>InvalidPointer</i>	- If the building's pointer doesn't point to anything or ObjectManager doesn't return valid shared_ptr to this tile.
<i>IllegalMove</i>	- Any IllegalException can be thrown by a Tile or Building if it breaks a placement rule.
<i>NotEnoughSpace</i>	(IllegalMove) - If the tile doesn't have enough space for the Building.

Reimplemented from [Course::TileBase](#).

6.54.2.2 getType()

```
std::string Whiskas::Spring::getType ( ) const [override], [virtual]
```

Reimplemented from [Course::TileBase](#).

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

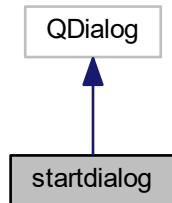
- E:/Gitin repo/ohjelmointi 3/whiskas/Game/tiles/spring.h
- E:/Gitin repo/ohjelmointi 3/whiskas/Game/tiles/spring.cpp

6.55 startdialog Class Reference

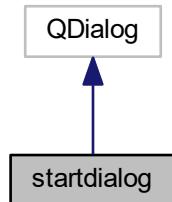
The startdialog class Selector dialog for the game start.

```
#include <startdialog.h>
```

Inheritance diagram for startdialog:



Collaboration diagram for startdialog:



Public Slots

- void **setXY** (int xy)

Signals

- void **size** (int, int)

Public Member Functions

- **startdialog** (QWidget *parent=nullptr)

6.55.1 Detailed Description

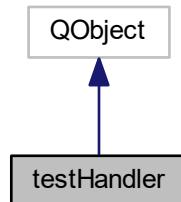
The startdialog class Selector dialog for the game start.

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

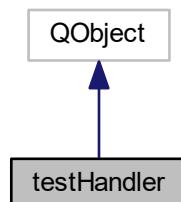
- E:/Gitin repo/ohjelmointi 3/whiskas/Game/dialogs/startdialog.h
- E:/Gitin repo/ohjelmointi 3/whiskas/Game/dialogs/startdialog.cpp

6.56 testHandler Class Reference

Inheritance diagram for testHandler:



Collaboration diagram for testHandler:

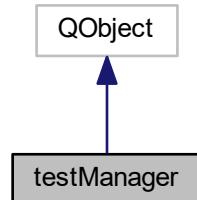


The documentation for this class was generated from the following file:

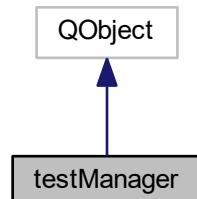
- E:/Gitin repo/ohjelmointi 3/whiskas/UnitTests/handlerTest/tst_testhandler.cpp

6.57 testManager Class Reference

Inheritance diagram for testManager:



Collaboration diagram for testManager:



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

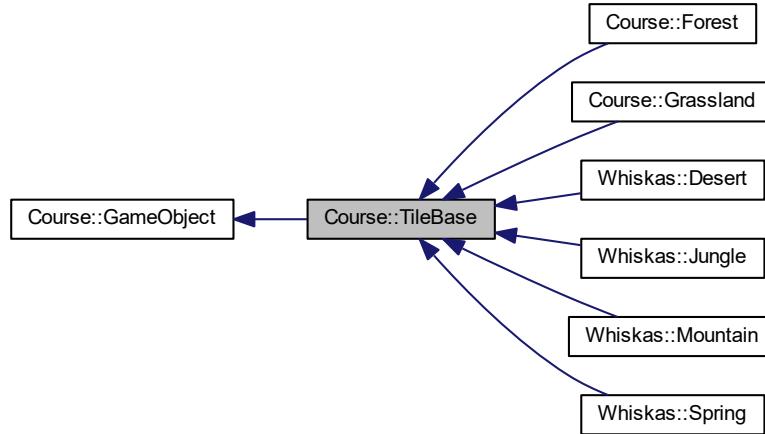
- E:/Gitin repo/ohjelmointi 3/whiskas/UnitTests/managerTest/tst_testmanager.cpp

6.58 Course::TileBase Class Reference

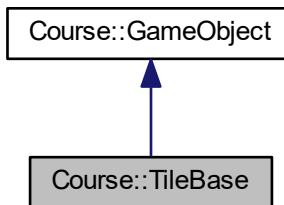
The [TileBase](#) class is a base-class for different Tile-objects in the game.

```
#include <tilebase.h>
```

Inheritance diagram for Course::TileBase:



Collaboration diagram for Course::TileBase:



Public Member Functions

- `TileBase ()=delete`
Disabled parameterless constructor.
- `TileBase (const Coordinate &location, const std::shared_ptr< iGameEventHandler > &eventhandler, const std::shared_ptr< iObjectManager > &objectmanager, const unsigned int &max_build=2, const unsigned int &max_work=3, const ResourceMap &production={})`
Constructor for the class.
- `~TileBase () override=default`
Default destructor.
- `std::string getType () const override`
getType Returns a string describing objects type. This should be overriden in each inherited class. Makes checking object's type easier for students.
- `virtual void addBuilding (const std::shared_ptr< BuildingBase > &building)`
Adds a new Building-object to the tile.

- virtual void `removeBuilding` (const std::shared_ptr< `BuildingBase` > &building)
Removes a Building-object from this Tile.
- virtual void `addWorker` (const std::shared_ptr< `WorkerBase` > &worker)
Adds a new Worker-object to this Tile.
- virtual void `removeWorker` (const std::shared_ptr< `WorkerBase` > &worker)
Removes a Worker-object from this Tile.
- virtual bool `generateResources` ()
Sends information to the EventHandler on what resources were generated by this Tile.
- virtual unsigned int `getBuildingCount` () const final
Returns the amount of spaces that are being taken from the building-capacity.
- virtual unsigned int `getWorkerCount` () const final
Returns the amount of spaces that are being taken from the worker-capacity.
- virtual bool `hasSpaceForWorkers` (int amount) const final
Checks if the tile has enough space for workers.
- virtual bool `hasSpaceForBuildings` (int amount) const final
Checks if the tile has enough space for buildings.
- virtual std::vector< std::shared_ptr< `WorkerBase` > > `getWorkers` () const final
Returns a vector of pointers to Workers in the Tile.
- virtual std::vector< std::shared_ptr< `BuildingBase` > > `getBuildings` () const final
Returns a vector of pointer to Buildings in the Tile.

Public Attributes

- const unsigned int **MAX_BUILDINGS**
- const unsigned int **MAX_WORKERS**
- const ResourceMap **BASE_PRODUCTION**

Additional Inherited Members

6.58.1 Detailed Description

The `TileBase` class is a base-class for different Tile-objects in the game.

Tile is responsible for:

- Generating resources.
- Checking Tile-specific object placement rules.
 Each Tile has some Base-production which is multiplied by worker's efficiency, when generating resources.
 Resource generation can also gain flat bonuses from buildings. Tiles also know how many Buildings or Workers can be placed on them.

6.58.2 Constructor & Destructor Documentation

6.58.2.1 TileBase()

```
Course::TileBase::TileBase (
    const Coordinate & location,
    const std::shared_ptr< iGameEventHandler > & eventhandler,
    const std::shared_ptr< iObjectManager > & objectmanager,
    const unsigned int & max_build = 2,
    const unsigned int & max_work = 3,
    const ResourceMap & production = {} )
```

Constructor for the class.

6.58.3 Member Function Documentation

6.58.3.1 addBuilding()

```
void Course::TileBase::addBuilding (
    const std::shared_ptr< BuildingBase > & building ) [virtual]
```

Adds a new Building-object to the tile.

Phases:

1. Tile checks if it has space for the building.
2. Building checks whether it can be placed on this tile.
3. Building is added to this Tile.
4. Tile update's Building's location.

Parameters

<i>building</i>	A pointer to the Building that is being added.
-----------------	--

Postcondition

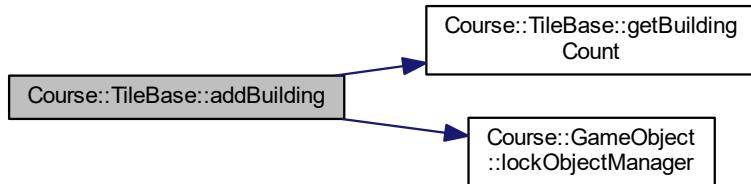
Exception guarantee: Basic

Exceptions

<i>InvalidPointer</i>	- If the building's pointer doesn't point to anything or ObjectManager doesn't return valid shared_ptr to this tile.
<i>IllegalMove</i>	- Any IllegalException can be thrown by a Tile or Building if it breaks a placement rule.
<i>NotEnoughSpace</i>	(IllegalMove) - If the tile doesn't have enough space for the Building.

Reimplemented in [Whiskas::Mountain](#), [Whiskas::Spring](#), [Whiskas::Desert](#), and [Course::Forest](#).

Here is the call graph for this function:



Here is the caller graph for this function:



6.58.3.2 addWorker()

```
void Course::TileBase::addWorker (
    const std::shared_ptr< WorkerBase > & worker ) [virtual]
```

Adds a new Worker-object to this Tile.

Phases:

1. Tile checks if it has space for the Worker.
2. Worker checks whether it can be placed on this Tile.
3. Worker is added to this Tile.
4. Tile updates Worker's location.

Parameters

<code>worker</code>	A pointer to the Worker-object that is being added.
---------------------	---

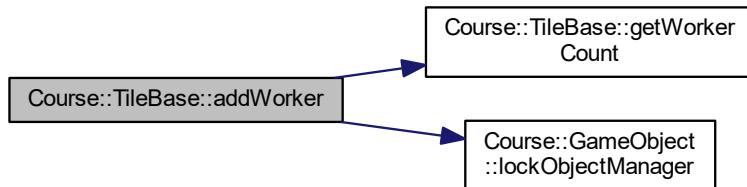
Postcondition

Exception guarantee: Basic

Exceptions

<code>InvalidPointer</code>	- If the Worker's pointer doesn't point to anything or ObjectManager doesn't return valid shared_ptr to this tile.
<code>IllegalMove</code>	- Any IllegalException can be thrown by a Tile or Worker if it breaks a placement rule.
<code>NotEnoughSpace</code>	(IllegalMove) - If the tile doesn't have enough space for the Worker.

Here is the call graph for this function:

**6.58.3.3 generateResources()**

```
bool Course::TileBase::generateResources ( ) [virtual]
```

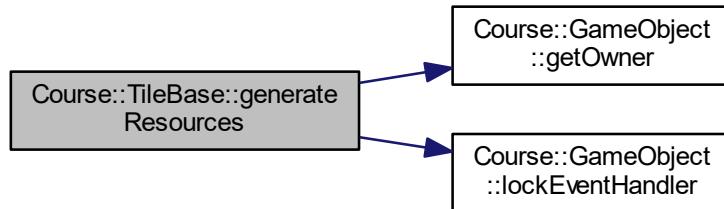
Sends information to the EventHandler on what resources were generated by this Tile.

1. Call tileWorkAction for each Worker.
2. Calculate Tile's production based on Base-production multiplied by Workers' efficiency.
3. Calls buildings' getProduction() and adds the flat bonus.
4. Sends information to GameEventHandler.
5. Returns GameEventHandler's response.

Postcondition

Exception guarantee: Basic

Here is the call graph for this function:

**6.58.3.4 getBuildingCount()**

```
unsigned int Course::TileBase::getBuildingCount ( ) const [final], [virtual]
```

Returns the amount of spaces that are being taken from the building-capacity.

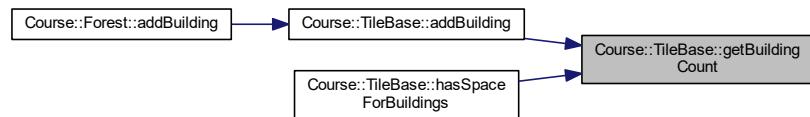
Returns

The amount of space taken.

Postcondition

Exception guarantee: No-throw

Here is the caller graph for this function:



6.58.3.5 getBuildings()

```
std::vector< std::shared_ptr< BuildingBase > > Course::TileBase::getBuildings ( ) const [final],  
[virtual]
```

Returns a vector of pointer to Buildings in the Tile.

Postcondition

Exception Guarantee: No-throw

6.58.3.6 getType()

```
std::string Course::TileBase::getType ( ) const [override], [virtual]
```

getType Returns a string describing objects type. This should be overriden in each inherited class. Makes checking object's type easier for students.

Returns

std::string that represents Object's type.

Postcondition

Exception guarantee: No-throw

Note

You can use this in e.g. debugging and similar printing.

Reimplemented from [Course::GameObject](#).

Reimplemented in [Whiskas::Jungle](#), [Whiskas::Mountain](#), [Whiskas::Spring](#), and [Whiskas::Desert](#).

6.58.3.7 getWorkerCount()

```
unsigned int Course::TileBase::getWorkerCount ( ) const [final], [virtual]
```

Returns the amount of spaces that are being taken from the worker-capacity.

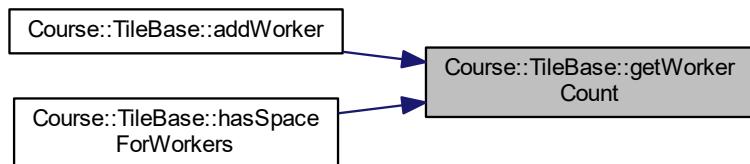
Returns

The amount of space taken.

Postcondition

Exception guarantee: No-throw

Here is the caller graph for this function:



6.58.3.8 getWorkers()

```
std::vector< std::shared_ptr< WorkerBase > > Course::TileBase::getWorkers ( ) const [final], [virtual]
```

Returns a vector of pointers to Workers in the Tile.

Postcondition

Exception Guarantee: No-throw

6.58.3.9 hasSpaceForBuildings()

```
bool Course::TileBase::hasSpaceForBuildings (
    int amount ) const [final], [virtual]
```

Checks if the tile has enough space for buildings.

Parameters

<i>amount</i>	Amount of buildingspace wanted.
---------------	---------------------------------

Note

Uses getMaxBuildings()

Postcondition

Exception guarantee: No-throw

Here is the call graph for this function:

**6.58.3.10 hasSpaceForWorkers()**

```
bool Course::TileBase::hasSpaceForWorkers (
    int amount ) const [final], [virtual]
```

Checks if the tile has enough space for workers.

Parameters

<i>amount</i>	Amount of workerspace wanted.
---------------	-------------------------------

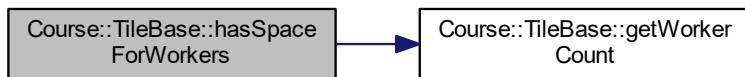
Note

Uses getMaxWorkers()

Postcondition

Exception guarantee: No-throw

Here is the call graph for this function:

**6.58.3.11 removeBuilding()**

```
void Course::TileBase::removeBuilding (
    const std::shared_ptr< BuildingBase > & building ) [virtual]
```

Removes a Building-object from this Tile.

Phases:

1. Reset the Building's location to nothing.
2. Remove the Building from m_buildings.

Parameters

<i>building</i>	A pointer to the Building-object being removed.
-----------------	---

Postcondition

Exception guarantee: Basic

Exceptions

<i>InvalidPointer</i>	- If the Building's pointer doesn't point to anything.
-----------------------	--

6.58.3.12 removeWorker()

```
void Course::TileBase::removeWorker (
```

```
const std::shared_ptr< WorkerBase > & worker ) [virtual]
```

Removes a Worker-object from this Tile.

Phases:

1. Reset the Worker's location to nothing.
2. Remove the Worker from this Tile.

Parameters

<code>worker</code>	A pointer to the Worker-object being removed.
---------------------	---

Exceptions

<code>InvalidPointer</code>	- If the Worker's pointer doesn't point to anything.
-----------------------------	--

Postcondition

Exception guarantee: Basic @exceptions See `std::vector::erase`

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

- E:/Gitin repo/ohjelmointi 3/whiskas/Course/CourseLib/tiles/tilebase.h
- E:/Gitin repo/ohjelmointi 3/whiskas/Course/CourseLib/tiles/tilebase.cpp

6.59 Whiskas::Turn Class Reference

The [Turn](#) class Handles the turn event. Updates the player inventories, resets attacked and moved for units.

```
#include <turn.h>
```

Public Member Functions

- [`Turn`](#) (`const std::shared_ptr< gameManager > &manager)`
Turn Constructor.
- `std::shared_ptr< LeaguePlayer > getInTurn ()`
getInTurn Get the current player
- `void swapTurn ()`
swapTurn Swap the player in turn, and reset the attacked and moved statuses Generate resources for players
- `int getTurnCounter ()`
getTurnCounter
- `void setInTurn (std::shared_ptr< LeaguePlayer > ToBeInTurn)`
setInTurn forfefully modify the current player

6.59.1 Detailed Description

The [Turn](#) class Handles the turn event. Updates the player inventories, resets attacked and moved for units.

6.59.2 Constructor & Destructor Documentation

6.59.2.1 Turn()

```
Whiskas::Turn::Turn (
    const std::shared_ptr< gameManager > & manager )
```

[Turn](#) Constructor.

Parameters

<i>manager</i>	The gameManager
----------------	---------------------------------

6.59.3 Member Function Documentation

6.59.3.1 getInTurn()

```
std::shared_ptr< LeaguePlayer > Whiskas::Turn::getInTurn ( )
```

getInTurn Get the current player

Returns

Ptr to current player

6.59.3.2 getTurnCounter()

```
int Whiskas::Turn::getTurnCounter ( )
```

getTurnCounter

Returns

the current turn number

6.59.3.3 setInTurn()

```
void Whiskas::Turn::setInTurn (
    std::shared_ptr< LeaguePlayer > ToBeInTurn )
```

setInTurn forfefully modify the current player

Parameters

<i>ToBeInTurn</i>	<input type="checkbox"/>
-------------------	--------------------------

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

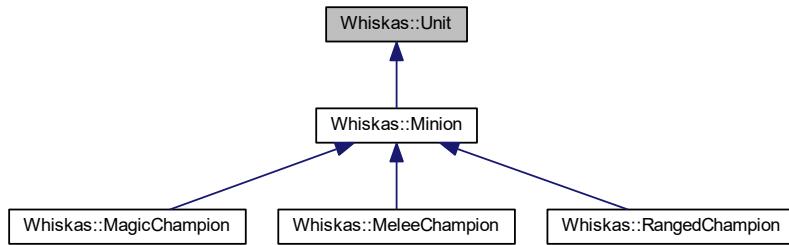
- E:/Gitin repo/ohjelmointi 3/whiskas/Game/turn.h
- E:/Gitin repo/ohjelmointi 3/whiskas/Game/turn.cpp

6.60 Whiskas::Unit Class Reference

The [Unit](#) class handles the move values.

```
#include <unit.h>
```

Inheritance diagram for Whiskas::Unit:



Public Member Functions

- **Unit** (int movement)
- int [getMoveValue \(\)](#)
getMoveValue
- void [setMoved](#) (bool hasMoved)
setMoved modifies *moved* boolean
- bool [getMoved \(\)](#)

Protected Attributes

- int **movement_**
- bool **hasMoved_** =false

6.60.1 Detailed Description

The [Unit](#) class handles the move values.

6.60.2 Member Function Documentation

6.60.2.1 getMoveValue()

```
int Whiskas::Unit::getMoveValue ( )  
  
getMoveValue
```

Returns

unit move value

6.60.2.2 setMoved()

```
void Whiskas::Unit::setMoved (   
    bool hasMoved )
```

setMoved modifies moved boolean

Parameters

<i>hasMoved</i>	<input type="checkbox"/>
-----------------	--------------------------

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

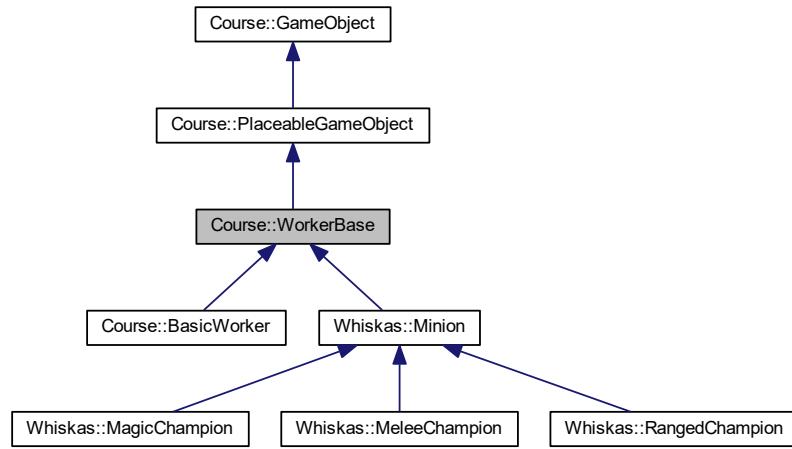
- E:/Gitin repo/ohjelmointi 3/whiskas/Game/minion/unit.h
- E:/Gitin repo/ohjelmointi 3/whiskas/Game/minion/unit.cpp

6.61 Course::WorkerBase Class Reference

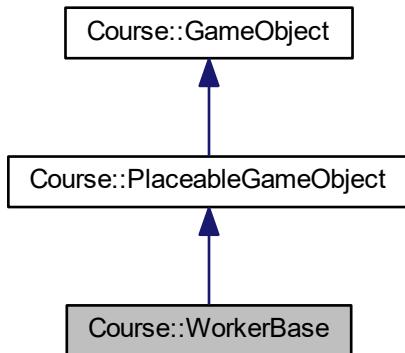
The [WorkerBase](#) class is an abstract base-class for Worker-objects.

```
#include <workerbase.h>
```

Inheritance diagram for Course::WorkerBase:



Collaboration diagram for Course::WorkerBase:



Public Member Functions

- **WorkerBase ()=delete**
Disabled parameterless constructor.
- **WorkerBase (const std::shared_ptr< iGameEventHandler > &eventhandler, const std::shared_ptr< iObjectManager > &objectmanager, const std::shared_ptr< PlayerBase > &owner, const int &tilespaces=1, const ResourceMap &cost={}, const ResourceMapDouble &efficiency={})**
- **~WorkerBase () override=default**
Default destructor.
- **virtual const ResourceMapDouble tileWorkAction ()**
Performs worker's action when working a Tile for resources.

- virtual void `doSpecialAction ()=0`
Performs the Worker's special action. (If any)
- virtual void `setResourceFocus (BasicResource new_focus) final`
Sets new resourcetype for the worker to focus on.
- virtual BasicResource `getResourceFocus () const final`
Returns the currently focused resourcetype.
- bool `canBePlacedOnTile (const std::shared_ptr< TileBase > &target) const override`
Default placement rule for workers:
- std::string `getType () const override`
GetType Returns a string describing objects type. This should be overriden in each inherited class. Makes checking object's type easier for students.

Public Attributes

- const ResourceMapDouble **WORKER_EFFICIENCY**
- const ResourceMap **RECRUITMENT_COST**

Additional Inherited Members

6.61.1 Detailed Description

The `WorkerBase` class is an abstract base-class for Worker-objects.

- Workers can perform actions
- Tiles request production-multipliers from Workers

6.61.2 Member Function Documentation

6.61.2.1 `canBePlacedOnTile()`

```
bool Course::WorkerBase::canBePlacedOnTile (
    const std::shared_ptr< TileBase > & target ) const [override], [virtual]
```

Default placement rule for workers:

- Tile must have space for the worker
- Tile must have same owner as the worker

Parameters

<code>target</code>	is the Tile that worker is being placed on.
---------------------	---

Returns

True - Only if both conditions are met.

Postcondition

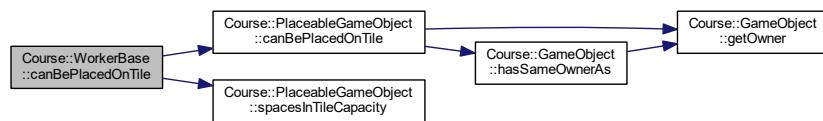
Exception guarantee: Basic

Note

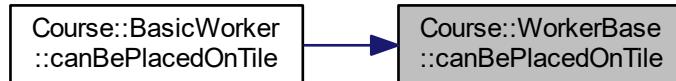
Override to change placement rules for derived worker.

Reimplemented from [Course::PlaceableGameObject](#).

Here is the call graph for this function:



Here is the caller graph for this function:

**6.61.2.2 doSpecialAction()**

```
virtual void Course::WorkerBase::doSpecialAction ( ) [pure virtual]
```

Performs the Worker's special action. (If any)

Note

Hint: You can use game-eventhandler for more creative solutions.

Implemented in [Course::BasicWorker](#), and [Whiskas::Minion](#).

6.61.2.3 getResourceFocus()

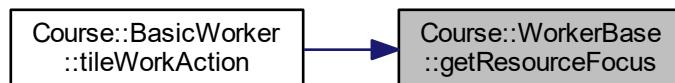
```
BasicResource Course::WorkerBase::getResourceFocus ( ) const [final], [virtual]
```

Returns the currently focused resourcetype.

Postcondition

Exception guarantee: No-throw

Here is the caller graph for this function:



6.61.2.4 getType()

```
std::string Course::WorkerBase::getType ( ) const [override], [virtual]
```

`getType` Returns a string describing objects type. This should be overriden in each inherited class. Makes checking object's type easier for students.

Returns

`std::string` that represents Object's type.

Postcondition

Exception guarantee: No-throw

Note

You can use this in e.g. debugging and similar printing.

Reimplemented from [Course::PlaceableGameObject](#).

Reimplemented in [Whiskas::Minion](#), [Whiskas::MeleeChampion](#), [Whiskas::MagicChampion](#), and [Whiskas::RangedChampion](#).

6.61.2.5 setResourceFocus()

```
void Course::WorkerBase::setResourceFocus (
    BasicResource new_focus ) [final], [virtual]
```

Sets new resourcetype for the worker to focus on.

Parameters

<i>new_focus</i>	the new resource focus.
------------------	-------------------------

Postcondition

Exception guarantee: No-throw

6.61.2.6 tileWorkAction()

```
const ResourceMapDouble Course::WorkerBase::tileWorkAction () [virtual]
```

Performs worker's action when working a Tile for resources.

Returns

Returns the final working efficiency of a worker.

Note

This is called by a Tile when it generates resources.

Reimplemented in [Course::BasicWorker](#).

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

- E:/Gitin repo/ohjelmointi 3/whiskas/Course/CourseLib/workers/workerbase.h
- E:/Gitin repo/ohjelmointi 3/whiskas/Course/CourseLib/workers/workerbase.cpp

6.62 Course::WorldGenerator Class Reference

The [WorldGenerator](#) class is a singleton for generating tiles for the game.

```
#include <worldgenerator.h>
```

Public Member Functions

- [WorldGenerator](#) (const [WorldGenerator](#) &)=delete
- [WorldGenerator](#) & [operator=](#) (const [WorldGenerator](#) &)=delete
- [WorldGenerator](#) ([WorldGenerator](#) &&)=delete
- [WorldGenerator](#) & [operator=](#) ([WorldGenerator](#) &&)=delete
- template<typename T >
void [addConstructor](#) (unsigned int weight)

Register a Tile's constructor for use in map generation.
- void [generateMap](#) (unsigned int size_x, unsigned int size_y, unsigned int seed, const std::shared_ptr<iObjectManager > &objectmanager, const std::shared_ptr<iGameEventHandler > &eventhandler)

Generates Tile-objects and sends them to ObjectManager.

Static Public Member Functions

- static [WorldGenerator & getInstance \(\)](#)
Used to get a reference to the Singleton instance.

6.62.1 Detailed Description

The [WorldGenerator](#) class is a singleton for generating tiles for the game.

Students use this to create new Tile-objects that form the gameworld.

Usage:

1. Use [WorldGenerator::getInstance\(\)](#) to get a reference to the singleton.
2. Call addConstructor with each Tile type you created.
3. Generate the map through the reference.

6.62.2 Member Function Documentation

6.62.2.1 addConstructor()

```
template<typename T >
void Course::WorldGenerator::addConstructor (
    unsigned int weight ) [inline]
```

Register a Tile's constructor for use in map generation.

Note

Do this only once per Tile type or they won't be equally common. Use the Tile's type as the template parameter: [addConstructor<Forest>\(\)](#);

Parameters

<i>weight</i>	represents the rarity of the Tile, high being common.
---------------	---

6.62.2.2 generateMap()

```
void Course::WorldGenerator::generateMap (
    unsigned int size_x,
    unsigned int size_y,
    unsigned int seed,
```

```
const std::shared_ptr< iObjectManager > & objectmanager,
const std::shared_ptr< iGameEventHandler > & eventhandler )
```

Generates Tile-objects and sends them to ObjectManager.

Parameters

<i>size_x</i>	is the horizontal size of the map area.
<i>size_y</i>	is the vertical size of the map area.
<i>seed</i>	is the seed-value used in the generation.
<i>objectmanager</i>	points to the ObjectManager that receives the generated Tiles.
<i>eventhandler</i>	points to the student's GameEventHandler.

Postcondition

Exception guarantee: No-throw

6.62.2.3 getInstance()

```
WorldGenerator & Course::WorldGenerator::getInstance ( ) [static]
```

Used to get a reference to the Singleton instance.

Returns

Reference to the Singleton instance.

Postcondition

Exception guarantee: No-throw

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

- E:/Gitin repo/ohjelmointi 3/whiskas/Course/CourseLib/core/worldgenerator.h
- E:/Gitin repo/ohjelmointi 3/whiskas/Course/CourseLib/core/worldgenerator.cpp

Index

~Coordinate
 Course::Coordinate, 34

~IllegalAction
 Course::IllegalAction, 113

~InvalidPointer
 Course::InvalidPointer, 115

~KeyError
 Course::KeyError, 121

~NotEnoughSpace
 Course::NotEnoughSpace, 152

~OwnerConflict
 Course::OwnerConflict, 158

addBuilding
 Course::Forest, 61
 Course::TileBase, 199
 Whiskas::Desert, 54
 Whiskas::gameManager, 71
 Whiskas::Mountain, 147
 Whiskas::Spring, 193

addConstructor
 Course::WorldGenerator, 216

addDescription
 Course::GameObject, 85

addHoldMarkers
 Course::BuildingBase, 28

addMinion
 Whiskas::gameManager, 71

addObject
 Course::PlayerBase, 165

addObjects
 Course::PlayerBase, 166

addPlayer
 Whiskas::gameManager, 72

addTile
 Whiskas::gameManager, 72

addTiles
 Course::iObjectManager, 116
 Whiskas::gameManager, 72

addWorker
 Course::TileBase, 200

attack
 Whiskas::gameManager, 73

Attackable
 Whiskas::Attackable, 16

attackMultiple
 Whiskas::gameManager, 73

BaseException
 Course::BaseException, 20

BasicWorker
 Course::BasicWorker, 23

boundingRect
 Course::SimpleMapItem, 187
 Whiskas::MapItem, 132

BuildingBase
 Course::BuildingBase, 27

buildingDialog, 31

canBePlacedOnTile
 Course::BasicWorker, 23
 Course::BuildingBase, 28
 Course::PlaceableGameObject, 160
 Course::WorkerBase, 212

checkBuildingAvailability
 Whiskas::gameManager, 74

checkForEnemies
 Whiskas::gameManager, 75

Coordinate
 Course::Coordinate, 33

Course::BaseException, 19
 BaseException, 20
 msg, 20

Course::BasicWorker, 21
 BasicWorker, 23
 canBePlacedOnTile, 23
 getType, 24
 tileWorkAction, 24

Course::BuildingBase, 25
 addHoldMarkers, 28
 BuildingBase, 27
 canBePlacedOnTile, 28
 getProduction, 29
 getType, 29
 holdCount, 30

Course::Coordinate, 32
 ~Coordinate, 34
 Coordinate, 33
 neighbour_at, 34
 neighbours, 35
 operator!=, 35
 operator<, 39
 operator<=, 40
 operator>, 42
 operator>=, 43
 operator+, 36
 operator+=, 37
 operator-, 38
 operator-=, 38
 operator=, 40

operator==, 41
 set_x, 43
 set_y, 43
 travel, 44
 x, 45
 y, 46
Course::Farm, 56
 Farm, 57
 getType, 58
Course::Forest, 59
 addBuilding, 61
 Forest, 60
 getType, 61
Course::GameObject, 81
 addDescription, 85
 GameObject, 84, 85
 getCoordinate, 86
 getCoordinatePtr, 86
 getDescription, 87
 getDescriptions, 88
 getOwner, 88
 getType, 88
 hasSameCoordinateAs, 89
 hasSameOwnerAs, 89
 lockEventHandler, 90
 lockObjectManager, 91
 removeDescription, 92
 removeDescriptions, 92
 setCoordinate, 92, 93
 setDescription, 93
 setDescriptions, 94
 setOwner, 94
 unsetCoordinate, 95
Course::Grassland, 102
 getType, 104
 Grassland, 104
Course::HeadQuarters, 105
 getType, 107
 HeadQuarters, 107
 onBuildAction, 108
Course::iGameEventHandler, 109
 modifyResource, 109
 modifyResources, 111
Course::IllegalAction, 112
 ~IllegalAction, 113
 IllegalAction, 113
Course::InvalidPointer, 113
 ~InvalidPointer, 115
 InvalidPointer, 115
Course::iObjectManager, 115
 addTiles, 116
 getTile, 116, 117
 getTiles, 117
Course::KeyError, 120
 ~KeyError, 121
 KeyError, 121
Course::NotEnoughSpace, 150
 ~NotEnoughSpace, 152
 NotEnoughSpace, 151
Course::Outpost, 152
 getProduction, 154
 getType, 154
 onBuildAction, 155
 Outpost, 154
Course::OwnerConflict, 156
 ~OwnerConflict, 158
 OwnerConflict, 157
Course::PlaceableGameObject, 158
 canBePlacedOnTile, 160
 currentLocationTile, 161
 getType, 161
 PlaceableGameObject, 159
 setLocationTile, 162
 spacesInTileCapacity, 163
Course::PlayerBase, 164
 addObject, 165
 addObjects, 166
 getName, 166
 getObjects, 167
 PlayerBase, 165
 removeObject, 167
 removeObjects, 168, 169
 setName, 169
Course::SimpleGameScene, 179
 drawItem, 181
 event, 181
 getScale, 182
 getSize, 182
 removeItem, 182
 setScale, 183
 setSize, 184
 SimpleGameScene, 180
 updateItem, 185
Course::SimpleMapItem, 186
 boundingRect, 187
 getBoundObject, 188
 getSize, 188
 isSameObj, 188
 paint, 189
 setSize, 190
 SimpleMapItem, 187
Course::TileBase, 196
 addBuilding, 199
 addWorker, 200
 generateResources, 201
 getBuildingCount, 202
 getBuildings, 202
 getType, 203
 getWorkerCount, 203
 getWorkers, 204
 hasSpaceForBuildings, 204
 hasSpaceForWorkers, 205
 removeBuilding, 206
 removeWorker, 206
 TileBase, 198
Course::WorkerBase, 210

canBePlacedOnTile, 212
doSpecialAction, 213
getResourceFocus, 213
getType, 214
setResourceFocus, 214
tileWorkAction, 215
Course::WorldGenerator, 215
 addConstructor, 216
 generateMap, 216
 getInstance, 217
currentLocationTile
 Course::PlaceableGameObject, 161

default_coordinate, 49
default_gameobjects, 50
default_playerbase, 51
default_test, 52
destroyObject
 Whiskas::gameManager, 75
doSpecialAction
 Course::WorkerBase, 213
drawItem
 Course::SimpleGameScene, 181
 MapWindow, 136
 Whiskas::GameScene, 98

endDialog, 55
endTurn
 Whiskas::gameEventHandler, 64
event
 Course::SimpleGameScene, 181
 Whiskas::GameScene, 98

Farm
 Course::Farm, 57
Forest
 Course::Forest, 60

gameEventHandler
 Whiskas::gameEventHandler, 64
gameManager
 Whiskas::gameManager, 71
GameObject
 Course::GameObject, 84, 85
GameScene
 Whiskas::GameScene, 96
generateMap
 Course::WorldGenerator, 216
generateResources
 Course::TileBase, 201
getActiveMinion
 Whiskas::gameEventHandler, 64
getActiveTile
 Whiskas::gameEventHandler, 65
getAttack
 Whiskas::Attackable, 17
getAttacked
 Whiskas::Attackable, 17
getBoundID
 Whiskas::Attackable, 17
getBoundObject
 Course::SimpleMapItem, 188
 Whiskas::MapItem, 132
getBuildingCount
 Course::TileBase, 202
getBuildings
 Course::TileBase, 202
getBuildingVector
 Whiskas::gameManager, 76
getCoordinate
 Course::GameObject, 86
getCoordinatePtr
 Course::GameObject, 86
getDescription
 Course::GameObject, 87
getDescriptions
 Course::GameObject, 88
getHealth
 Whiskas::Attackable, 17
getInstance
 Course::WorldGenerator, 217
getInTurn
 Whiskas::Turn, 208
getLastCoordinate
 Whiskas::GameScene, 99
getLastID
 Whiskas::GameScene, 99
getMinionVector
 Whiskas::gameManager, 76
getMoveValue
 Whiskas::Unit, 210
getName
 Course::PlayerBase, 166
getNexusLocation
 Whiskas::gameManager, 76
getObjects
 Course::PlayerBase, 167
getOwner
 Course::GameObject, 88
getPlayerPair
 Whiskas::gameManager, 77
getProduction
 Course::BuildingBase, 29
 Course::Outpost, 154
getResourceFocus
 Course::WorkerBase, 213
getScale
 Course::SimpleGameScene, 182
getSize
 Course::SimpleGameScene, 182
 Course::SimpleMapItem, 188
getTile
 Course::iObjectManager, 116, 117
 Whiskas::gameManager, 77
getTiles
 Course::iObjectManager, 117
 Whiskas::gameManager, 78

getTileVector
 Whiskas::gameManager, 78

getTurn
 Whiskas::gameEventHandler, 65

getTurnCounter
 Whiskas::Turn, 208

getType
 Course::BasicWorker, 24
 Course::BuildingBase, 29
 Course::Farm, 58
 Course::Forest, 61
 Course::GameObject, 88
 Course::Grassland, 104
 Course::HeadQuarters, 107
 Course::Outpost, 154
 Course::PlaceableGameObject, 161
 Course::TileBase, 203
 Course::WorkerBase, 214
 Whiskas::AltarBase, 15
 Whiskas::Desert, 54
 Whiskas::Jungle, 119
 Whiskas::Lifepump, 125
 Whiskas::MageAltar, 127
 Whiskas::MagicChampion, 129
 Whiskas::MeleeAltar, 140
 Whiskas::MeleeChampion, 142
 Whiskas::Minion, 144
 Whiskas::Mountain, 147
 Whiskas::Nexus, 149
 Whiskas::Quarry, 171
 Whiskas::RangedAltar, 173
 Whiskas::RangedChampion, 175
 Whiskas::Sawmill, 178
 Whiskas::Spring, 193

getWinner
 Whiskas::gameManager, 78

getWorkerCount
 Course::TileBase, 203

getWorkers
 Course::TileBase, 204

Grassland
 Course::Grassland, 104

handleLeftClick
 Whiskas::gameEventHandler, 65

handleMwindowClick
 Whiskas::gameEventHandler, 66

handleRightClick
 Whiskas::gameEventHandler, 66

hasSameCoordinateAs
 Course::GameObject, 89

hasSameOwnerAs
 Course::GameObject, 89

hasSpaceForBuildings
 Course::TileBase, 204

hasSpaceForWorkers
 Course::TileBase, 205

HeadQuarters
 Course::HeadQuarters, 107

holdCount
 Course::BuildingBase, 30

IllegalAction
 Course::IllegalAction, 113

initMap
 MapWindow, 136

InvalidPointer
 Course::InvalidPointer, 115

isSameObj
 Course::SimpleMapItem, 188
 Whiskas::MapItem, 133

KeyError
 Course::KeyError, 121

LeaguePlayer
 Whiskas::LeaguePlayer, 122

lockEventHandler
 Course::GameObject, 90

lockObjectManager
 Course::GameObject, 91

ManagerTest, 130

MapItem
 Whiskas::MapItem, 132

MapWindow, 134
 drawItem, 136
 initMap, 136
 mousePressEvent, 137
 openBuildingDialog, 137
 setGEHandler, 137
 setGManager, 137
 setSize, 138

modifyAttack
 Whiskas::Attackable, 18

modifyHealth
 Whiskas::Attackable, 18
 Whiskas::MeleeChampion, 142
 Whiskas::Minion, 145

modifyResource
 Course::iGameEventHandler, 109
 Whiskas::gameEventHandler, 67

modifyResources
 Course::iGameEventHandler, 111
 Whiskas::gameEventHandler, 67

mousePressEvent
 MapWindow, 137

move
 Whiskas::gameManager, 79

msg
 Course::BaseException, 20

neighbour_at
 Course::Coordinate, 34

neighbours
 Course::Coordinate, 35

NotEnoughSpace
 Course::NotEnoughSpace, 151

onBuildAction
 Course::HeadQuarters, 108
 Course::Outpost, 155
openBuildingDialog
 MapWindow, 137
operator!=
 Course::Coordinate, 35
operator<
 Course::Coordinate, 39
operator<=
 Course::Coordinate, 40
operator>
 Course::Coordinate, 42
operator>=
 Course::Coordinate, 43
operator+
 Course::Coordinate, 36
operator+=
 Course::Coordinate, 37
operator-
 Course::Coordinate, 38
operator-=
 Course::Coordinate, 38
operator=
 Course::Coordinate, 40
operator==
 Course::Coordinate, 41
Outpost
 Course::Outpost, 154
OwnerConflict
 Course::OwnerConflict, 157

paint
 Course::SimpleMapItem, 189
 Whiskas::MapItem, 133
PlaceableGameObject
 Course::PlaceableGameObject, 159
PlayerBase
 Course::PlayerBase, 165

removeBuilding
 Course::TileBase, 206
removeDescription
 Course::GameObject, 92
removeDescriptions
 Course::GameObject, 92
removeItem
 Course::SimpleGameScene, 182
 Whiskas::GameScene, 99
removeObject
 Course::PlayerBase, 167
removeObjects
 Course::PlayerBase, 168, 169
removeWorker
 Course::TileBase, 206
Resourcemap_operationsTest, 176

selectAttackTarget
 Whiskas::gameManager, 79

set_x
 Course::Coordinate, 43
set_y
 Course::Coordinate, 43
setActiveTile
 Whiskas::gameEventHandler, 68
setAttacked
 Whiskas::Attackable, 18
setCoordinate
 Course::GameObject, 92, 93
setDescription
 Course::GameObject, 93
setDescriptions
 Course::GameObject, 94
setGEHandler
 MapWindow, 137
setGManager
 MapWindow, 137
setInTurn
 Whiskas::Turn, 208
setLocationTile
 Course::PlaceableGameObject, 162
setMoved
 Whiskas::Unit, 210
setName
 Course::PlayerBase, 169
setOwner
 Course::GameObject, 94
setPlayerItems
 Whiskas::LeaguePlayer, 123
setResourceFocus
 Course::WorkerBase, 214
setScale
 Course::SimpleGameScene, 183
 Whiskas::GameScene, 100
setSize
 Course::SimpleGameScene, 184
 Course::SimpleMapItem, 190
 MapWindow, 138
 Whiskas::GameScene, 100
 Whiskas::MapItem, 134
SimpleGameScene
 Course::SimpleGameScene, 180
SimpleMapItem
 Course::SimpleMapItem, 187
SizeDialog, 190
spacesInTileCapacity
 Course::PlaceableGameObject, 163
spawnBuilding
 Whiskas::gameManager, 80
spawnMinion
 Whiskas::gameManager, 80
startdialog, 194
subtractPlayerResources
 Whiskas::gameEventHandler, 68

testHandler, 195
testManager, 196
TileBase

Course::TileBase, 198
 tileWorkAction
 Course::BasicWorker, 24
 Course::WorkerBase, 215
 travel
 Course::Coordinate, 44
 Turn
 Whiskas::Turn, 208
 unsetCoordinate
 Course::GameObject, 95
 updateItem
 Course::SimpleGameScene, 185
 Whiskas::GameScene, 101
 Whiskas::AltarBase, 13
 getType, 15
 Whiskas::Attackable, 15
 Attackable, 16
 getAttack, 17
 getAttacked, 17
 getBoundID, 17
 getHealth, 17
 modifyAttack, 18
 modifyHealth, 18
 setAttacked, 18
 Whiskas::CustomBuildingBase, 48
 Whiskas::Desert, 52
 addBuilding, 54
 getType, 54
 Whiskas::gameEventHandler, 62
 endTurn, 64
 gameEventHandler, 64
 getActiveMinion, 64
 getActiveTile, 65
 getTurn, 65
 handleLeftClick, 65
 handleMwindowClick, 66
 handleRightClick, 66
 modifyResource, 67
 modifyResources, 67
 setActiveTile, 68
 subtractPlayerResources, 68
 Whiskas::gameManager, 68
 addBuilding, 71
 addMinion, 71
 addPlayer, 72
 addTile, 72
 addTiles, 72
 attack, 73
 attackMultiple, 73
 checkBuildingAvailability, 74
 checkForEnemies, 75
 destroyObject, 75
 gameManager, 71
 getBuildingVector, 76
 getMinionVector, 76
 getNexusLocation, 76
 getPlayerPair, 77
 getTile, 77
 getTiles, 78
 getTileVector, 78
 getWinner, 78
 move, 79
 selectAttackTarget, 79
 spawnBuilding, 80
 spawnMinion, 80
 Whiskas::GameScene, 95
 drawItem, 98
 event, 98
 GameScene, 96
 getLastCoordinate, 99
 getLastID, 99
 removeItem, 99
 setScale, 100
 setSize, 100
 updateItem, 101
 Whiskas::Jungle, 118
 getType, 119
 Whiskas::LeaguePlayer, 121
 LeaguePlayer, 122
 setPlayerItems, 123
 Whiskas::Lifepump, 123
 getType, 125
 Whiskas::MageAltar, 126
 getType, 127
 Whiskas::MagicChampion, 128
 getType, 129
 Whiskas::MapItem, 130
 boundingRect, 132
 getBoundObject, 132
 isSameObj, 133
 MapItem, 132
 paint, 133
 setSize, 134
 Whiskas::MeleeAltar, 138
 getType, 140
 Whiskas::MeleeChampion, 141
 getType, 142
 modifyHealth, 142
 Whiskas::Minion, 143
 getType, 144
 modifyHealth, 145
 Whiskas::Mountain, 145
 addBuilding, 147
 getType, 147
 Whiskas::Nexus, 148
 getType, 149
 Whiskas::Quarry, 170
 getType, 171
 Whiskas::RangedAltar, 172
 getType, 173
 Whiskas::RangedChampion, 174
 getType, 175
 Whiskas::Sawmill, 176
 getType, 178
 Whiskas::Spring, 192

addBuilding, [193](#)
getType, [193](#)
Whiskas::Turn, [207](#)
 getInTurn, [208](#)
 getTurnCounter, [208](#)
 setInTurn, [208](#)
 Turn, [208](#)
Whiskas::Unit, [209](#)
 getMoveValue, [210](#)
 setMoved, [210](#)

x

Course::Coordinate, [45](#)

y

Course::Coordinate, [46](#)