Bronze Adventures

Generated by Doxygen 1.8.16

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, detaling 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 with in 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 diffrent 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 speccial 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. Afther an upgrade, the altar will go on cooldown for a set amount of turns. Afther 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 distrubution

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 desing.

1.4 content

- Moving
- Attacking
- Special Altar buildings
- Music (Currenlty 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:

Chapter 5

Class Index

5.1 Class List

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

Chapter 6

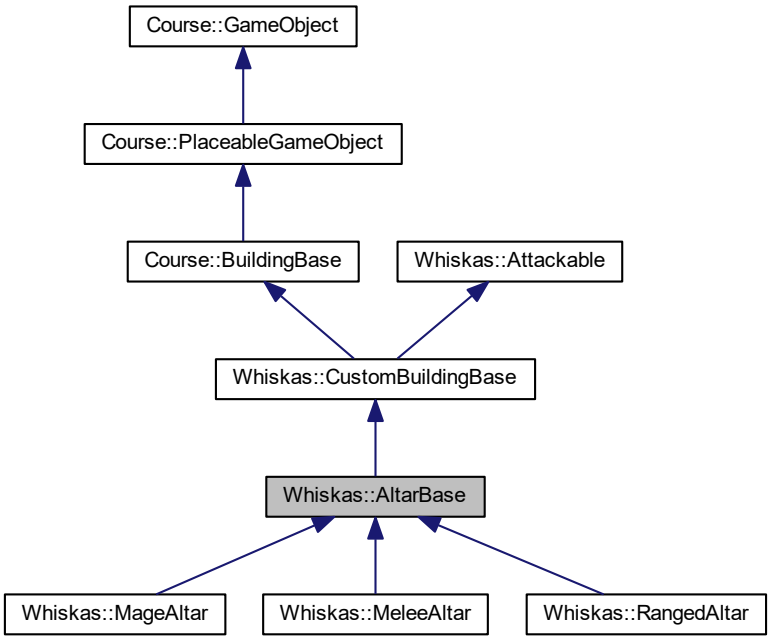
Class Documentation

6.1 Whiskas::AltarBase Class Reference

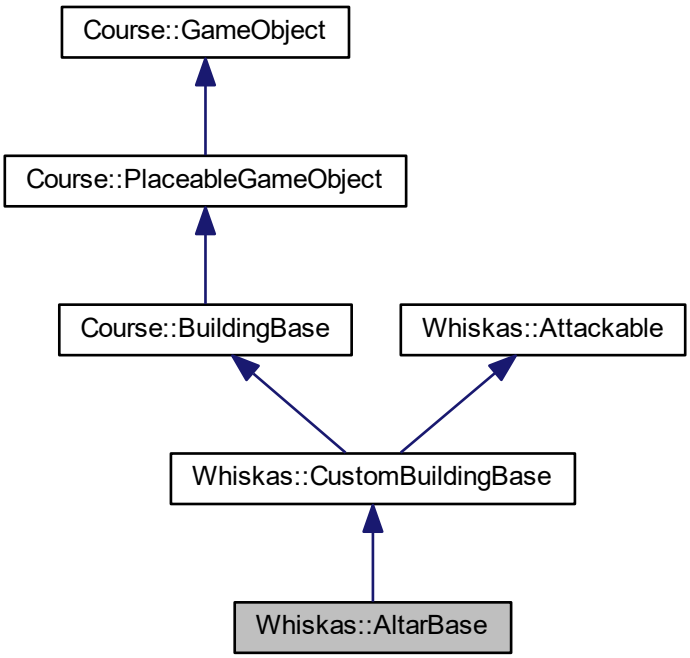
The AltarBase class is the inheritacne class for altar buildings.

`#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 inheritacne 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:

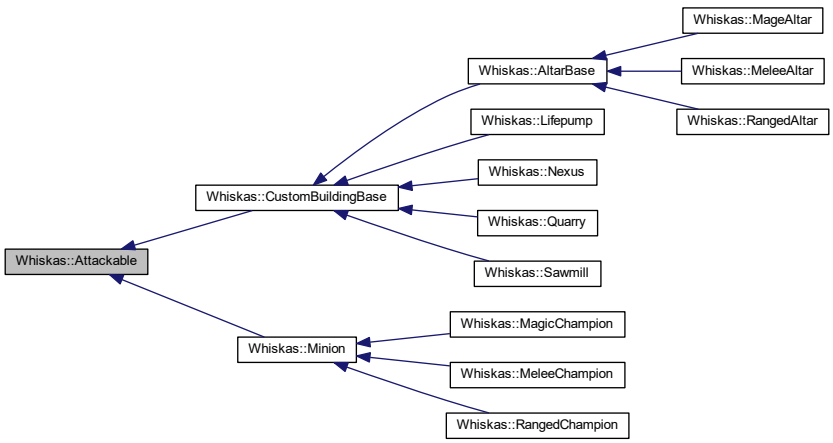
- E:/Gitin repo/ohjelmointi 3/whiskas/Game/buildings/AltarBase.h
- E:/Gitin repo/ohjelmointi 3/whiskas/Game/buildings/AltarBase.cpp

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 mofifier.
- 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

health	0=Dead
attack	damage
numberOfAttacks	how many times the unit can attack in a turn
boundID	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 mofifier.

Parameters

AModifier	

6.2.3.6 modifyHealth()

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

modifyHealth

Parameters

hModifier	

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:

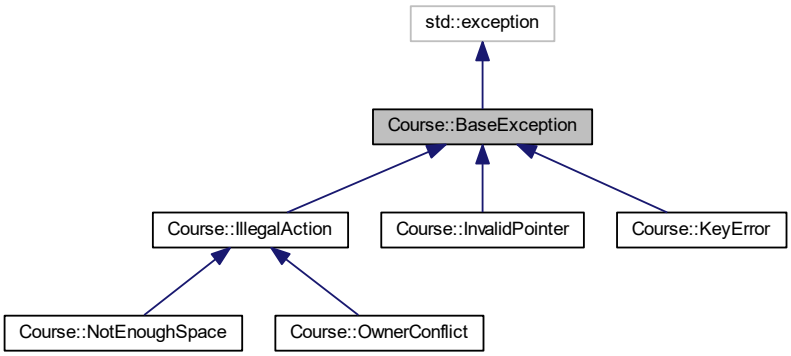
- E:/Gitin repo/ohjelmointi 3/whiskas/Game/minion/attackable.h
- E:/Gitin repo/ohjelmointi 3/whiskas/Game/minion/attackable.cpp

6.3 Course::BaseException Class Reference

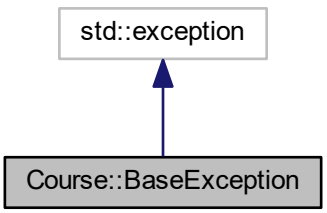
The Exception class is a base-class for custom exceptions in project.

`#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:

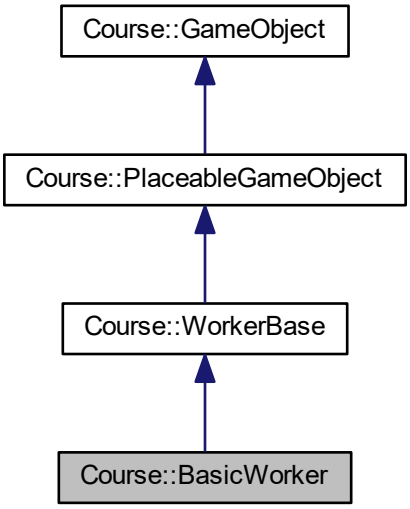
- E:/Gitin repo/ohjelmointi 3/whiskas/Course/CourseLib/exceptions/baseexception.h

6.4 Course::BasicWorker Class Reference

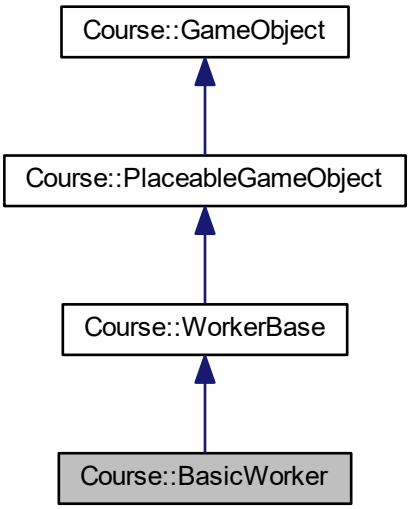
The BasicWorker class represents a "normal worker" in the game.

`#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

eventhandler	points to the student's GameEventHandler.
owner	points to the owning player.
descriptions	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

target	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

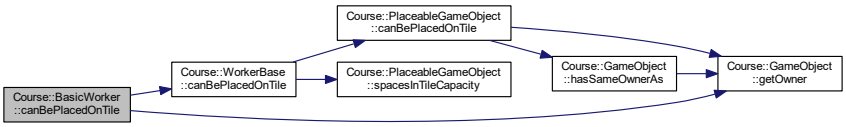
Override to change placement rules for derived worker.

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()

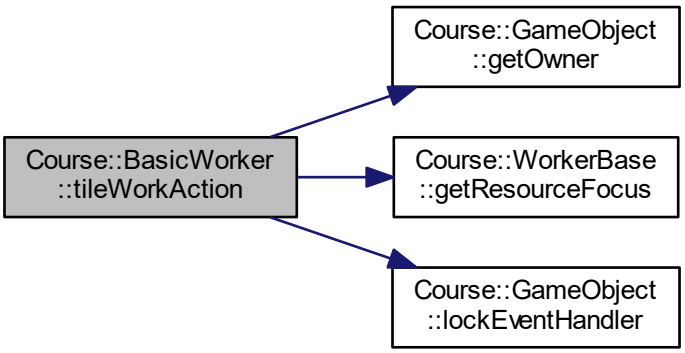
```
const ResourceMapDouble Course::BasicWorker::tileWorkAction ( )  [override], [virtual]
```

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:

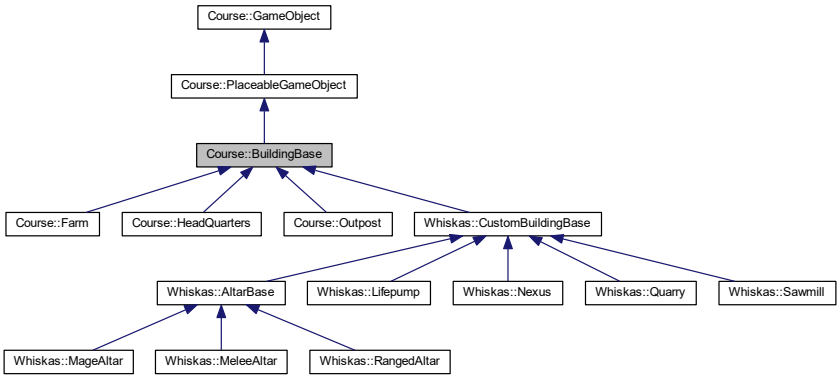
- E:/Gitin repo/ohjelmointi 3/whiskas/Course/CourseLib/workers/basicworker.h
- E:/Gitin repo/ohjelmointi 3/whiskas/Course/CourseLib/workers/basicworker.cpp

6.5 Course::BuildingBase Class Reference

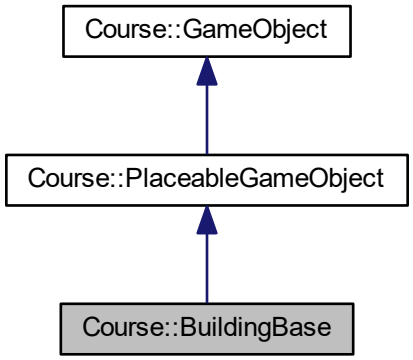
The BuildingBase class is a base-class for different buildings in the game.

```
#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 ()

Performs building's default action.

- virtual void onBuildAction ()

  Performs building's possible action after construction.

- virtual ResourceMap getProduction ()

  Return's building's production as a ResourceMap.

  .

- 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

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

Postcondition

Exception guarantee: No guarantee.

Exceptions

OwnerConflict	- 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 substraction.

Parameters

amount	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

target	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

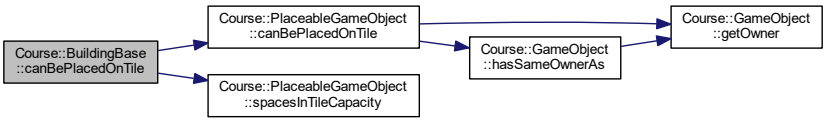
Override to modify placementrules for derived classes.

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:

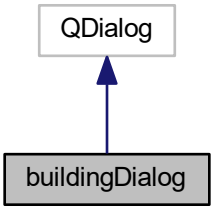
- E:/Gitin repo/ohjelmointi 3/whiskas/Course/CourseLib/buildings/buildingbase.h
- E:/Gitin repo/ohjelmointi 3/whiskas/Course/CourseLib/buildings/buildingbase.cpp

6.6 buildingDialog Class Reference

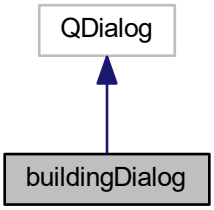
The buildingDialog class is a dialog for selecting the building to be built.

`#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 Return's the x-value
- int y () const

  y Return's 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 - Substracts 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 -= Substracts 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 doesnt' 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

x	X-coordinate
y	Y-coordinate

6.7.1.2 Coordinate() [2/3]

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

Copy-constructor.

Parameters

original	Reference to a Coordinate 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

original	Reference to Coordinate that is being copied
direction	What direction is the movement applied
steps	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

direction	Direction in x,y -coordinate system
steps	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

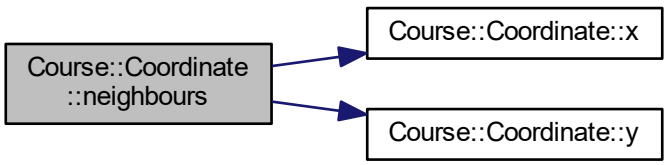
Returns

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 doesnt' have same x and y values as this

Parameters

other	The other Coordinate

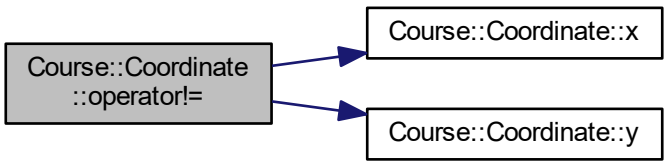
Returns

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

other	The other Coordinate

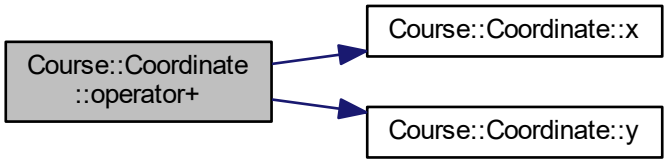
Returns

Coordinate with x = x(this) + x(other) and y = y(this) + y(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

other	The other coordinate

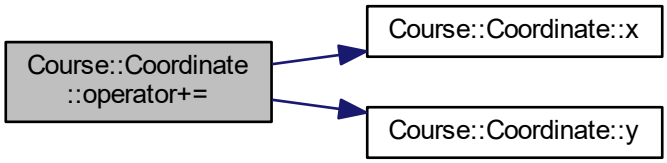
Returns

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 - Substracts 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

other	The other coordinate

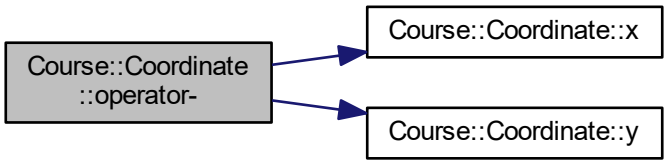
Returns

Coordinate with x = x(this) - x(other) and y = y(this) - y(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 -= Substracts other coordinate's x and y values from this

Parameters

other	The other coordinate

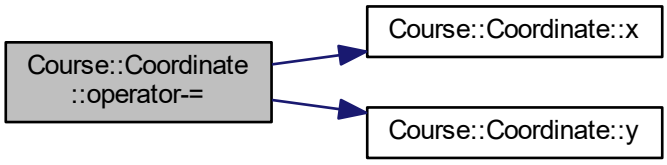
Returns

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

other	The other Coordinate

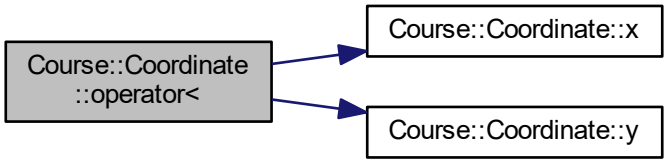
Returns

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

other	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

other	The other Coordinate

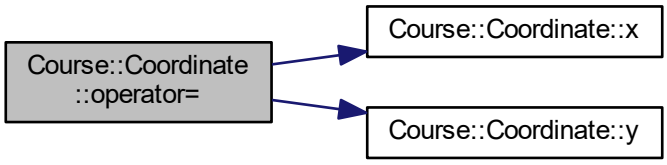
Returns

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

other	The other Coordinate

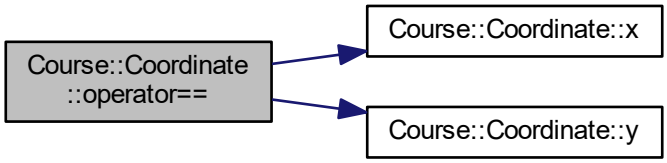
Returns

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

other	The other Coordinate

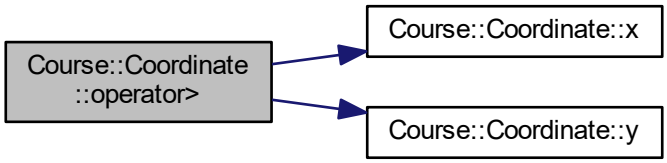
Returns

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:



6.7.2.13 operator>=()

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

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

Parameters

other	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

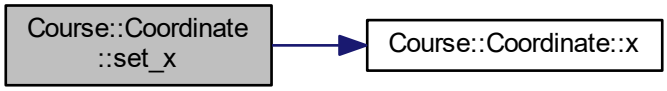
Parameters

x	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

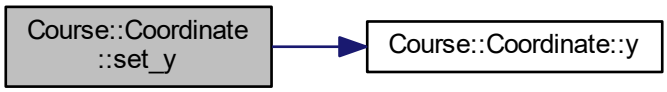
Parameters

y	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.

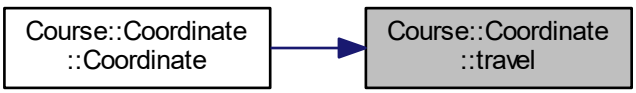
Parameters

direction	Direction in x,y -coordinate system
steps	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

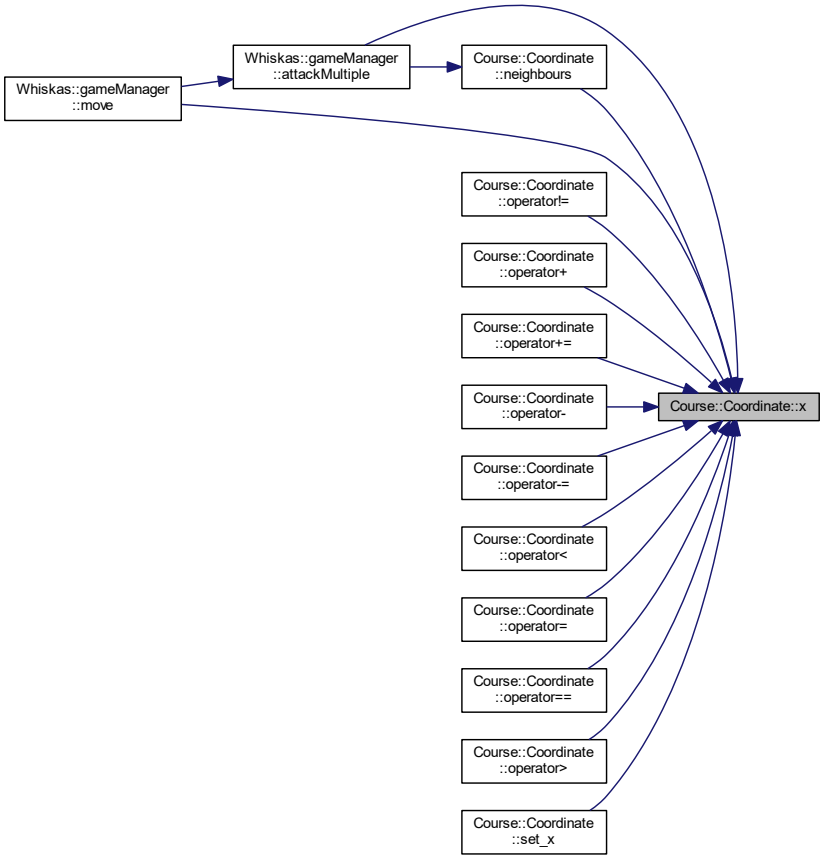
Returns

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

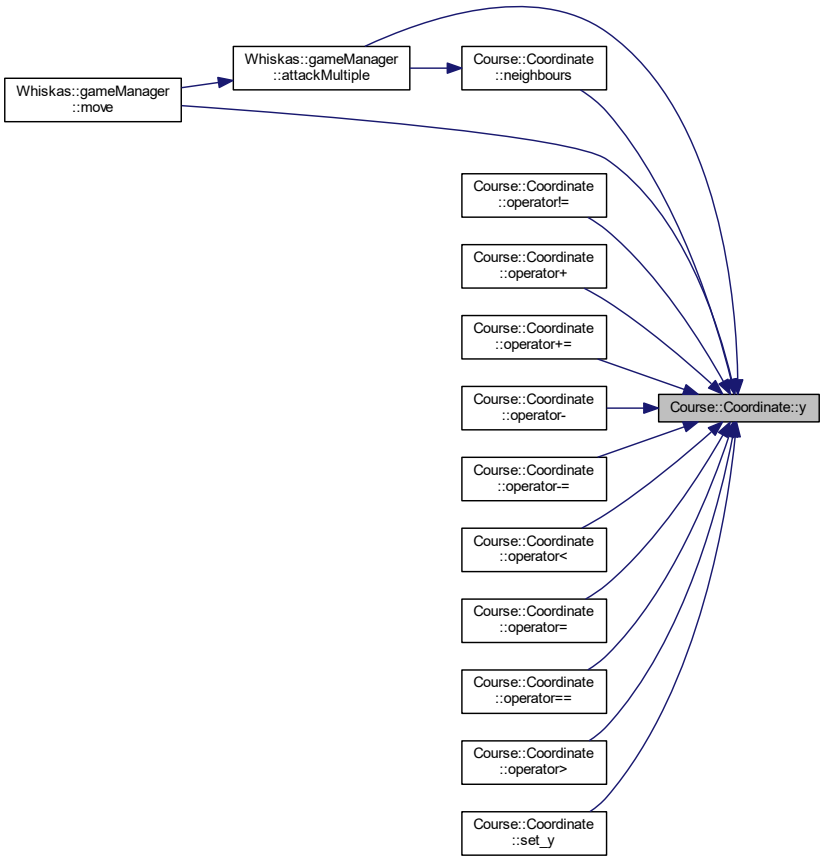
Returns

y-coordinate

Postcondition

Exception guarantee: No-throw

Here is the caller graph for this function:

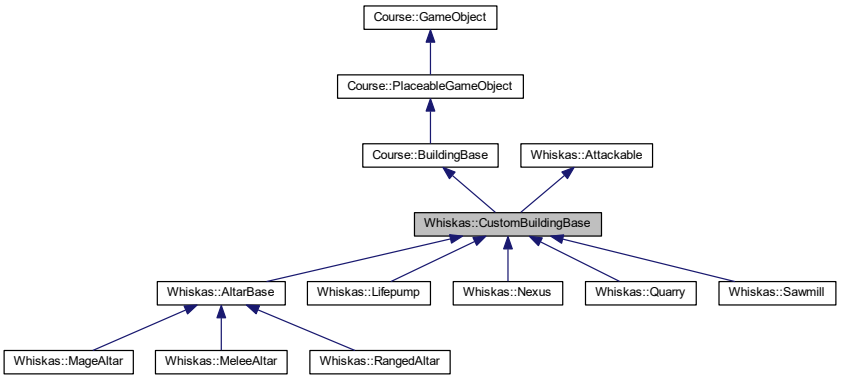


The documentation for this class was generated from the following files:

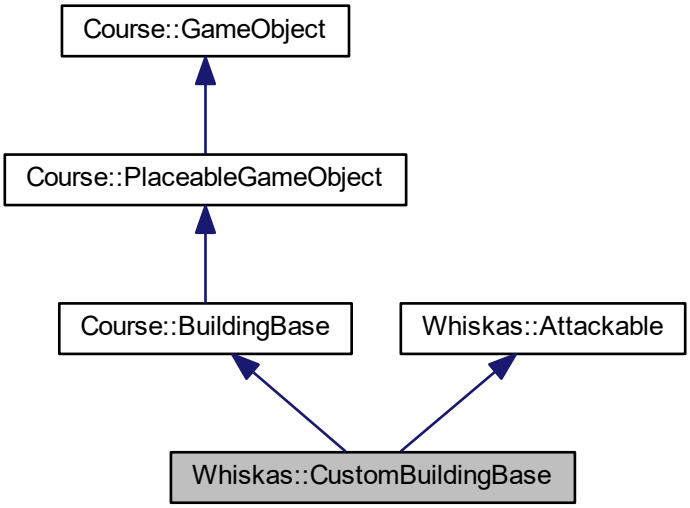
- E:/Gitin repo/ohjelmointi 3/whiskas/Course/CourseLib/core/coordinate.h
- E:/Gitin repo/ohjelmointi 3/whiskas/Course/CourseLib/core/coordinate.cpp

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

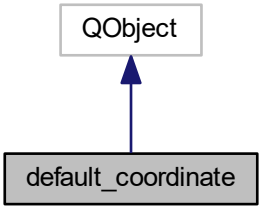
Additional Inherited Members

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

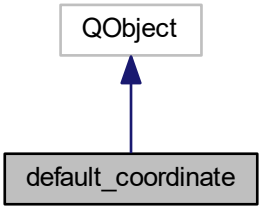
- E:/Gitin repo/ohjelmointi 3/whiskas/Game/buildings/custombuildingbase.h
- E:/Gitin repo/ohjelmointi 3/whiskas/Game/buildings/custombuildingbase.cpp

6.9 default_coordinate Class Reference

Inheritance diagram for default_coordinate:



Collaboration diagram for default_coordinate:

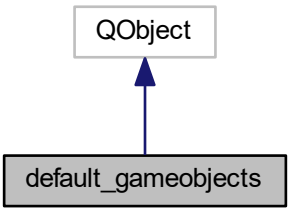


The documentation for this class was generated from the following file:

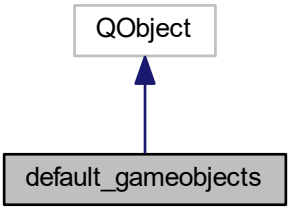
- E:/Gitin repo/ohjelmointi 3/whiskas/Course/UnitTests/coordinate_tests/tst_default_coordinate.cpp

6.10 default_gameobjects Class Reference

Inheritance diagram for default_gameobjects:



Collaboration diagram for default_gameobjects:

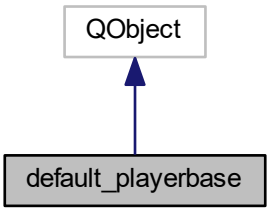


The documentation for this class was generated from the following file:

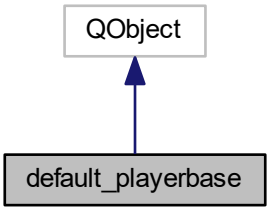
- E:/Gitin repo/ohjelmointi 3/whiskas/Course/UnitTests/gameobject_tests/tst_default_gameobjects.cpp

6.11 default_playerbase Class Reference

Inheritance diagram for default_playerbase:



Collaboration diagram for default_playerbase:

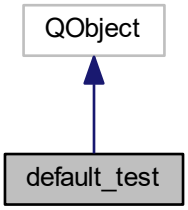


The documentation for this class was generated from the following file:

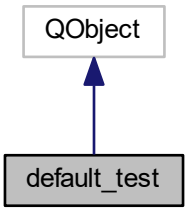
- E:/Gitin repo/ohjelmointi 3/whiskas/Course/UnitTests/playerbase_tests/tst_default_playerbase.cpp

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:

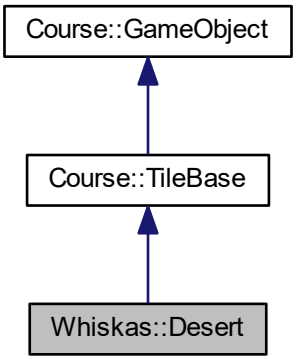
- E:/Gitin repo/ohjelmointi 3/whiskas/Course/UnitTests/placeablegameobject_tests/tst_default_test.cpp

6.13 Whiskas::Desert Class Reference

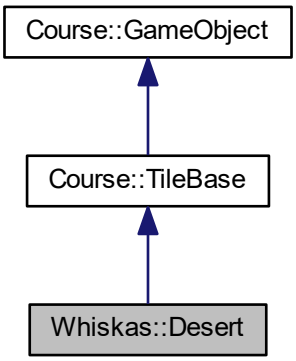
The Desert class represents the desert in the gameworld. The desert supports melee altars.

`#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

building	A pointer to the Building that is being added.

Postcondition

Exception guarantee: Basic

Exceptions

InvalidPointer	- If the building's pointer doesn't point to anything or ObjectManager doesn't return valid shared_ptr to this tile.
IllegalMove	- Any IllegalException can be thrown by a Tile or Building if it breaks a placement rule.
NotEnoughSpace	(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:

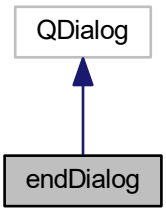
- E:/Gitin repo/ohjelmointi 3/whiskas/Game/tiles/desert.h
- E:/Gitin repo/ohjelmointi 3/whiskas/Game/tiles/desert.cpp

6.14 endDialog Class Reference

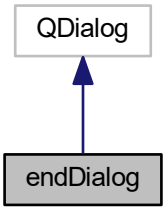
The endDialog class Shows the end message.

`#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:

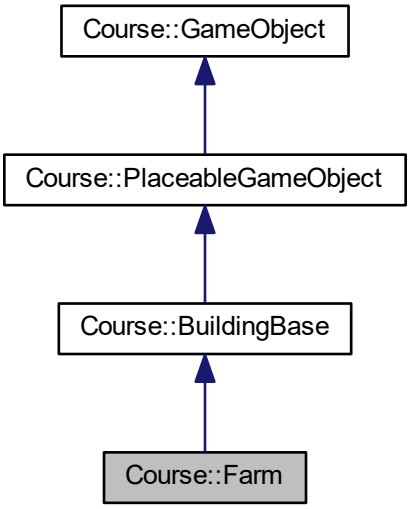
- E:/Gitin repo/ohjelmointi 3/whiskas/Game/dialogs/enddialog.h
- E:/Gitin repo/ohjelmointi 3/whiskas/Game/dialogs/enddialog.cpp

6.15 Course::Farm Class Reference

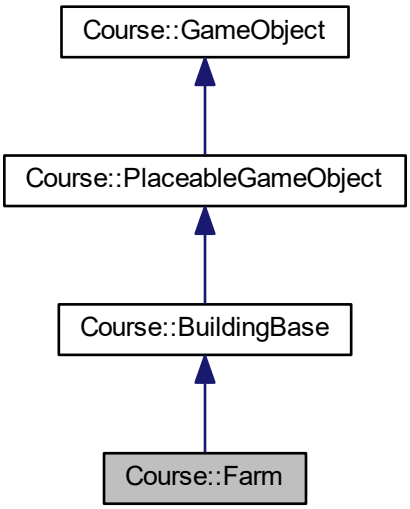
The Farm class represents a farm-building in the game.

`#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=Const↩ ResourceMaps::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

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

Postcondition

Exception Guarantee: No guarantee.

Exceptions

OwnerConflict	- 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:

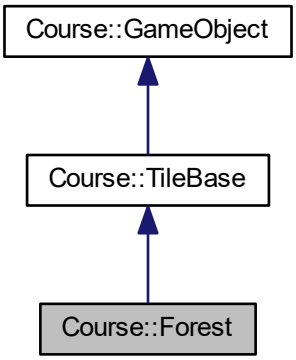
- E:/Gitin repo/ohjelmointi 3/whiskas/Course/CourseLib/buildings/farm.h
- E:/Gitin repo/ohjelmointi 3/whiskas/Course/CourseLib/buildings/farm.cpp

6.16 Course::Forest Class Reference

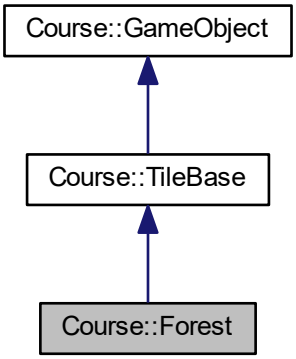
The Forest class represents Forest in the gameworld.

`#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

location	is the Coordinate where the Tile is located in the game.
eventhandler	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:

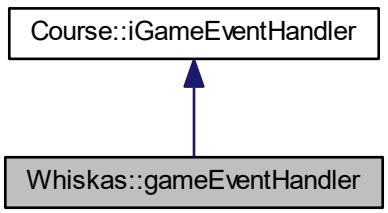
- E:/Gitin repo/ohjelmointi 3/whiskas/Course/CourseLib/tiles/forest.h
- E:/Gitin repo/ohjelmointi 3/whiskas/Course/CourseLib/tiles/forest.cpp

6.17 Whiskas::gameEventHandler Class Reference

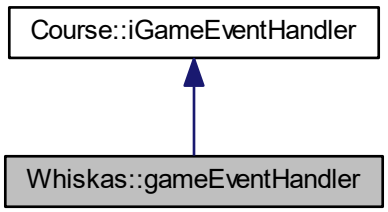
The gameEventHandler class Handles events the game can have. Most notably, the mouse presses. NOTE: The first few methods that have unused variables are there for the reason that course side documentation said that those are required by the course side code. We dont use them but we didnt know if we could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

turn	For easier accsess to the turn class
textBrowserRight	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

manager	GameManager
handler	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

scene	The game scene
manager	The gameManager

Here is the caller graph for this function:

6.17.3.6 handleMwindowClick()

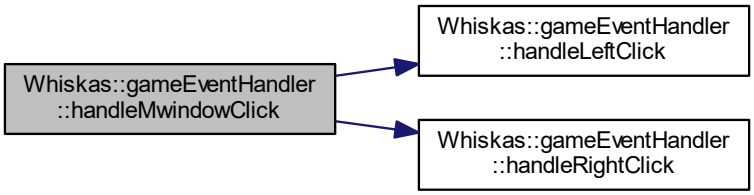
```
void Whiskas::gameEventHandler::handleMwindowClick (
            const std::shared_ptr< GameScene > & scene,
            const std::shared_ptr< gameManager > & manager,
            const QMouseEvent & event )
```

handleMwindowClick checks if the click was right, or left

Parameters

scene	passed on to other methods
manager	passed on to other methods
event	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

manager	The gameManager
scene	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

player	does nothing
resources	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

activeTile	

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

player	Target player
costs	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:

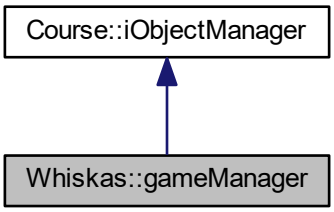
- E:/Gitin repo/ohjelmointi 3/whiskas/Game/handlerandmanager/gameeventhandler.h
- E:/Gitin repo/ohjelmointi 3/whiskas/Game/handlerandmanager/gameeventhandler.cpp

6.18 Whiskas::gameManager Class Reference

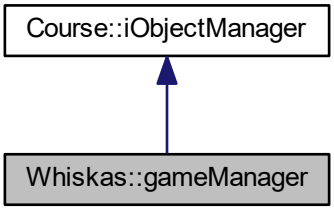
The gameManager class is a class that manages the game logic. It has access to allmost all the classes included in this game.

```
#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 ()

getBuildingVector returns all the bulding the gamemanager has.

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

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

- std::pair< std::shared_ptr< Whiskas::LeaguePlayer >, std::shared_ptr< Whiskas::LeaguePlayer > > getPlayerPair ()

  getPlayerPair Returns both players in the game

- 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 > &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.

- 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 allmost 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

m_gamescene	for easier access to graphics
browser	For easir managment 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

building	to be added

6.18.3.2 addMinion()

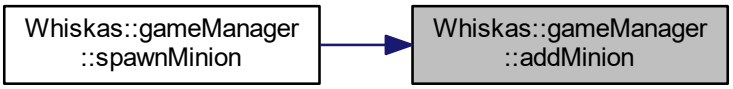
```
void Whiskas::gameManager::addMinion (
            const std::shared_ptr< Minion > & Minion )
```

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

Parameters

Minion	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

player	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

tile	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

vector	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

minionToAttack	Minion doing the attacking
toAttack	The target

Here is the call graph for this function:

Here is the caller graph for this function:

6.18.3.7 attackMultiple()

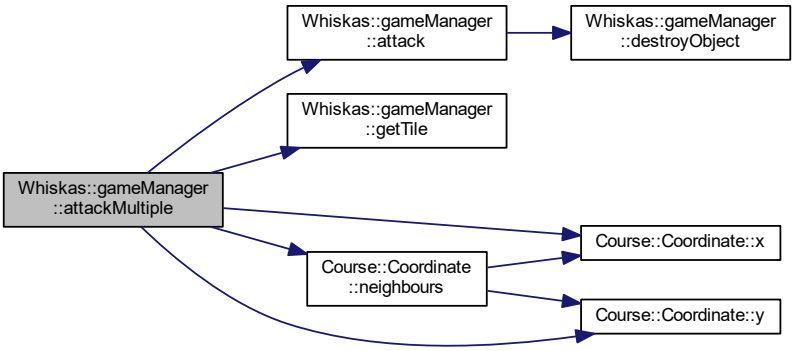
```
void Whiskas::gameManager::attackMultiple (
            const std::shared_ptr< Minion > & minionToAttack,
            const std::shared_ptr< Course::TileBase > & targetTile )
```

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

Parameters

minionToAttack	Minion doing the attacking
targetTile	

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

targetTile	
type	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::TileBase > & targetTile )
```

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

Parameters

minionTomove	Minion doing the moving
targetTile	Destination

Returns

true if enemies, false if no

Here is the caller graph for this function:

6.18.3.10 destroyObject()

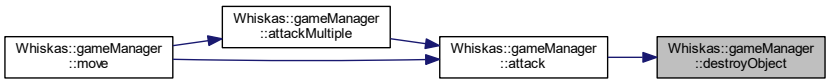
```
void Whiskas::gameManager::destroyObject (
            const std::shared_ptr< Attackable > & ObjectToDestroy )
```

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

ObjectToDestroy	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 bulding 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 excistance. 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

owner	Desired player

Returns

Tile containing the Nexus

6.18.3.14 getPlayerPair()

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

getPlayerPair 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

coordinate	

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

ID	

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

coordinates	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()

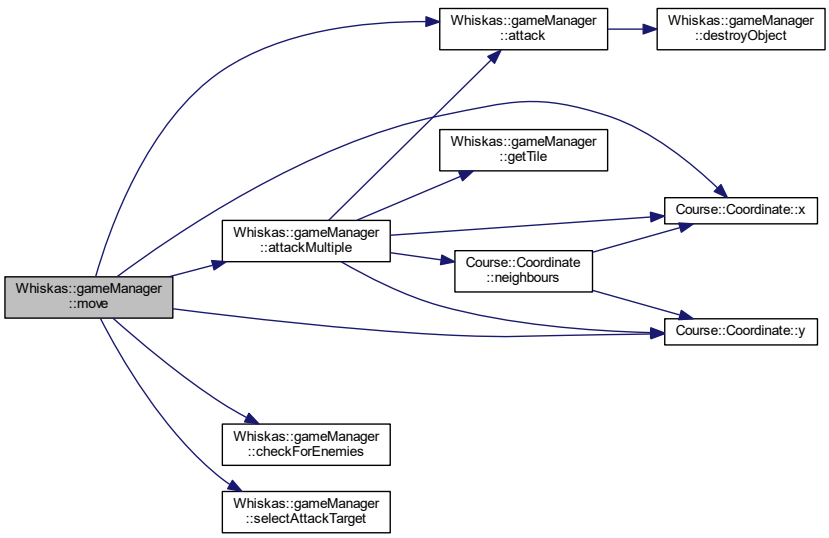
```
void Whiskas::gameManager::move (
            const std::shared_ptr< Minion > & minionToMove,
            const std::shared_ptr< Course::TileBase > & targetTile )
```

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

minionToMove	The minion doing the moving
targetTile	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

targetTile	

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

handler	buildings handler
manager	recieves the tile
player	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.

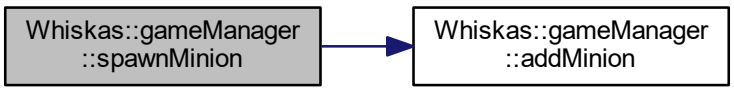
Parameters

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

Returns

true if succefull, false otherwise

Here is the call graph for this function:



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

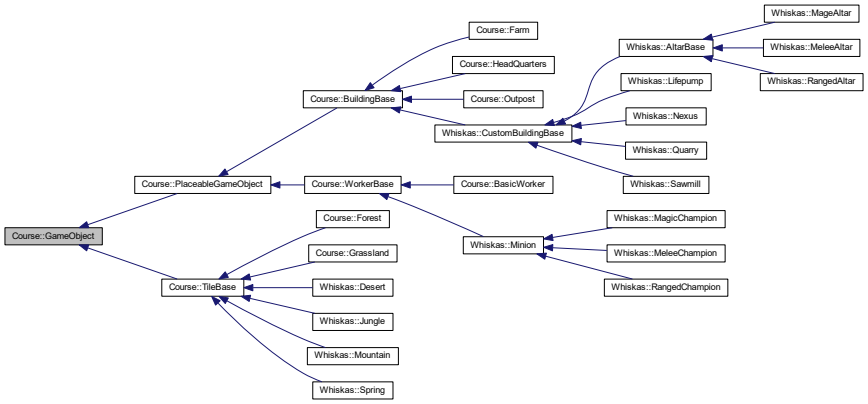
- E:/Gitin repo/ohjelmointi 3/whiskas/Game/handlerandmanager/gamemanager.h
- E:/Gitin repo/ohjelmointi 3/whiskas/Game/functions/functions.cpp
- E:/Gitin repo/ohjelmointi 3/whiskas/Game/handlerandmanager/gamemanager.cpp
- E:/Gitin repo/ohjelmointi 3/whiskas/UnitTests/handlerTest/tst_testhandler.cpp
-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

6.19 Course::GameObject Class Reference

The GameObject class is base-class that contain's 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

Returns the content with specified key from the descriptions.

- virtual void removeDescription (const std::string &key) final

  Removes 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

original	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

eventhandler	a shared pointer to Game's GameEventHandler.
objectmanager	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

owner	a shared pointer to player that "owns" the object.
eventhandler	a shared pointer to Game's GameEventHandler.
objectmanager	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

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

coordinate	a shared pointer to coordinates where the object is located.
eventhandler	a shared pointer to Game's GameEventHandler.
objectmanager	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

key	Key for the content
content	Content being stored

Postcondition

Exception guarantee: Strong

Exceptions

KeyError	- 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.

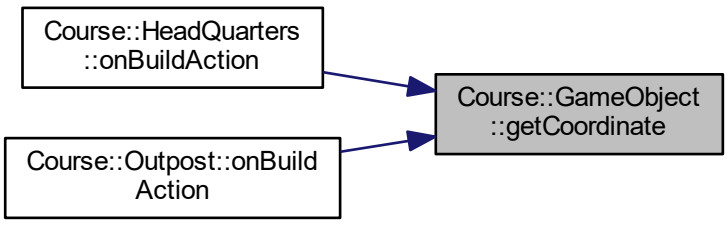
Postcondition

Exception guaranee: Strong

Exceptions

InvalidPointer	- 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

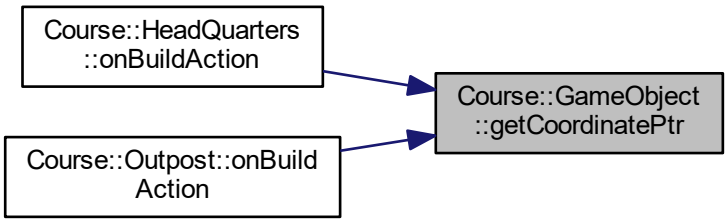
Exceptions

See	std::make_shared

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

key	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.

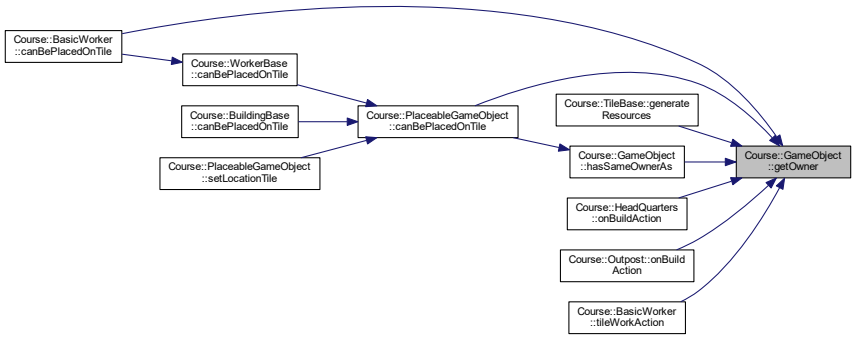
Returns

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

other	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

other	The other GameObject

Returns

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

Postcondition

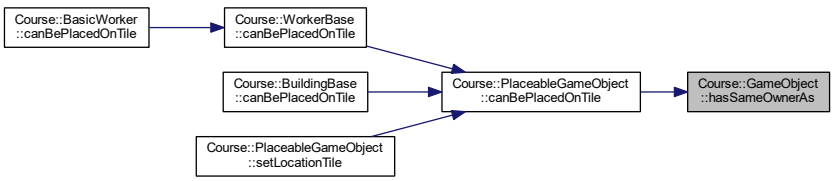
Excepetion guarantee: Strong

Exceptions

ExpiredPointer	- 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.

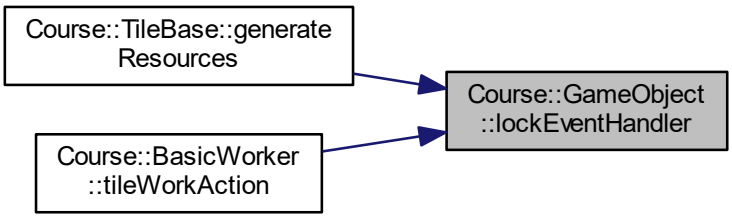
Returns

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.

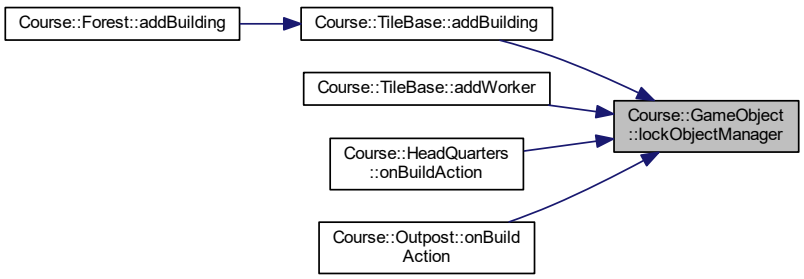
Returns

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

key	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

coordinate	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

coordinate	A shared-pointer to the new coordinate.

Postcondition

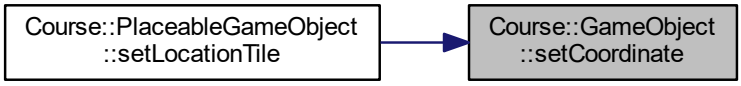
Exception guarantee: No-throw

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

key	Key for the content
content	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

desciptions	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

owner	a shared pointer to the new "owner".

Postcondition

Exception guarantee: No-throw

6.19.3.19 unsetCoordinate()

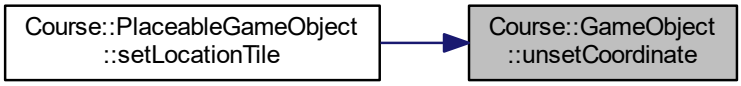
```
void Course::GameObject::unsetCoordinate ( )  [final], [virtual]
```

Removes the coordinate from GameObject.

Postcondition

Exception guarantee: No-throw

Here is the caller graph for this function:

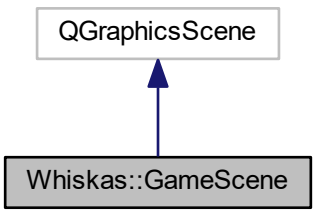


The documentation for this class was generated from the following files:

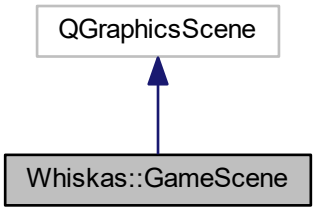
- E:/Gitin repo/ohjelmointi 3/whiskas/Course/CourseLib/core/gameobject.h
- E:/Gitin repo/ohjelmointi 3/whiskas/Course/CourseLib/core/gameobject.cpp

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()

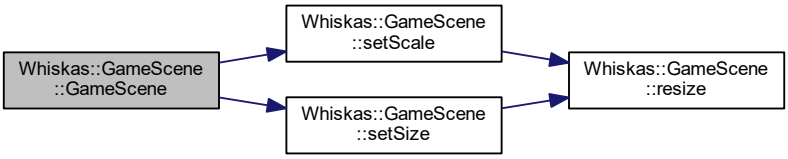
```
Whiskas::GameScene::GameScene (
            QWidget * qt_parent = nullptr,
            int width = 10,
            int height = 10,
            int scale = 50 )
```

Constructor.

Parameters

qt_parent	points to the parent object per Qt's parent-child-system.
width	in tiles for the game map.
height	in tiles for the game map.
scale	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

obj	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

event	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()

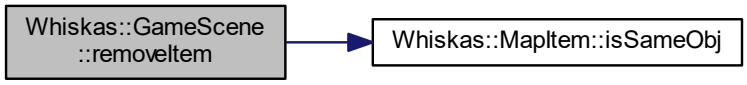
```
void Whiskas::GameScene::removeItem (
            std::shared_ptr< Course::GameObject > obj )
```

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

Parameters

obj	shared ptr to the object being deleted

Here is the call graph for this function:



6.20.2.6 setScale()

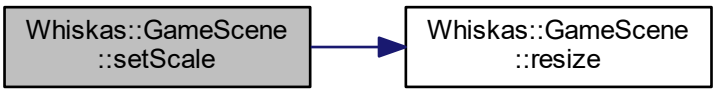
```
void Whiskas::GameScene::setScale (
            int scale )
```

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

width	(number of tiles)
height	(number of tiles)

Here is the call graph for this function:

Here is the caller graph for this function:

6.20.2.8 updateItem()

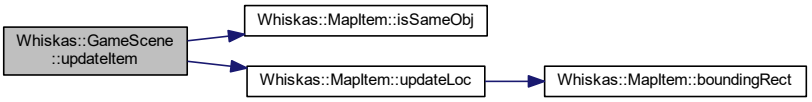
```
void Whiskas::GameScene::updateItem (
            std::shared_ptr< Course::GameObject > obj )
```

updates the position of obj.

Parameters

obj	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:

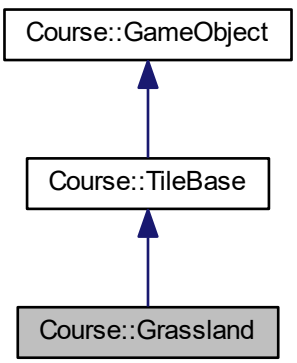
- E:/Gitin repo/ohjelmointi 3/whiskas/Game/graphics/gamescene.h
- E:/Gitin repo/ohjelmointi 3/whiskas/Game/graphics/gamescene.cpp

6.21 Course::Grassland Class Reference

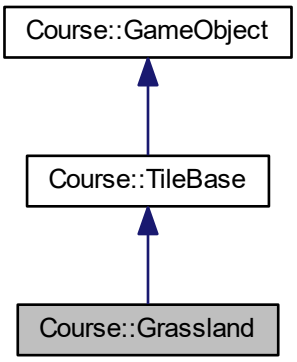
The Grassland class represents Grassland in the gameworld.

```
#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

location	is the Coordinate where the Tile is located in the game.
eventhandler	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:

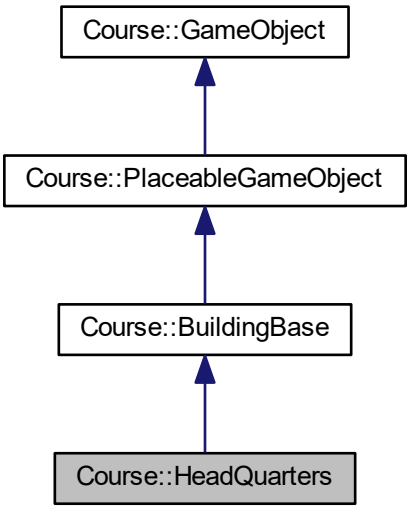
- E:/Gitin repo/ohjelmointi 3/whiskas/Course/CourseLib/tiles/grassland.h
- E:/Gitin repo/ohjelmointi 3/whiskas/Course/CourseLib/tiles/grassland.cpp

6.22 Course::HeadQuarters Class Reference

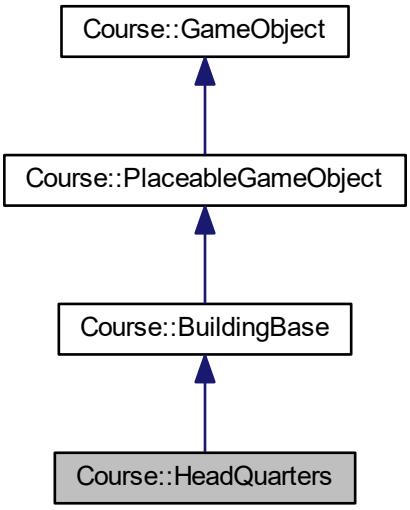
The HeadQuarters class represents a player's HeadQuarters-building.

```
#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

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

Postcondition

Exception Guarantee: No guarantee.

Exceptions

OwnerConflict	- 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]
```

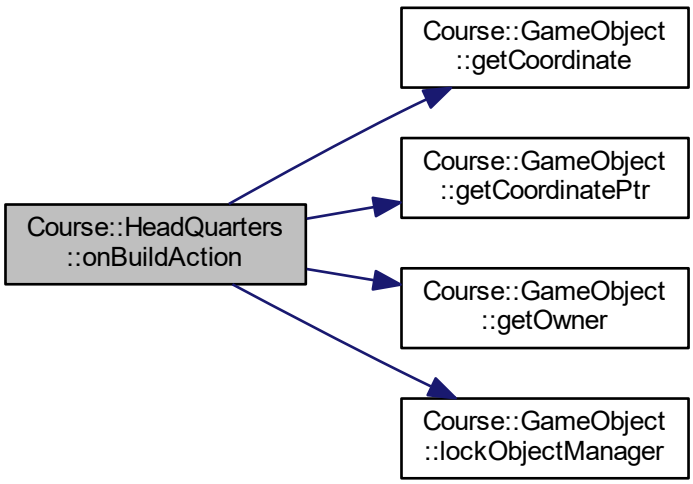
Sets neighbouring Tiles' ownership to this building's ownership in 3 tile-radius, if the Tiles don't already have an owner.

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:

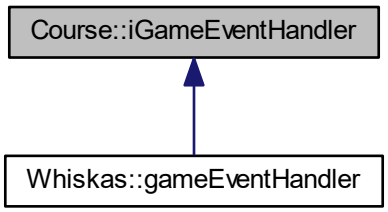
- E:/Gitin repo/ohjelmointi 3/whiskas/Course/CourseLib/buildings/headquarters.h
- E:/Gitin repo/ohjelmointi 3/whiskas/Course/CourseLib/buildings/headquarters.cpp

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

player	Pointer to the Player whose resource is being modified.
resource	Defines the modified resource.
amount	Defines the amount of change.

Postcondition

Exception guarantee: Basic

Returns

True - Modification was succesful.
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

player	Pointer to the Player whose resources are being modified.
resources	ResourceMap containing change amounts.

Returns

True - Modification was succesful.
False - Modification failed.

Implemented in Whiskas::gameEventHandler.

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

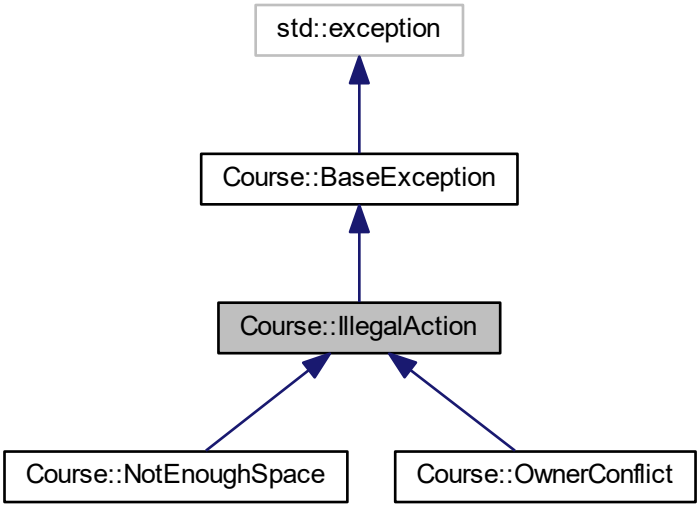
- E:/Gitin repo/ohjelmointi 3/whiskas/Course/CourseLib/interfaces/igameeventhandler.h

6.24 Course::IllegalAction Class Reference

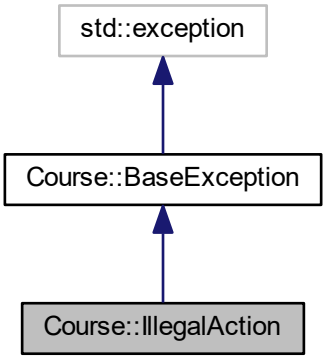
The IllegalAction exception is usually used in cases, where an illegal game action was attempted.

`#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

msg	std::string 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:

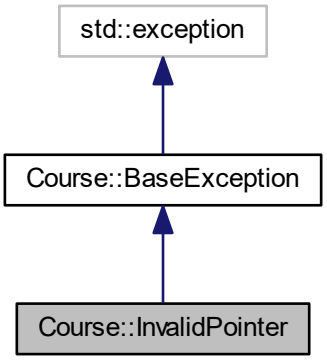
- E:/Gitin repo/ohjelmointi 3/whiskas/Course/CourseLib/exceptions/illegalaction.h

6.25 Course::InvalidPointer Class Reference

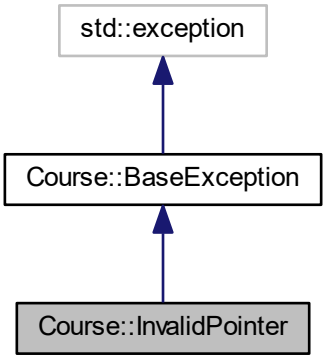
The InvalidPointer exception is usually used in cases, where data can't be accessed through a pointer.

```
#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

msg	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:

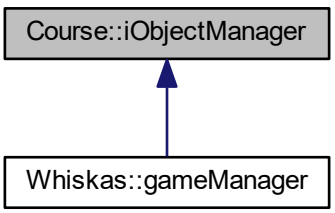
- E:/Gitin repo/ohjelmointi 3/whiskas/Course/CourseLib/exceptions/invalidpointer.h

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

tiles	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

coordinate	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

id	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

coordinates	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:

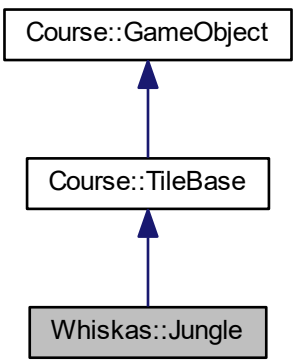
- E:/Gitin repo/ohjelmointi 3/whiskas/Course/CourseLib/interfaces/iobjectmanager.h

6.27 Whiskas::Jungle Class Reference

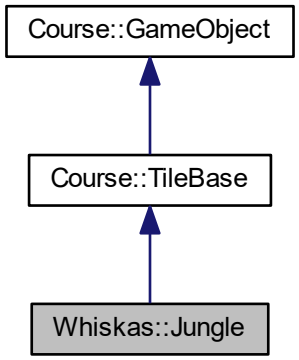
The Jungle class represents the jungle in the gameworld. The jungle supports ranged altars and sawmills.

`#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:

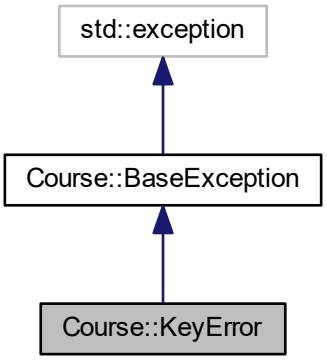
- E:/Gitin repo/ohjelmointi 3/whiskas/Game/tiles/jungle.h
- E:/Gitin repo/ohjelmointi 3/whiskas/Game/tiles/jungle.cpp

6.28 Course::KeyError Class Reference

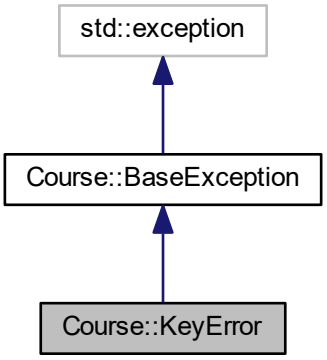
The KeyError class is an Exception-class for cases where the used key is invalid.

`#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

msg	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:

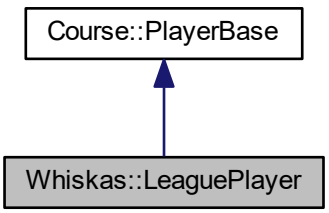
- E:/Gitin repo/ohjelmointi 3/whiskas/Course/CourseLib/exceptions/keyerror.h

6.29 Whiskas::LeaguePlayer Class Reference

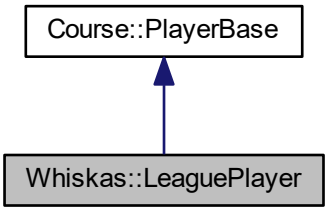
The LeaguePlayer class The playerClass for this game.

`#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

name	
playerItems	Resource the player has at the start of the game

6.29.3 Member Function Documentation

6.29.3.1 setPlayerItems()

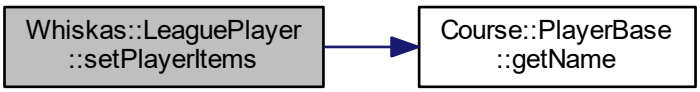
```
void Whiskas::LeaguePlayer::setPlayerItems (
            AdvancedResourceMap toSet )
```

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

Parameters

toSet	New items

Here is the call graph for this function:



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

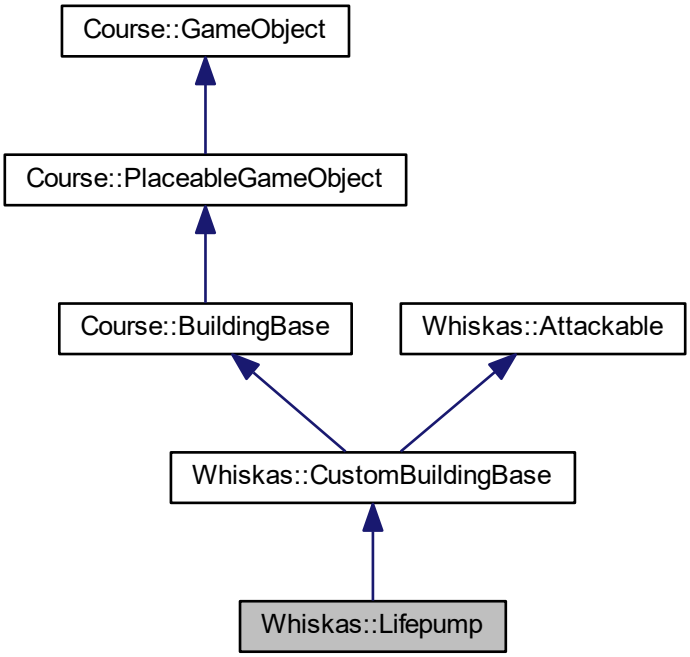
- E:/Gitin repo/ohjelmointi 3/whiskas/Game/leagueplayer.h
- E:/Gitin repo/ohjelmointi 3/whiskas/Game/leagueplayer.cpp

6.30 Whiskas::Lifepump Class Reference

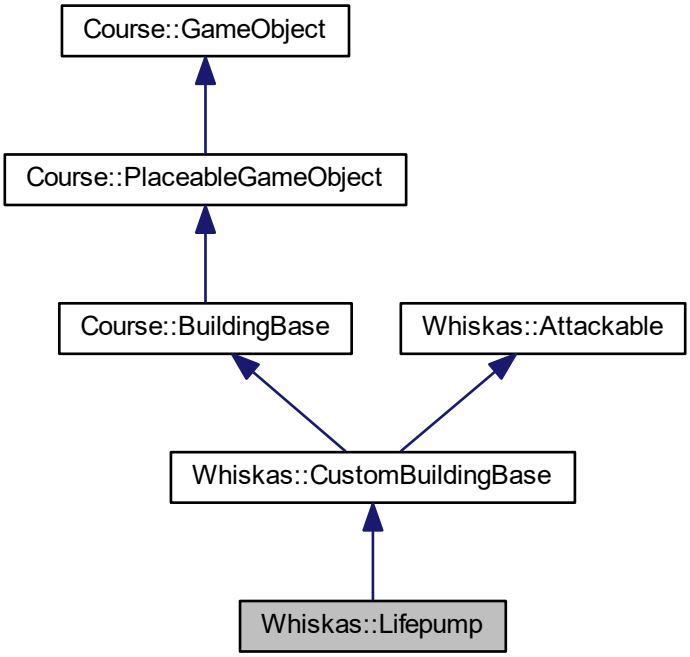
The Lifepump class represents a lifepump-building in the game.

```
#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:

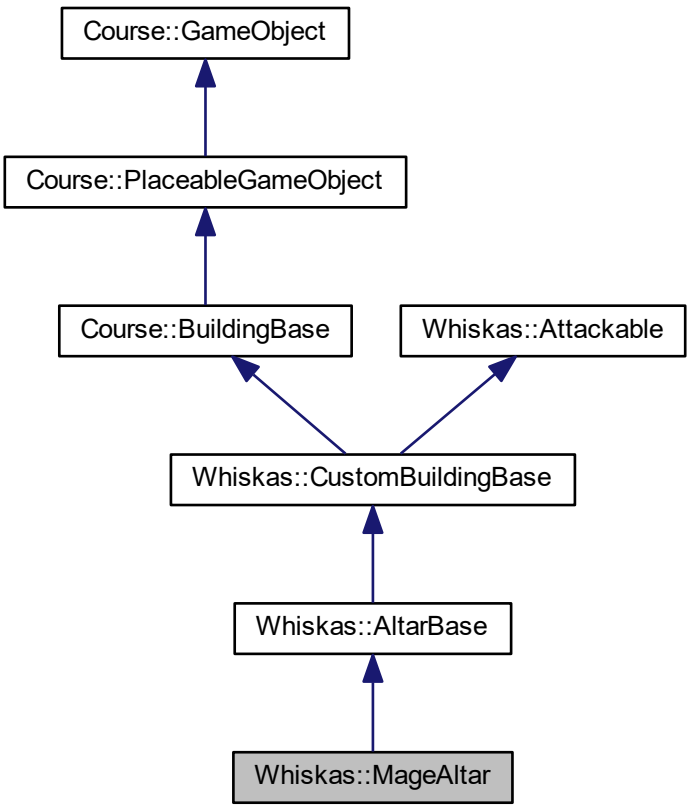
- E:/Gitin repo/ohjelmointi 3/whiskas/Game/buildings/lifepump.h
- E:/Gitin repo/ohjelmointi 3/whiskas/Game/buildings/lifepump.cpp

6.31 Whiskas::MageAltar Class Reference

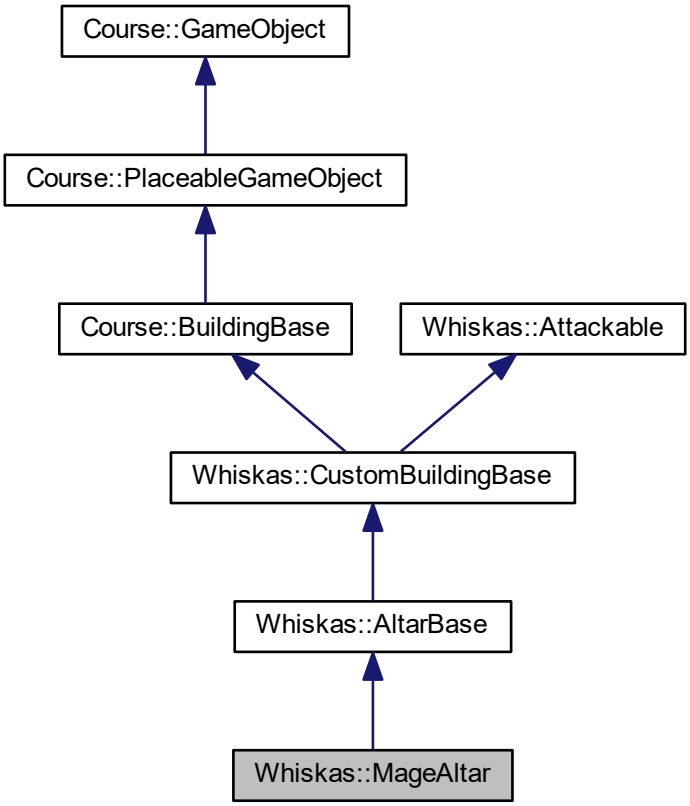
The MageAltar class is an altar, that upgrades minions to mages.

```
#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:

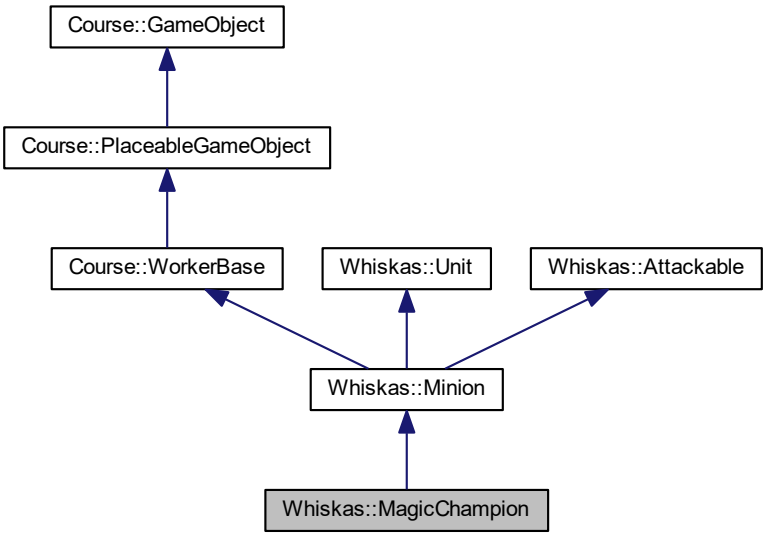
- E:/Gitin repo/ohjelmointi 3/whiskas/Game/buildings/magealtar.h
- E:/Gitin repo/ohjelmointi 3/whiskas/Game/buildings/magealtar.cpp

6.32 Whiskas::MagicChampion Class Reference

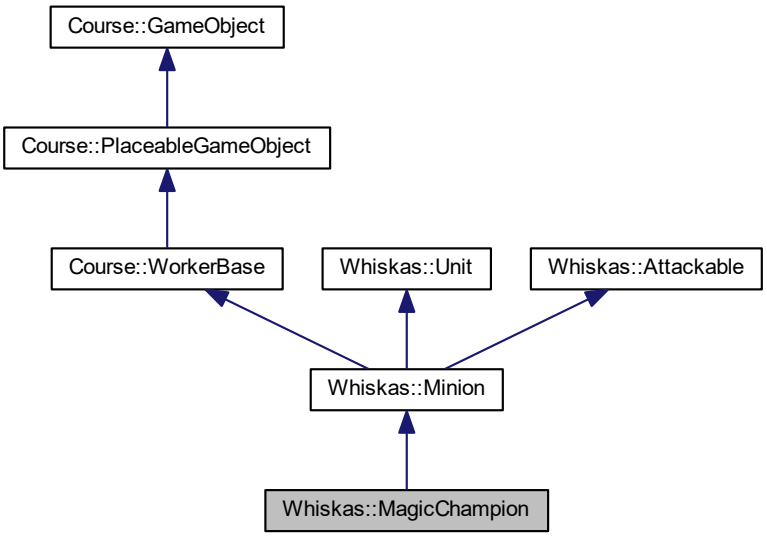
The MagicChampion class Subminion, has a higher dmg and an area attack.

```
#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 numberOfAttacks=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]
```

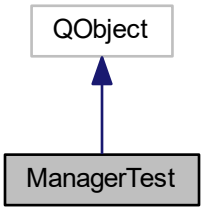
Reimplemented from Course::WorkerBase.

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

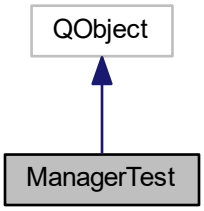
- E:/Gitin repo/ohjelmointi 3/whiskas/Game/minion/magicchampion.h
- E:/Gitin repo/ohjelmointi 3/whiskas/Game/minion/magicchampion.cpp

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:

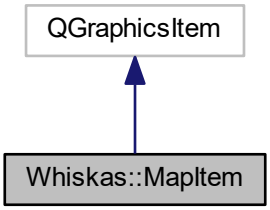
- E:/Gitin repo/ohjelmointi 3/whiskas/UnitTests/unittestfolder/managertest.h
- E:/Gitin repo/ohjelmointi 3/whiskas/UnitTests/unittestfolder/managertest.cpp

6.34 Whiskas::MapItem Class Reference

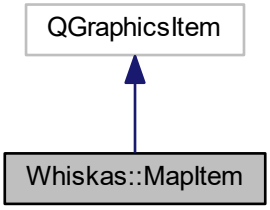
The MapItem class is a custom QGraphicsItem that acts as a single GameObject's graphical element.

`#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

obj	shared_ptr to the obj.
size	of the created item in pixels.

Precondition

obj must have a valid Coordinate.

6.34.3 Member Function Documentation

6.34.3.1 boundingRect()

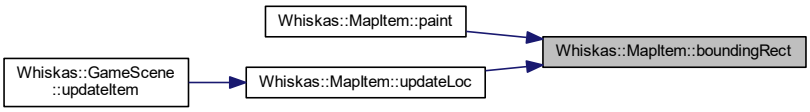
```
QRectF Whiskas::MapItem::boundingRect ( ) const  [override]
```

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.

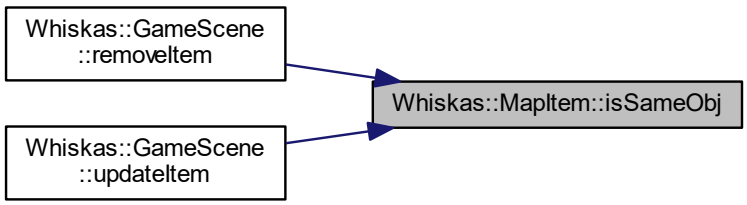
Parameters

obj	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

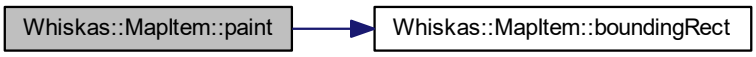
Parameters

painter	
option	
widget	

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

size	of the object in pixels.

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

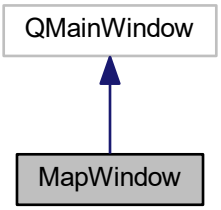
- E:/Gitin repo/ohjelmointi 3/whiskas/Game/graphics/mapitem.h
- E:/Gitin repo/ohjelmointi 3/whiskas/Game/graphics/mapitem.cpp

6.35 MapWindow Class Reference

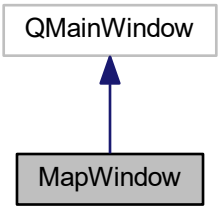
The MapWindow class Manages the UI interface.

```
#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

obj	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

x	
y	

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

event	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

manager	ObjectManager that recieves the building
handler	buildings handler
owner	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

nHandler	the new game handler

6.35.2.6 setGManager()

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

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

setGManager sets the curretn game manager

Parameters

manager	the new game manager

6.35.2.7 setSize()

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

setSize sets the game size

Parameters

width	in tiles
height	int tiles

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

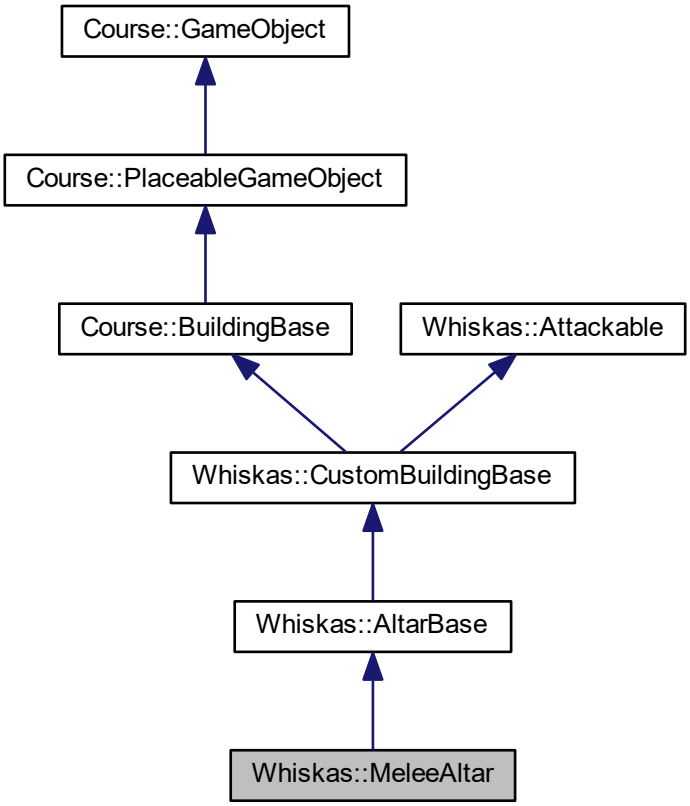
- E:/Gitin repo/ohjelmointi 3/whiskas/Game/mapwindow.hh
- E:/Gitin repo/ohjelmointi 3/whiskas/Game/mapwindow.cc

6.36 Whiskas::MeleeAltar Class Reference

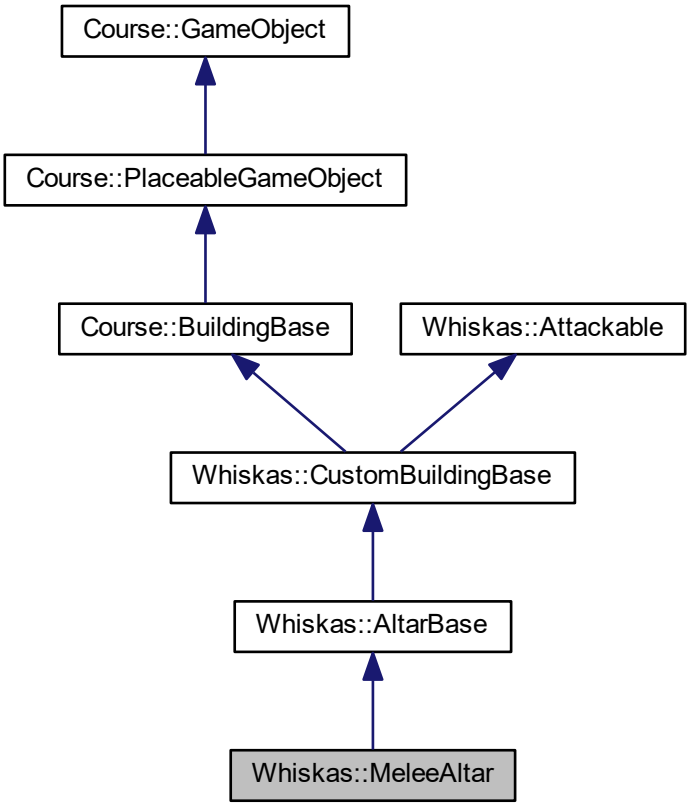
The MeleeAltar class is an altar that upgrades minions to melee champs.

```
#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:

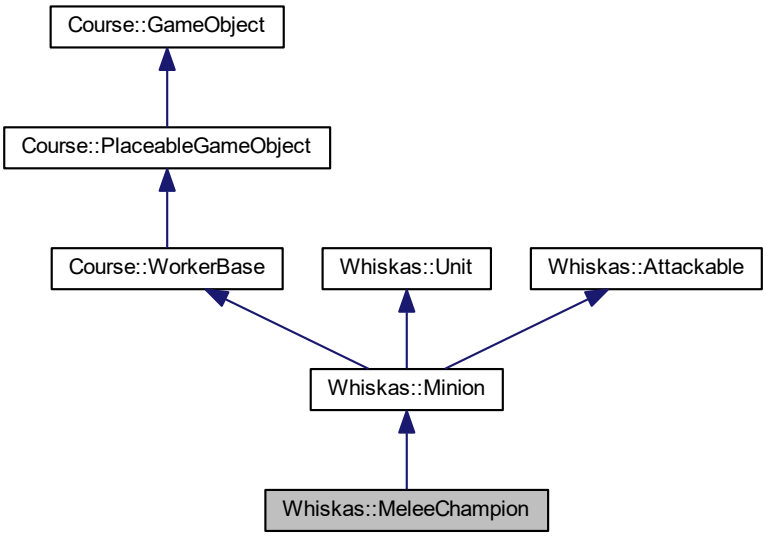
- E:/Gitin repo/ohjelmointi 3/whiskas/Game/buildings/meleealtar.h
- E:/Gitin repo/ohjelmointi 3/whiskas/Game/buildings/meleealtar.cpp

6.37 Whiskas::MeleeChampion Class Reference

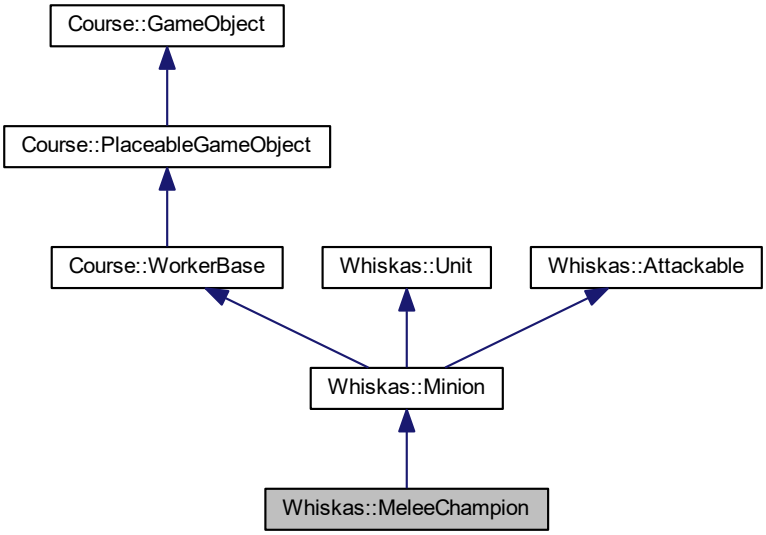
The MeleeChampion class is a minion subtype. Has a melee attack of higher dmg and better defenses.

```
#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

hModifier	

Returns

true if dead, false otherwise

Reimplemented from Whiskas::Attackable.

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

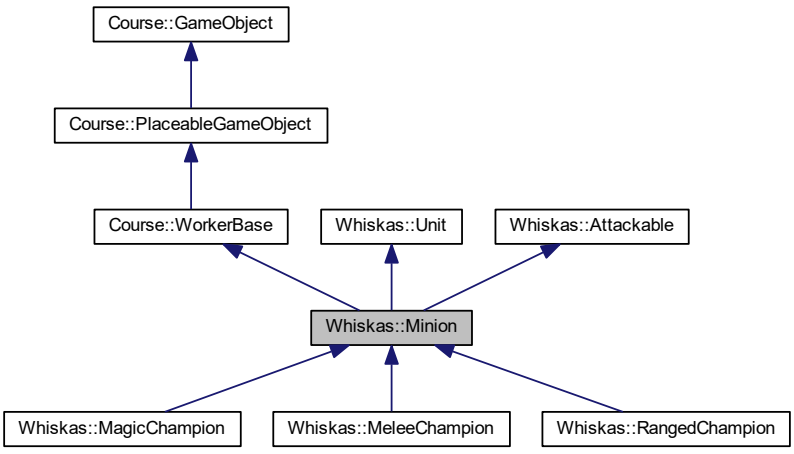
- E:/Gitin repo/ohjelmointi 3/whiskas/Game/minion/meleechampion.h
- E:/Gitin repo/ohjelmointi 3/whiskas/Game/minion/meleechampion.cpp

6.38 Whiskas::Minion Class Reference

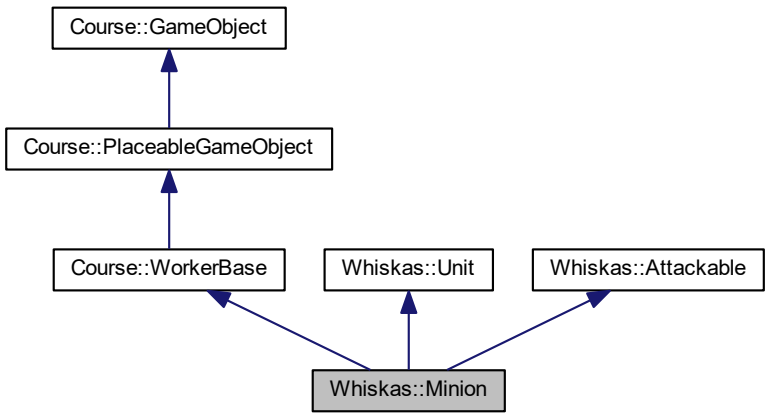
The Minion class is a subtype of worker It can be upgraded to champions using altars Is attackable and a unit (so it can attack and move)

```
#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

hModifier	

Returns

true if dead, false otherwise

Reimplemented from Whiskas::Attackable.

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

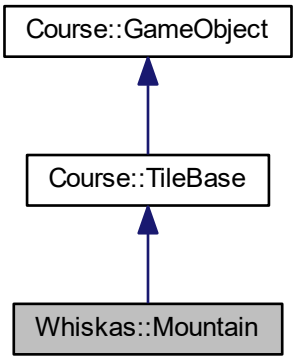
- E:/Gitin repo/ohjelmointi 3/whiskas/Game/minion/minion.h
- E:/Gitin repo/ohjelmointi 3/whiskas/Game/minion/minion.cpp

6.39 Whiskas::Mountain Class Reference

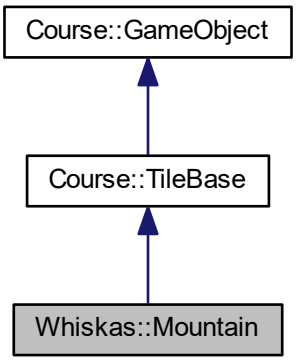
The Mountain class represents the mountain in the gameworld. The mountain supports melee altars and quarries.

```
#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

building	A pointer to the Building that is being added.

Postcondition

Exception guarantee: Basic

Exceptions

InvalidPointer	- If the building's pointer doesn't point to anything or ObjectManager doesn't return valid shared_ptr to this tile.
IllegalMove	- Any IllegalException can be thrown by a Tile or Building if it breaks a placement rule.
NotEnoughSpace	(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:

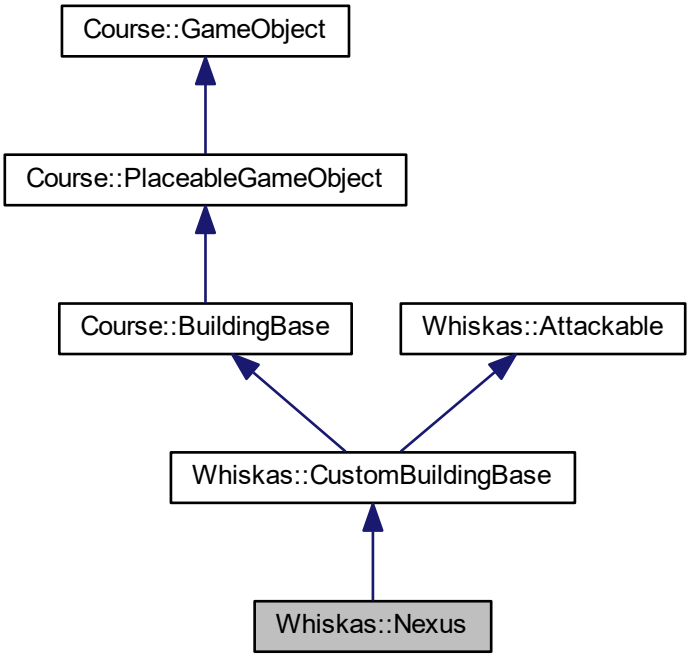
- E:/Gitin repo/ohjelmointi 3/whiskas/Game/tiles/mountain.h
- E:/Gitin repo/ohjelmointi 3/whiskas/Game/tiles/mountain.cpp

6.40 Whiskas::Nexus Class Reference

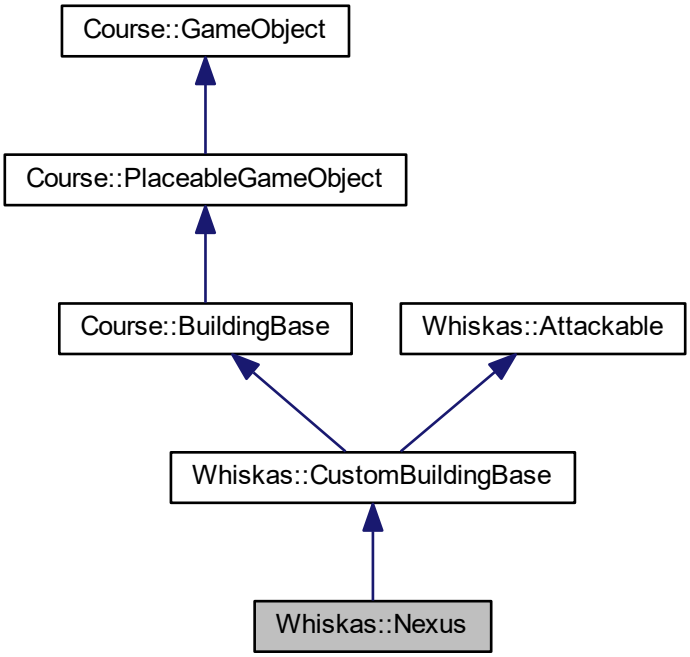
The Nexus class represents a Nexus-building in the game.

`#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:

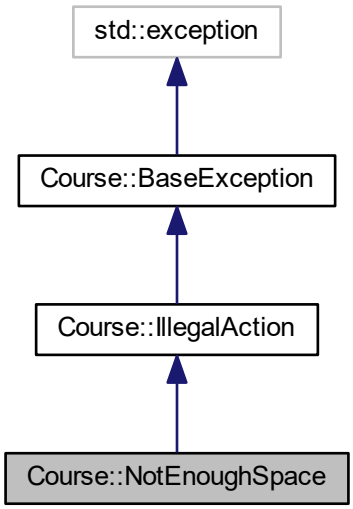
- E:/Gitin repo/ohjelmointi 3/whiskas/Game/buildings/nexus.h
- E:/Gitin repo/ohjelmointi 3/whiskas/Game/buildings/nexus.cpp

6.41 Course::NotEnoughSpace Class Reference

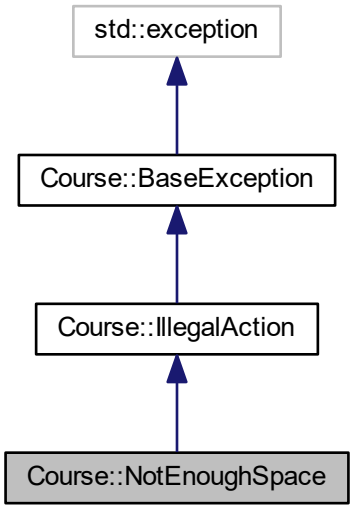
The NotEnoughSpace class is an Exception-class for errors where GameObjects are being placed onto Tiles with no space available.

```
#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

msg	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:

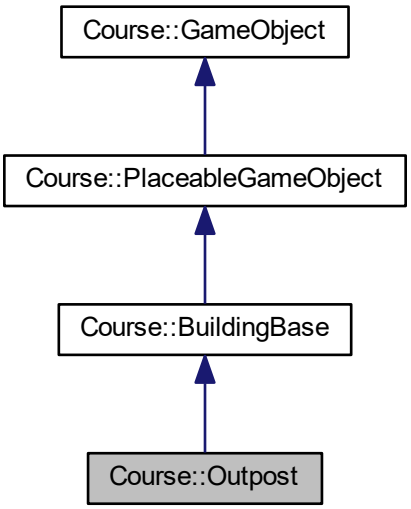
- E:/Gitin repo/ohjelmointi 3/whiskas/Course/CourseLib/exceptions/notenoughspace.h

6.42 Course::Outpost Class Reference

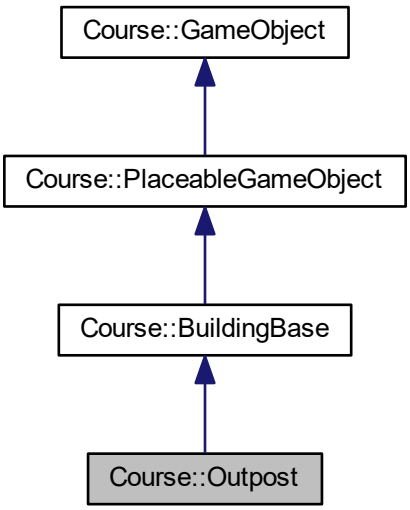
The Outpost class represents a player's Outpost-building.

```
#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=Const↩ ResourceMaps::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

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

Postcondition

Exception Guarantee: No guarantee.

Exceptions

OwnerConflict	- 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]
```

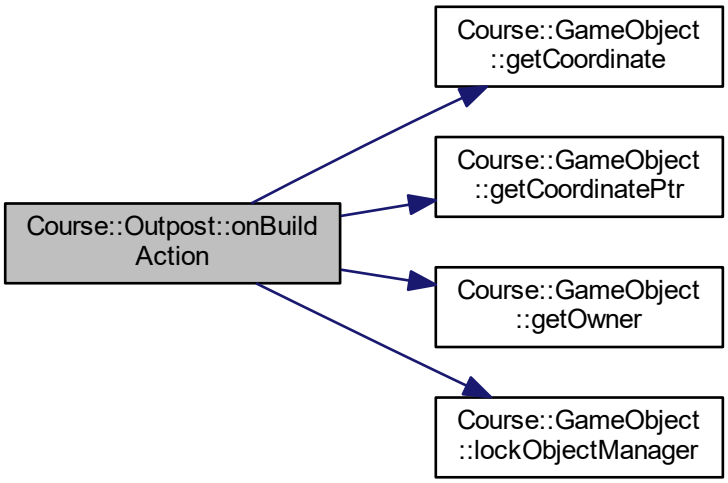
Sets neighbouring Tiles' ownership to this building's ownership in 1 tile-radius, if the Tiles don't already have an owner.

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:

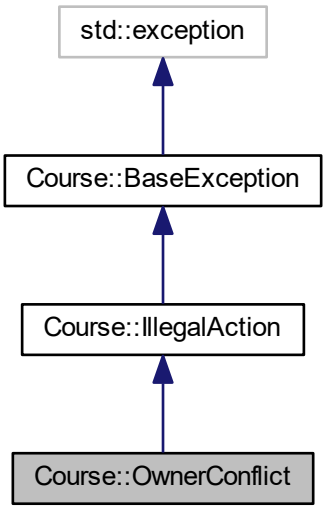
- E:/Gitin repo/ohjelmointi 3/whiskas/Course/CourseLib/buildings/outpost.h
- E:/Gitin repo/ohjelmointi 3/whiskas/Course/CourseLib/buildings/outpost.cpp

6.43 Course::OwnerConflict Class Reference

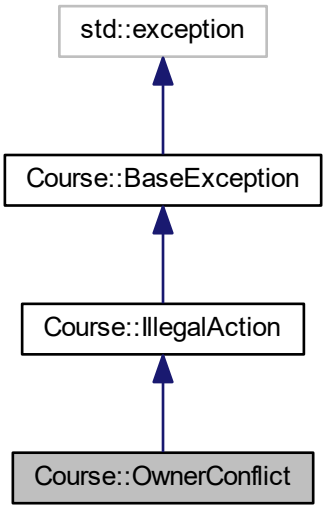
The OwnerConflict class is an Exception-class for errors where an operation is conflicting with a GameObject's ownership.

```
#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

msg	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:

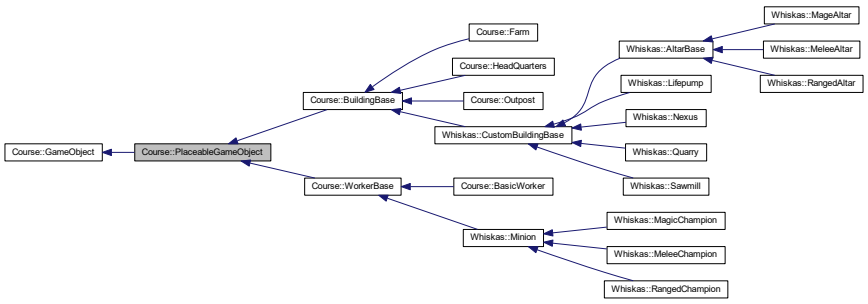
- E:/Gitin repo/ohjelmointi 3/whiskas/Course/CourseLib/exceptions/ownerconflict.h

6.44 Course::PlaceableGameObject Class Reference

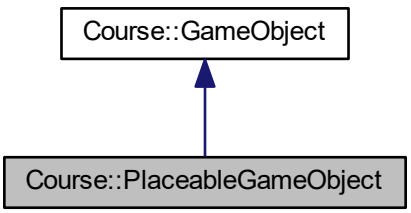
The PlaceableGameObject class represents GameObjects that can be placed on Tile Objects.

```
#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

eventhandler	points to the student's GameEventHandler.
objectmanager	points to the studtent's ObjectManager
owner	points to the owning player.
tilespace	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

target	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

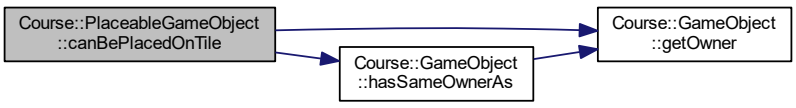
Exception guarantee: Basic

Note

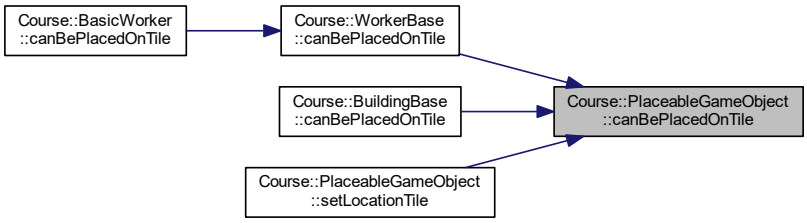
Override to change default behaviour

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]
```

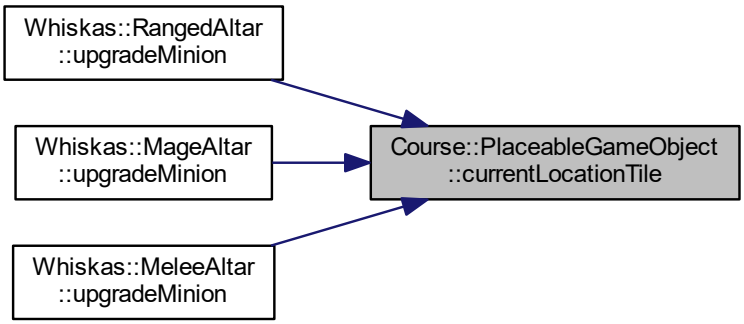
Returns a shared_ptr to current location-tile.

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

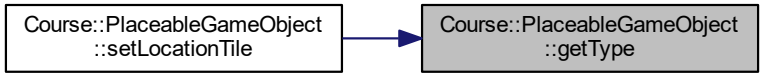
Note

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

tile	points to the Tile where the object is placed.

Postcondition

Exception guarantee: Strong

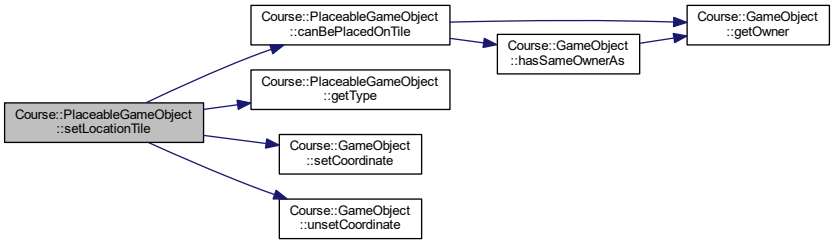
Note

nullptr can be used to clear the location.

Exceptions

IllegalAction	- If canBePlacedOnTile 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.

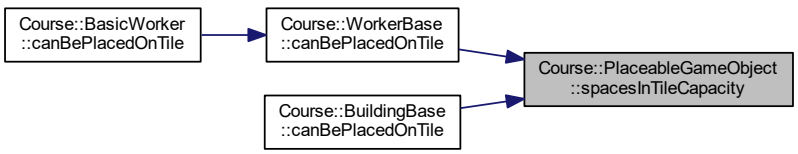
Returns

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:

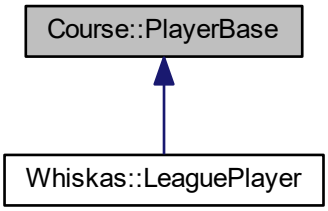
- E:/Gitin repo/ohjelmointi 3/whiskas/Course/CourseLib/core/placeablegameobject.h
- E:/Gitin repo/ohjelmointi 3/whiskas/Course/CourseLib/core/placeablegameobject.cpp

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

name	A std::string for player's name
objects	(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

object	Is a weak pointer to the stored GameObject

Postcondition

Exception guarantee: Strong

Exceptions

See	std::vector::push_back()

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

objects	Is an std::vector of weak GameObject-pointers.

Postcondition

Exception guarantee: Strong

Exceptions

See	std::vector::insert()

6.45.3.3 getName()

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

Returns the Player-object's name.

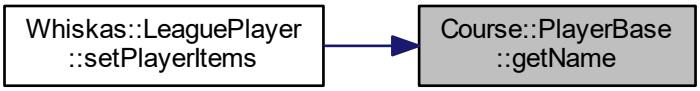
Returns

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

id	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

object	a weak pointer to GameObject

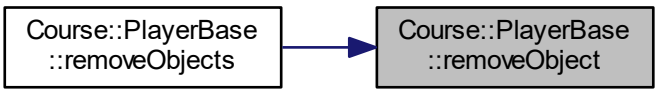
Postcondition

Exception guarantee: Basic

Exceptions

ExpiredPointer	- object is expired
KeyError	- 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

objects	A vector of ObjectId (unsigned int) for GameObjects that are removed

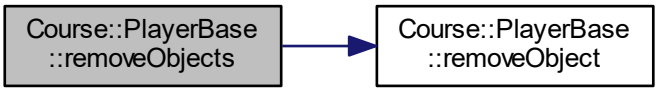
Postcondition

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

objects	A vector of weak GameObject-pointers

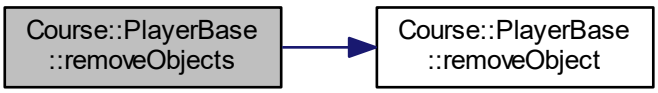
Postcondition

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

new_name	The new name.

Postcondition

Exception guarantee: No-throw

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

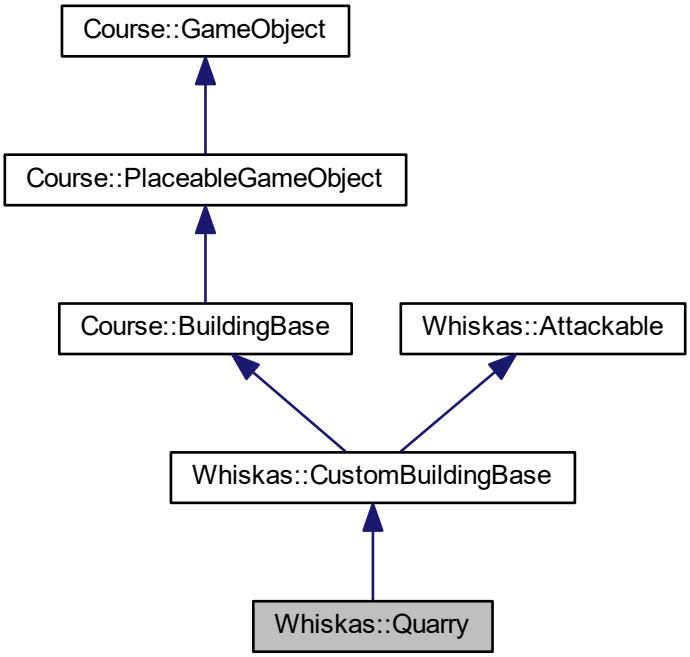
- E:/Gitin repo/ohjelmointi 3/whiskas/Course/CourseLib/core/playerbase.h
- E:/Gitin repo/ohjelmointi 3/whiskas/Course/CourseLib/core/playerbase.cpp

6.46 Whiskas::Quarry Class Reference

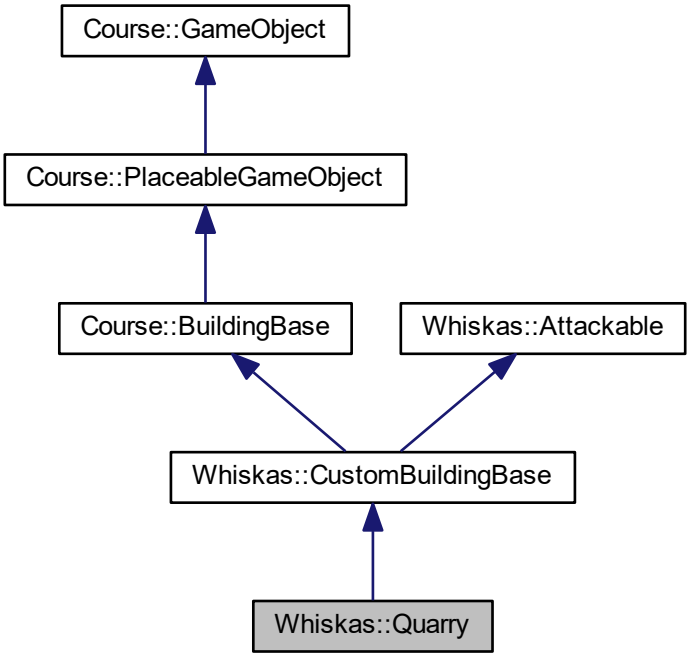
The Quarry class represents a quarry-building in the game.

```
#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=QUAR↩ RY_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:

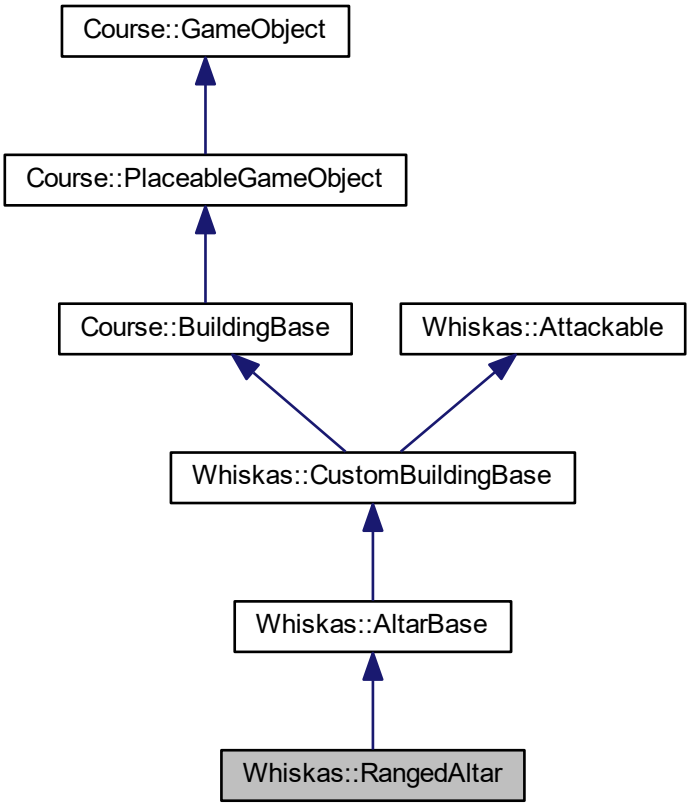
- E:/Gitin repo/ohjelmointi 3/whiskas/Game/buildings/quarry.h
- E:/Gitin repo/ohjelmointi 3/whiskas/Game/buildings/quarry.cpp

6.47 Whiskas::RangedAltar Class Reference

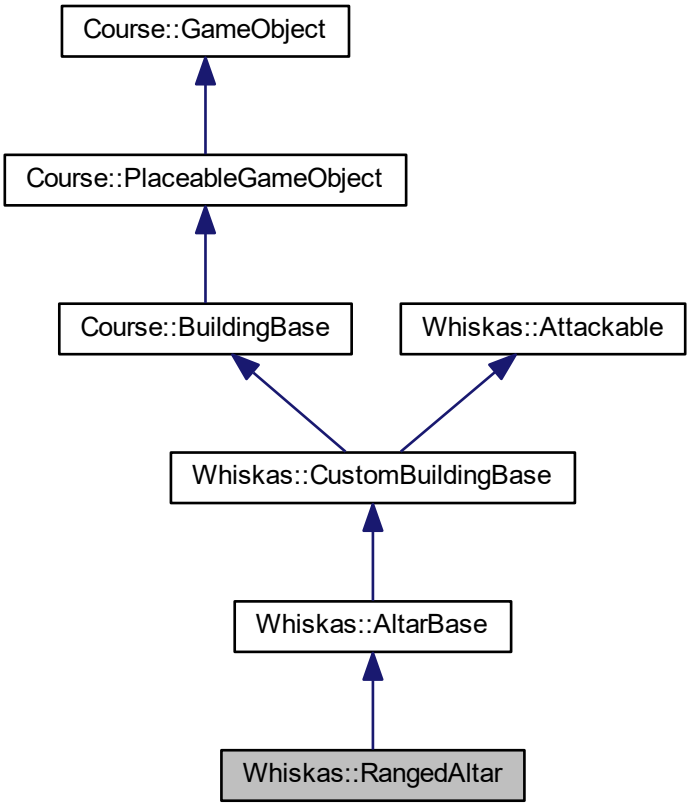
The RangedAltar class is an altar that upgrades minions to ranged units.

```
#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:

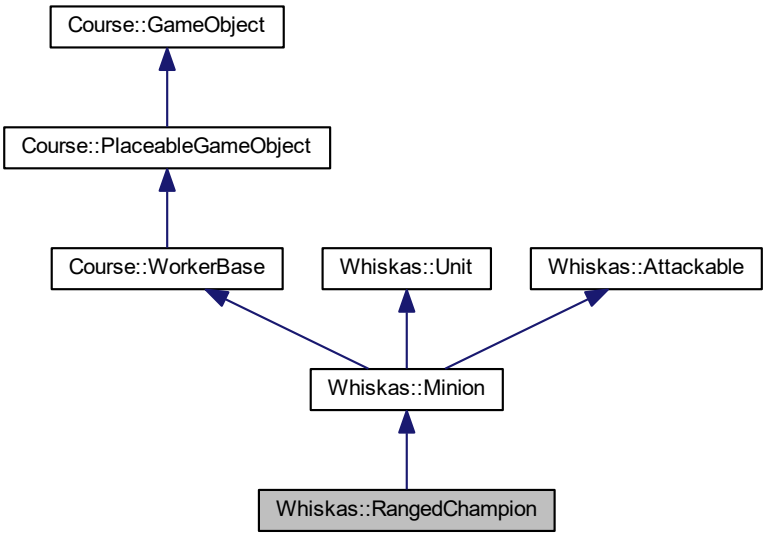
- E:/Gitin repo/ohjelmointi 3/whiskas/Game/buildings/rangedaltar.h
- E:/Gitin repo/ohjelmointi 3/whiskas/Game/buildings/rangedaltar.cpp

6.48 Whiskas::RangedChampion Class Reference

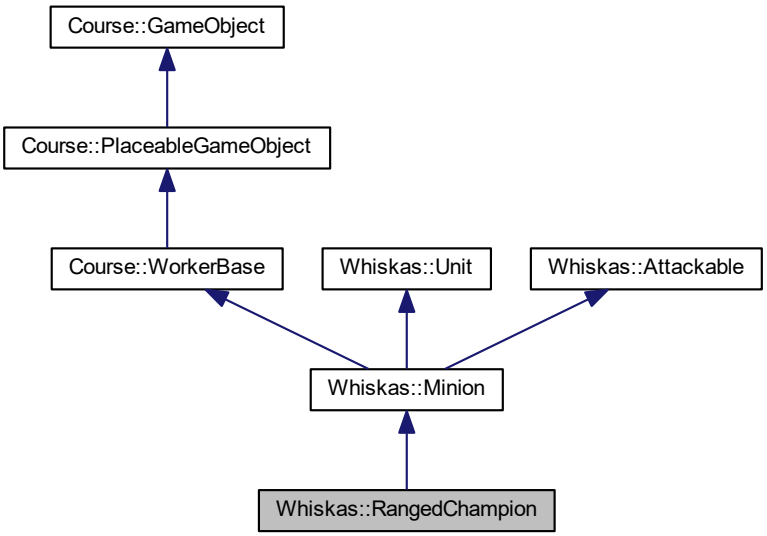
The RangedChampion class Has a ranged attack and higher movement.

```
#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 numberOfAttacks=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]
```

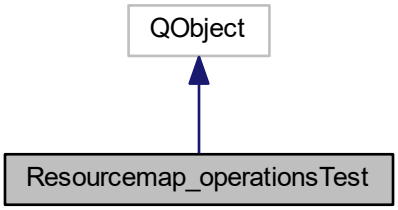
Reimplemented from Whiskas::Minion.

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

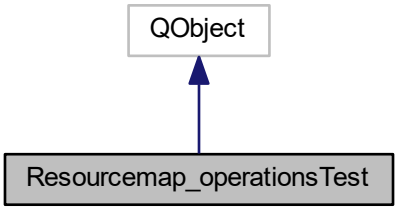
- E:/Gitin repo/ohjelmointi 3/whiskas/Game/minion/rangedchampion.h
- E:/Gitin repo/ohjelmointi 3/whiskas/Game/minion/rangedchampion.cpp

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:

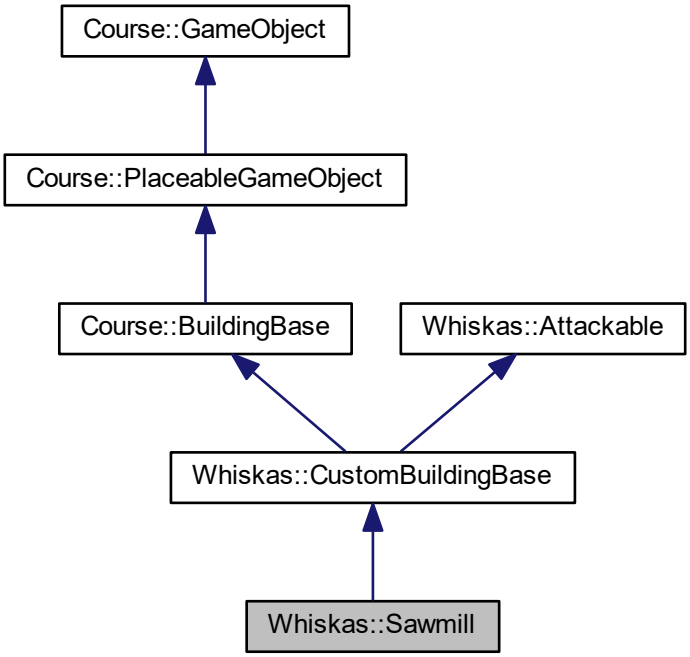
- E:/Gitin repo/ohjelmointi 3/whiskas/Course/UnitTests/resourcemap_operations/tst_resourcemap_operationstest.↩ cpp

6.50 Whiskas::Sawmill Class Reference

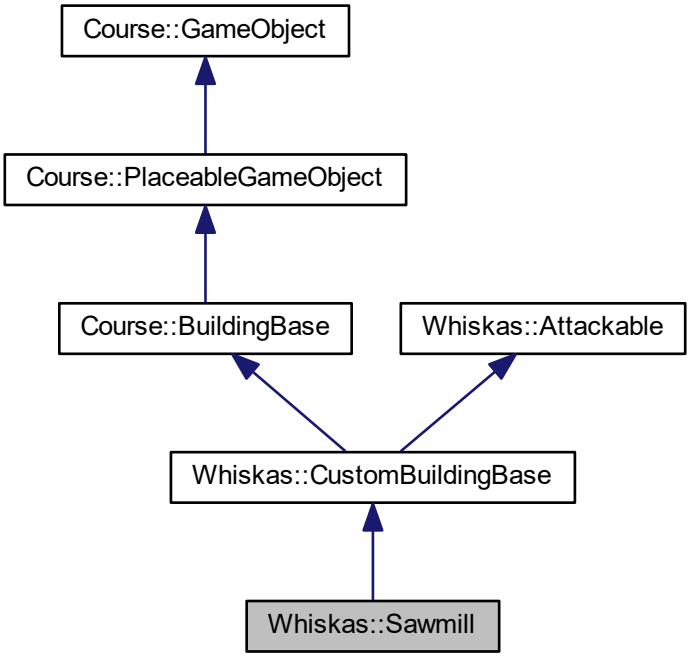
The Sawmill class represents a Sawmill-building in the game.

```
#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 AdvancedResource↩ Map &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:

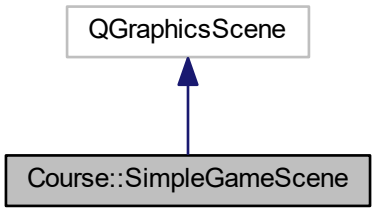
- E:/Gitin repo/ohjelmointi 3/whiskas/Game/buildings/sawmill.h
- E:/Gitin repo/ohjelmointi 3/whiskas/Game/buildings/sawmill.cpp

6.51 Course::SimpleGameScene Class Reference

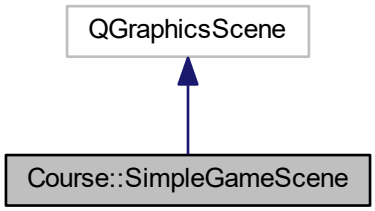
The SimpleGameScene is a custom QGraphicsScene that shows a simple rendering of the game map.

```
#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.

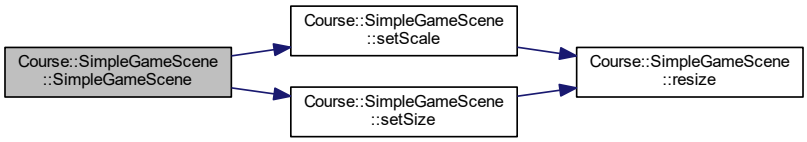
Parameters

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

Precondition

0 < width <= 100 && 0 < height <= 100 && 0 < scale <= 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

obj	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

event	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.

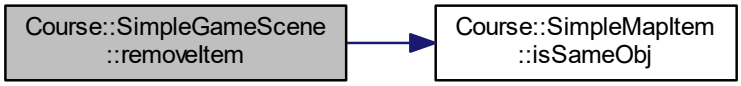
Parameters

obj	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

scale	in pixels.

Precondition

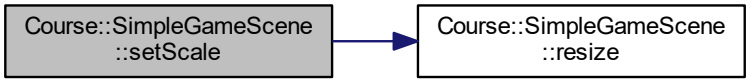
0 < scale <= 500

Postcondition

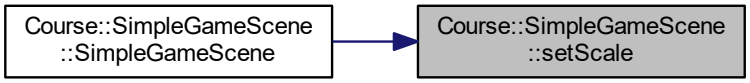
Scene scale is set to scale.

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

width	in tiles.
height	in tiles.

Precondition

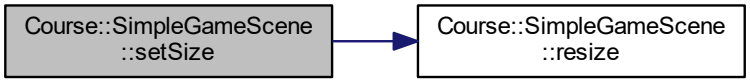
width and height must each be between 1 and 100.

Postcondition

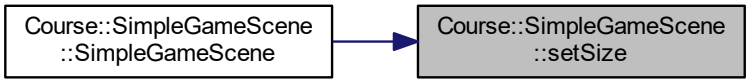
width and height are set to given sizes.

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()

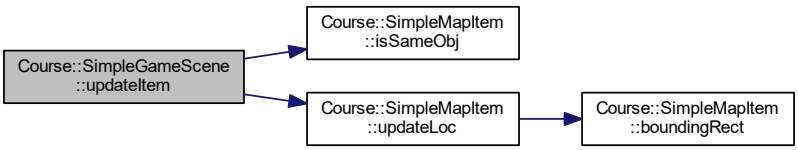
```
void Course::SimpleGameScene::updateItem (
            std::shared_ptr< Course::GameObject > obj )
```

updates the position of obj.

Parameters

obj	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:

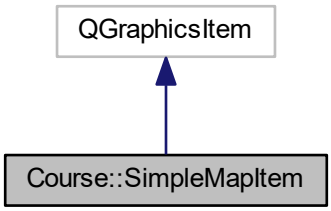
- E:/Gitin repo/ohjelmointi 3/whiskas/Course/CourseLib/graphics/simplegamescene.h
- E:/Gitin repo/ohjelmointi 3/whiskas/Course/CourseLib/graphics/simplegamescene.cpp

6.52 Course::SimpleMapItem Class Reference

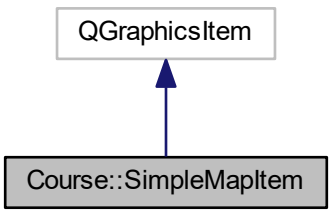
The SimpleMapItem class is a custom QGraphicsItem that acts as a single GameObject's graphical element.

`#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

obj	shared_ptr to the obj.
size	of the created item in pixels.

Precondition

obj must have a valid Coordinate.

6.52.3 Member Function Documentation

6.52.3.1 boundingRect()

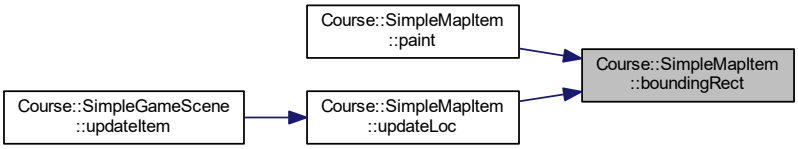
```
QRectF Course::SimpleMapItem::boundingRect ( ) const  [override]
```

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.

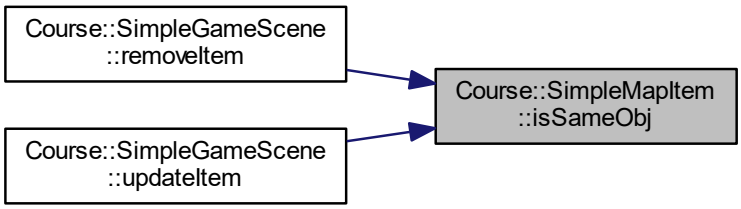
Parameters

obj	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

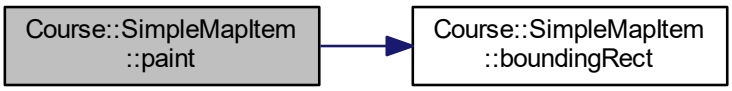
Parameters

painter	
option	
widget	

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

size	of the object in pixels.

Precondition

0 < size <= 500

Postcondition

Exception guarantee: No-throw

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

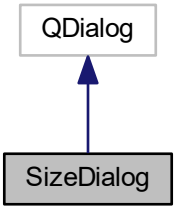
- E:/Gitin repo/ohjelmointi 3/whiskas/Course/CourseLib/graphics/simplemapitem.h
- E:/Gitin repo/ohjelmointi 3/whiskas/Course/CourseLib/graphics/simplemapitem.cpp

6.53 SizeDialog Class Reference

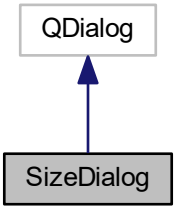
The SizeDialog class Sets the mapSize.

```
#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:

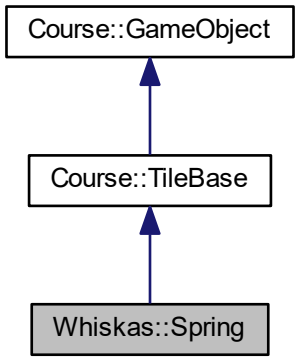
- E:/Gitin repo/ohjelmointi 3/whiskas/Game/dialogs/sizedialog.h
- E:/Gitin repo/ohjelmointi 3/whiskas/Game/dialogs/sizedialog.cpp

6.54 Whiskas::Spring Class Reference

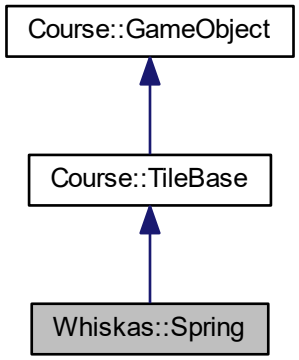
The Spring class represents the spring in the gameworld. The spring supports mage altars and lifepumps.

`#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::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
- 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

building	A pointer to the Building that is being added.

Postcondition

Exception guarantee: Basic

Exceptions

InvalidPointer	- If the building's pointer doesn't point to anything or ObjectManager doesn't return valid shared_ptr to this tile.
IllegalMove	- Any IllegalException can be thrown by a Tile or Building if it breaks a placement rule.
NotEnoughSpace	(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:

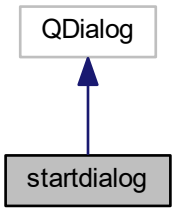
- E:/Gitin repo/ohjelmointi 3/whiskas/Game/tiles/spring.h
- E:/Gitin repo/ohjelmointi 3/whiskas/Game/tiles/spring.cpp

6.55 startdialog Class Reference

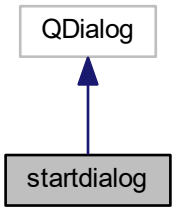
The startdialog class Selector dialog for the game start.

```
#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

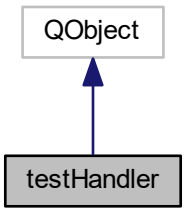
The startdialog class Selector dialog for the game start.

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

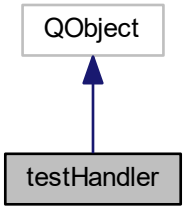
- E:/Gitin repo/ohjelmointi 3/whiskas/Game/dialogs/startdialog.h
- E:/Gitin repo/ohjelmointi 3/whiskas/Game/dialogs/startdialog.cpp

6.56 testHandler Class Reference

Inheritance diagram for testHandler:



Collaboration diagram for testHandler:

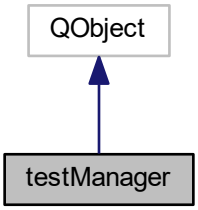


The documentation for this class was generated from the following file:

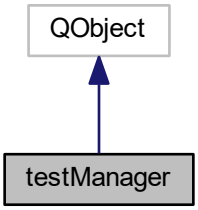
- E:/Gitin repo/ohjelmointi 3/whiskas/UnitTests/handlerTest/tst_testhandler.cpp

6.57 testManager Class Reference

Inheritance diagram for testManager:



Collaboration diagram for testManager:



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

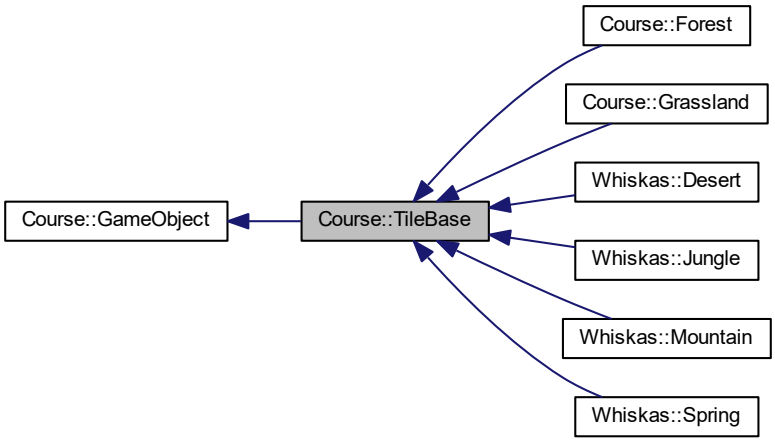
- E:/Gitin repo/ohjelmointi 3/whiskas/UnitTests/managerTest/tst_testmanager.cpp

6.58 Course::TileBase Class Reference

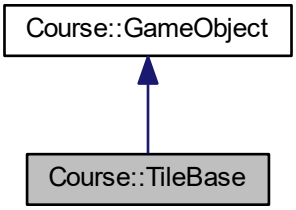
The TileBase class is a base-class for different Tile-objects in the game.
.

```
#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

building	A pointer to the Building that is being added.

Postcondition

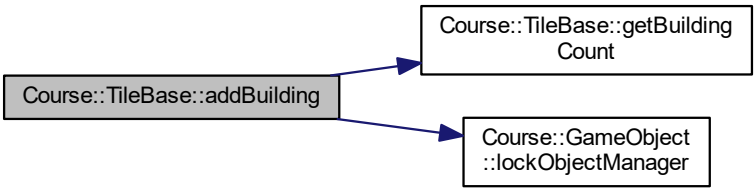
Exception guarantee: Basic

Exceptions

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

worker	A pointer to the Worker-object that is being added.

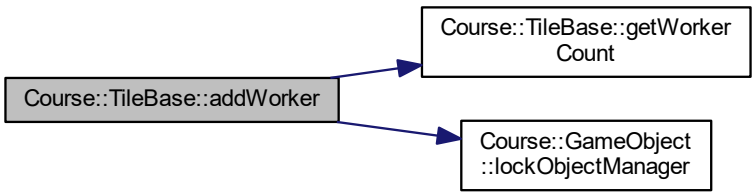
Postcondition

Exception guarantee: Basic

Exceptions

InvalidPointer	- If the Worker's pointer doesn't point to anything or ObjectManager doesn't return valid shared_ptr to this tile.
IllegalMove	- Any IllegalException can be thrown by a Tile or Worker if it breaks a placement rule.
NotEnoughSpace	(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]
```

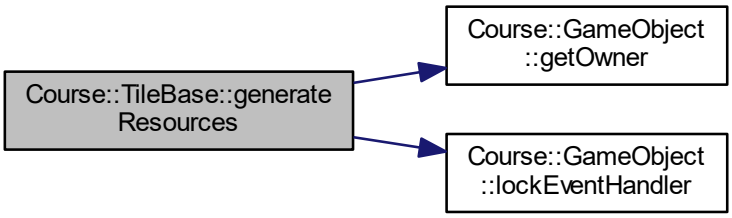
Sends information to the EventHandler on what resources were generated by this Tile.
.

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.

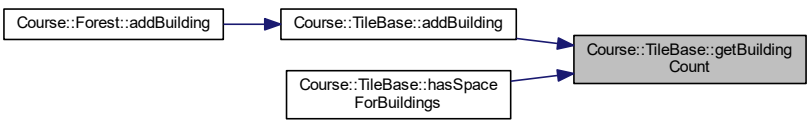
Returns

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.

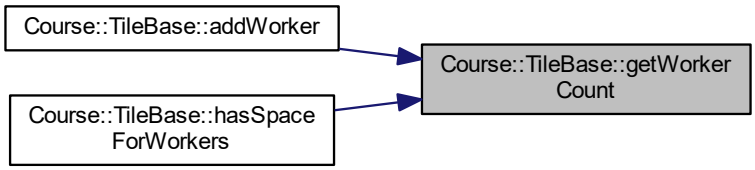
Returns

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

amount	Amount of buildingspace wanted.

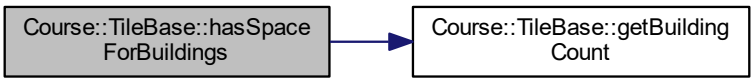
Note

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

amount	Amount of workerspace wanted.

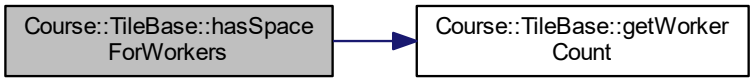
Note

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

building	A pointer to the Building-object being removed.

Postcondition

Exception guarantee: Basic

Exceptions

InvalidPointer	- 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

worker	A pointer to the Worker-object being removed.

Exceptions

InvalidPointer	- 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

manager	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

ToBeInTurn	

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

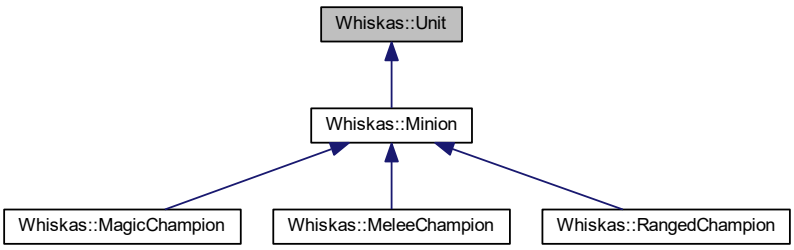
- E:/Gitin repo/ohjelmointi 3/whiskas/Game/turn.h
- E:/Gitin repo/ohjelmointi 3/whiskas/Game/turn.cpp

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

hasMoved	

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

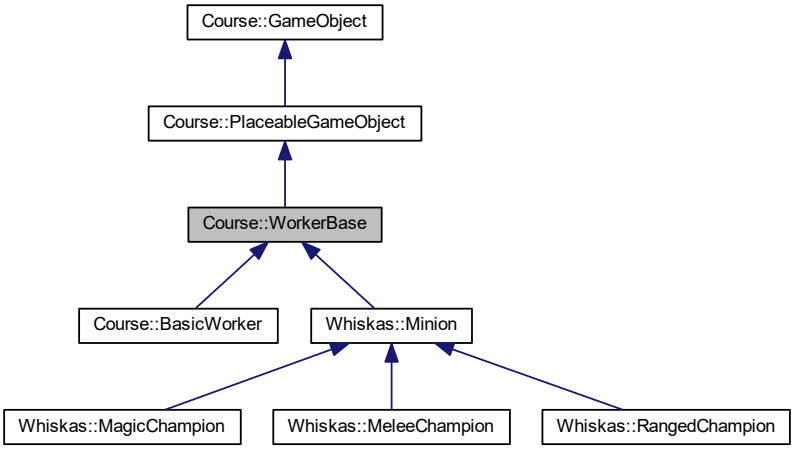
- E:/Gitin repo/ohjelmointi 3/whiskas/Game/minion/unit.h
- E:/Gitin repo/ohjelmointi 3/whiskas/Game/minion/unit.cpp

6.61 Course::WorkerBase Class Reference

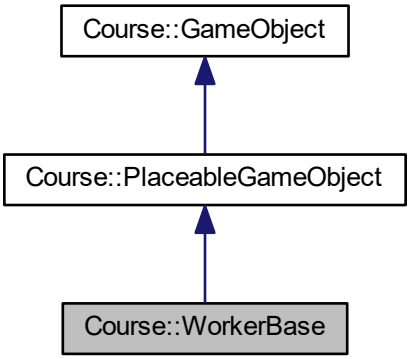
The WorkerBase class is an abstract base-class for Worker-objects.

```
#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

target	is the Tile that worker is being placed on.

Returns

True - Only if both conditions are met.

Postcondition

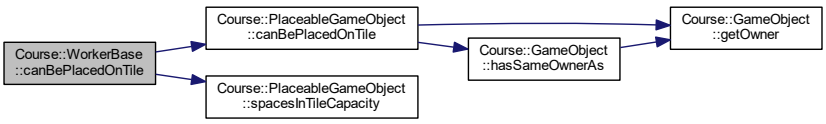
Exception guarantee: Basic

Note

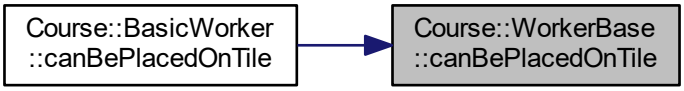
Override to change placement rules for derived worker.

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()

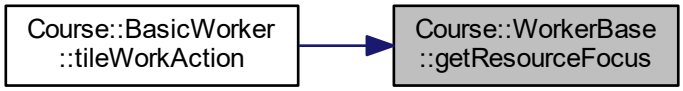
```
BasicResource Course::WorkerBase::getResourceFocus ( ) const  [final], [virtual]
```

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

new_focus	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

weight	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

size_x	is the horizontal size of the map area.
size_y	is the vertical size of the map area.
seed	is the seed-value used in the generation.
objectmanager	points to the ObjectManager that receives the generated Tiles.
eventhandler	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